Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Arduino based Crop Dryer

Arjun S, Malavika A P, Siva Suresh, Sreesha S Kumar
Bachelor Of Technology
College Of Engineering Attingal
Department of Electrical and Electronics Engineering

Abstract - Conventional sources of energy contribute significantly to environmental pollution and global warming, creating an urgent need for clean and sustainable alternatives. This project presents the design and development of a smart, hybrid solar crop dryer system intended to improve drying efficiency, reduce post-harvest losses, and minimize dependence on conventional energy sources. The proposed system utilizes an advanced heat sink–assisted drying mechanism to ensure uniform heat distribution and controlled moisture removal for various agricultural products. A multi-crop mode selection feature is integrated using an LCD/touch display mounted on the exterior of the dryer. Pre-programmed drying presets for crops such as rice, chilli, coconut, banana, and spices allow users to select optimal temperature and time settings easily, ensuring product quality and energy efficiency. To enhance reliability and user safety, the system incorporates an intelligent fault detection and protection unit, capable of identifying over-temperature conditions, fan failure, and short-circuit faults. Automatic safety actions and alerts help prevent damage to the system and the stored crops. The dryer operates on a arduino based energy system. Furthermore, the system is integrated with IoT-based monitoring and control, replacing conventional Bluetooth methods. Real-time data on temperature, humidity, energy status, and system faults can be monitored and controlled remotely through a web or mobile application. This futuristic and scalable approach demonstrates the potential of smart renewable-energy-based crop drying technology for modern agriculture. The proposed system supports sustainable farming practices, improves energy utilization, and contributes to the development of intelligent agri-infrastructure suitable for future smart agriculture systems

CHAPTER 1

INTRODUCTION

In agriculture, crop drying refers to the removal of moisture content from the crop either by keeping in sunlight or by means of some mechanical device. The method of sun drying has been widely put into practice since ancient times. Still this method is used by many people due to its simplicity and low cost. However, this method compromises with the quality of the crop. The crops dried in the open air are more prone towards the activity of insects, fungi etc. Also, the method is highly weather dependent, which compromises its reliability. It cannot be used during night or monsoons. Another disadvantage is that the crops are to be spread over as a single, thin layer, which consumes a lot of area. These problems have paved way to the development of industrial crop dryers.

Industrial crop dryers are mechanical devices used to effectively reduce the moisture content of the crop, to about 10-15% of the initial moisture content. Drying can be defined as the rate of moisture removal. As far as crops are concerned, there are two kinds of moisture content, unbound moisture and bound moisture. Unbound moisture refers to the moisture content present on the surface of the crop. Bound moisture is the moisture content present in the interior. When the crop is simply heated, it loses the unbound moisture content. However the bound moisture may still be retained. For optimum quality, the bound moisture content must be reduced to 10-15% of the initial moisture content. In order to ensure this, three parameters should be monitored and controlled, temperature, humidity and pressure drop. By carefully monitoring and controlling these parameters, optimum drying is achieved. This is the basic principle behind an industrial crop dryer. The system is microcontroller based. Parameter monitoring and automatic control is carried out by the controller. Sensors are used to collect or monitor parameter values, such as temperature, pressure and humidity. Heater and fans are used through relays as actuators. The microcontroller coordinates the working of sensors and actuators.

CHAPTER 2

OBJECTIVES OF PROJECT

1. To design an efficient crop drying system that reduces moisture content quickly and safely.
2. To improve drying speed compared to traditional sun drying methods.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

3. To maintain uniform drying across all parts of the crop to avoid spoilage.
4. To utilize heat sink technology for better heat storage and controlled heat release.
5. To reduce post-harvest losses caused by moisture, fungi, and bacteria.
6. To ensure energy efficiency by minimizing power consumption during operation.
7. To enable drying in all weather conditions, including rainy and cloudy days.
8. To improve the quality of dried crops (color, taste, and nutrients).
9. To develop a low-cost and user-friendly system suitable for farmers.
10. To incorporate automation (optional advanced feature) like temperature and humidity control for smart drying.
11. If you want, I can also give IEEE-style objectives, problem statement, or block diagram explanation for your project.

CHAPTER 3

IMPLEMENTATION OF PROJECT

1. Introduction to Implementation

The implementation of a crop dryer involves designing and integrating mechanical, electrical, and control systems to efficiently remove moisture from agricultural products. In this project, a heat sink-based drying system is developed to ensure uniform heat distribution and improved drying efficiency.
Traditional sun drying is highly dependent on weather conditions and is prone to contamination. Hence, this project provides a controlled drying environment, ensuring faster and hygienic drying.

2. System Architecture

The system is divided into the following major sections:

1. Heating Element for heating purpose.
2. Heat Distribution System (Heat Sink) for absorb and dissipate heat.
3. Drying Chamber, where the crops are placed for drying.
4. Sensing and Control Unit for monitors conditions and ensures safety.
5. Power Supply Unit is the backbone that keeps everything alive.

All these units are integrated to form a compact and efficient drying system.

3. Detailed Implementation

3.1 Drying Chamber Design

The drying chamber is the main enclosure where crops are placed for drying.

- Constructed using**aluminum, or insulated metal sheets**
- Designed as a **closed rectangular box**
- Inner walls are insulated to **reduce heat loss**
- A **transparent glass/acrylic cover** can be used for monitoring
- Holds crops in a controlled environment.

3.2 Heating Unit Implementation

The heating unit is responsible for generating heat required for drying.

- A **nichrome wire heating coil** is used
- Powered by **AC supply (230V)**
- Mounted at the bottom or side of the chamber

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Heat Sink Role:

- Attached to the heating element
- Distributes heat evenly
- Prevents overheating of specific areas
- Improves energy efficiency

3.3Uniform heat distribution

A Gear motor is responsible for uniform heat distribution by the movement of the motor.

- Powered by DC supply (12V)
- Mounted at the bottom of the chamber.
- Connected to the chamber so that the motion of the motor can cause a movement of the crop and get even heat distribution.

3.4 Sensor Integration

Sensors are used for monitoring system conditions.

- DHT11 Temperature and Humidity Sensor.
- Operating voltage: 3.3V – 5V.
- The DHT11 sensor continuously measures the ambient temperature and sends the data to the controller.

3.5 Control System Implementation

A microcontroller (Arduino) is used for automation. An Arduino board(ESP32), 5 VRelay module(10A, 230V) and a LCD display combined to form the control system to perform the controlling purpose

- Reads temperature from sensor
- The data from the sensor Compares with preset value and Controls heater using relay
- If the Temperature < Set Value the Heater ON
- If Temperature ≥ Set Value the Heater OFF
- This ensures safe and controlled drying process**.**

3.7 Assembly and Integration

All components are assembled carefully:

- Heating coil fixed securely at the bottom of the chamber.
- Heat sink attached properly
- Motor aligned for even heat distribution
- Sensors placed at correct positions
- Wiring insulated and protected

4. Testing and Performance Evaluation

After assembling the setup, the test conducted:

- Loaded the crops (grains, fruits, vegetables). Set the desired temperature (40°C–60°C)
- Run the system and monitor
- Drying time reduced compared to sun drying
- Uniform drying achieved
- No contamination observed

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

CHAPTER 4

KEY FEATURES

1.Automated Temperature Control

- Temperature is from35 to 50°C via Arduino and DHT11sensor.

2. Real-Time Monitoring

- The LCD display shows temperature, humidity, drying status. It gives the current status of the system.

3. Efficient Drying

- Uniform heat distribution all over the crops, faster moisture removal (reduces time by ~50%).

4. Customizable Settings

- Programmable drying time and temperature via Arduino

5. Crop Quality

- Preserves nutritional value with controlled drying

6. Multi-Crop Compatibility

- Adjustable settings for different crops (e.g., grains, herbs, fruits)

7. Data Logging

- Optional feature to track drying parameters over time

8.Compact Design

- Portable and space-efficient for small farms or businesses

9. Low Maintenance

- Simple, durable components (Arduino, sensors, heater)

10. Safety Features

- Overheat protection, automatic shut-off

11. Cost-Effective:

- Affordable compared to industrial dryers, DIY-friendly

12.Remote Monitoring

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Potential for IoT integration (e.g., Wi-Fi alerts)

CHAPTER 5

LITERATURE SURVEY

[1]. An Arduino-based rice grain dryer was developed to maintain a controlled temperature range of 50–60°C. The system utilized temperature sensors and relay-controlled heating elements to ensure uniform drying. The results demonstrated reduced drying time and improved grain quality compared to traditional sun drying methods. However, the system was primarily designed for rice and lacked adaptability for other crops.

[2]. An automated coffee bean dryer using Arduino Uno and DHT11 sensors was proposed. The system continuously monitored temperature and humidity and controlled heating elements accordingly. This approach improved drying speed and reduced moisture content effectively, although it resulted in higher energy consumption.

[3]. A seaweed drying system incorporating Arduino and an OLED display was developed for real-time monitoring. The use of forced air circulation through fans ensured uniform heat distribution. The system performed well in humid and rainy conditions but was limited to small-scale applications.

[4]. A smart grain dryer using multiple sensors such as LDR, rain sensor, temperature sensor, and real-time clock (RTC) was introduced. The system automatically adjusted drying parameters based on environmental conditions, enhancing automation and reducing manual intervention. However, the increased number of components led tohigher system complexity and cost.

[5]. An Arduino-based moringa leaf dryer demonstrated efficient moisture reduction from approximately 70% to below 10%. The system employed humidity-based control of fans and heaters, preserving the nutritional quality of the leaves. However, the system was more suitable for leafy crops than for grains.

[6]. A fuzzy logic-based drying system was implemented to improve temperature regulation. The fuzzy controller provided smoother control compared to conventional ON/OFF methods, reducing energy consumption and improving drying uniformity. The complexity of implementing fuzzy logic algorithms was a limitation.

[7]. A solar-powered crop dryer integrated with Arduino was proposed to reduce dependence on electrical energy. The system utilized solar collectors along with sensors for monitoring temperature and humidity. While the system was environmentally friendly, its performance depended heavily on sunlight availability.

[8]. An IoT-based smart dryer system using Arduino and ESP8266 Wi-Fi module enabled remote monitoring and control through mobile applications. This system allowed farmers to track drying conditions in real time, improving usability. However, it required stable internet connectivity.

[9]. A hybrid solar-electric drying system was designed to ensure continuous operation. The Arduino controller managed switching between solar and electrical energy sources, maintaining optimal drying conditions. The system improved reliability but increasedoverall cost.

[10]. A PID-controlled drying system was developed to achieve precise temperature control. The PID controller minimized temperature fluctuations and improved system stability. However, proper tuning of PID parameters was necessary for optimal performance.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

[11]. A conveyor belt-based drying system was introduced for continuous drying applications. The Arduino controlled motor speed and temperature, ensuring uniform exposure of crops to heat. This system was effective for large-scale operations but unsuitable for small farmers due to its complexity.

[12]. A heat pump-based dryer was proposed to improve energy efficiency by recycling heat. Arduino was used for monitoring and controlling system parameters. Although the system significantly reduced energy consumption, it involved high initial investment.

[13]. A microwave-assisted drying system was developed to achieve rapid moisture removal. Arduino controlled the power levels and duration of operation. The system reduced drying time considerably but required careful control to prevent overheating.

[14] An infrared drying system was implemented using Arduino to control IR emitters.This method provided faster and more uniform heating compared to conventional techniques. However, the system required precise control to avoid surface overheating.

[15]. A tray dryer with forced air circulation was designed using Arduino. Fans distributed heated air evenly across multiple trays, improving drying uniformity. This method was simple and effective for small-scale applications.

[16]. A moisture sensor-based automatic dryer was developed to stop the drying process once the desired moisture level was achieved. This approach prevented over-drying and improved energy efficiency.

[17]. A GSM-based monitoring system was introduced to send SMS alerts regarding drying status. This system was useful in remote areas without internet connectivity butlacked advanced control features.

[18]. An AI-based smart drying system utilized machine learning algorithms to predict drying time and optimize parameters. Arduino was used for data acquisition. Although highly efficient, the system was complex and expensive.

[19]. A portable mini crop dryer was designed for small-scale farmers using low-cost components and Arduino control. The system was affordable and easy to use but had limited drying capacity.

[20]. An industrial-scale drying system using PLC automation was developed for bulk crop processing. Compared to Arduino-based systems, PLC systems offered higher reliability and scalability but at a significantly higher cos

CHAPTER 6

COMPONENTS REQUIRED

6.1 MICROCONTROLLER (ESP32)

The microprocessor is capable of all kinds of data processing, such as arithmetic and logical operations. It consists of the processor only, and memory, I/O devices and all additional requirements must be interfaced externally. The microcontroller, on the other hand, is a single chip which contains the processor, memory and I/O devices. It is similar to a computer on a chip. It is easier to handle and makes the system concise. Due to these advantages, microcontroller is selected.
ESP32 is a powerful 32-bit microcontroller with built-in WiFi and Bluetooth capabilities, widely used for IoT and embedded systems.In this system, ESP32 reads temperature from DHT11 sensor. Then Controls the relay module. The Communicates is with LCD display via I2C. It receives temperature setpoint via Bluetooth device.

Key Features

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- 32-bit dual-core Tensilica processor
- Operating voltage: 3.3V
- Clock speed: up to 240 MHz
- Built-in WiFi (802.11 b/g/n) and Bluetooth Classic + B

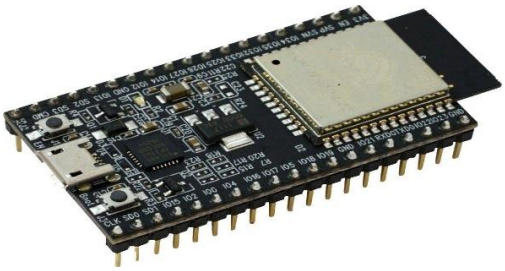


Fig 7.1

Technical Specification

Microcontroller	ESP32
CPU	Dual-coreXtensa LX6
Clock Speed	Upto 240 MHz
Operating Voltage	2.2v-3.6v(typically3.3v)
Flash Memory	4MB (external,varies by module)
SRAM	520 KB
Wifi	802.11b/g/n
Bluetooth	Bluetooth v4.2(classic+BLE)
Operating Temperature	-40°Cto+125°C
Power Consumption	Ultra-low Power modes available

Table 8.1

6.2 Pin Diagram

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

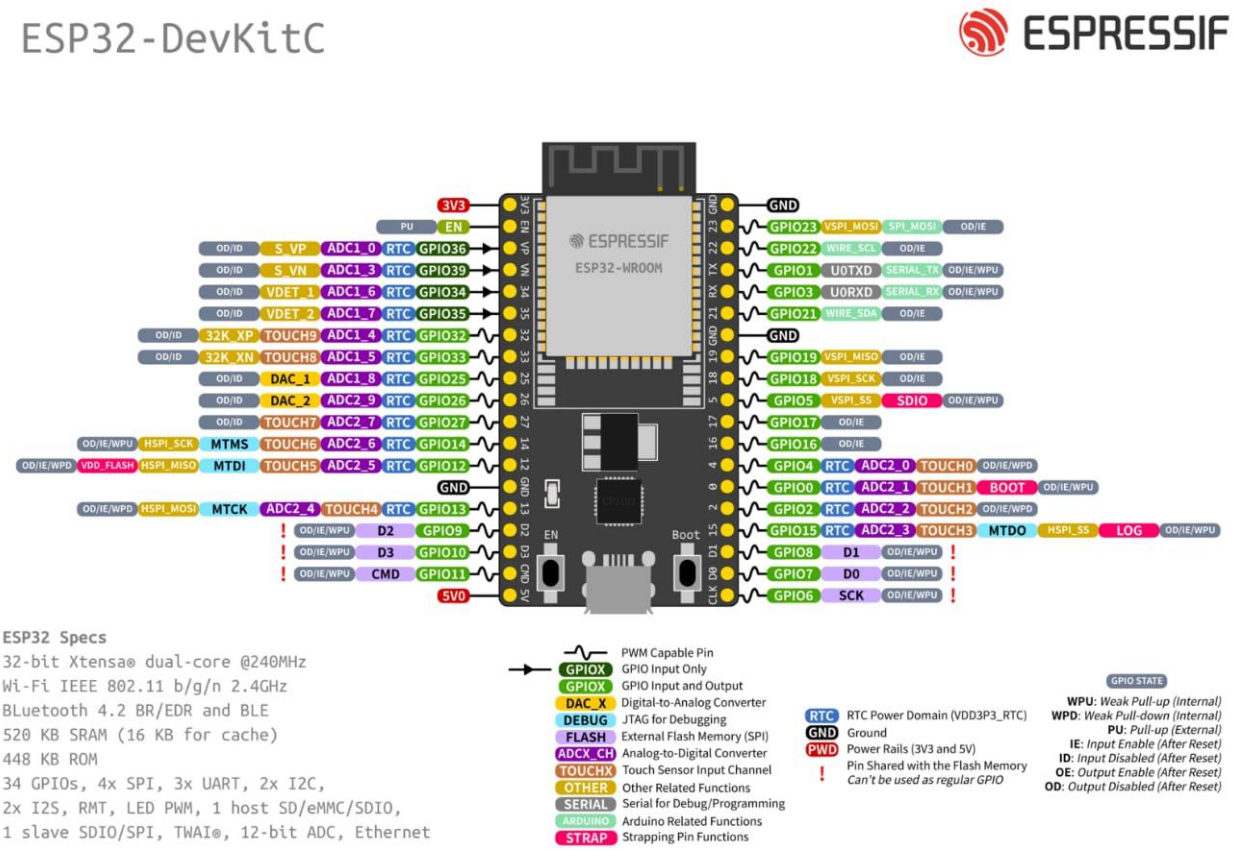


Fig 7.2

ESP32 Pin Specifications (Important Points)

1. Power Pins

- 3V3 → 3.3V output
- VIN → 5V input (from USB/external supply)
- GND → Ground

---

2. GPIO Pins (General Purpose Pins)

- Used for input/output operations
- Example: GPIO2, GPIO4, GPIO5, GPIO18, etc.
- Can be used for:
    - Digital input
    - Digital output
    - Interrupts

---

3. ADC Pins (Analog Input)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Converts analog signal to digital
- Resolution: 12-bit
- Voltage range: 0–3.3V

ADC1 (Important – safe to use)

- GPIO32, 33, 34, 35, 36, 39

ADC2 (avoid when WiFi is ON)

- GPIO0, 2, 4, 12–15, 25–27

---

4. DAC Pins (Analog Output)

- Provides analog voltage output
- Pins:
    - GPIO25
    - GPIO26

---

5. PWM Pins

- Used for motor speed control, LED brightness
- Almost all GPIO pins support PWM

---

6. Communication Pins

UART (Serial Communication)

- TX → GPIO1
- RX → GPIO3

I2C (LCD, Sensors)

- SDA → GPIO21
- SCL → GPIO22

SPI (High-speed communication)

- MOSI → GPIO23
- MISO → GPIO19
- SCK → GPIO18
- CS → GPIO5

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

---

7. Input Only Pins

- GPIO34, 35, 36, 39
- Cannot be used as output

---

8. Special Pins (Important for Exams)

- EN (Enable) → Reset pin
- GPIO0 → Boot mode selection
- GPIO2 → Used during boot
- GPIO15 → Boot configuration

---

9. Pins Used in Your Project

- GPIO4 → DHT11 sensor
- GPIO21 & 22 → LCD (I2C)
- GPIO26 → Relay (heater/motor)

6.2 SENSOR

DHT11 Temperature and Humidity Sensor

For automatic control of the parameters, it is essential that the parameter values are received continuously. Here the parameters to be controlled are temperatureand humidity. The microcontroller requires the values of these parameters so as to take the appropriate controlling action. Sensor ICs are directly connected to the microcontroller input and are thus capable of direct and reliable communication with microcontroller. DHT11 is a low-cost digital sensor used for measuring temperature and humidity.The sensor continuously measures the ambient temperature and sends the data to the ESP32 for monitoring and control.

Key Features

- Operating voltage: 3.3V – 5V
- Temperature range: 0°C to 50°C
- Temperature accuracy: ±2°C
- Humidity range: 20% – 90% RH

Digital signal output

Sampling rate: 1 reading per second

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

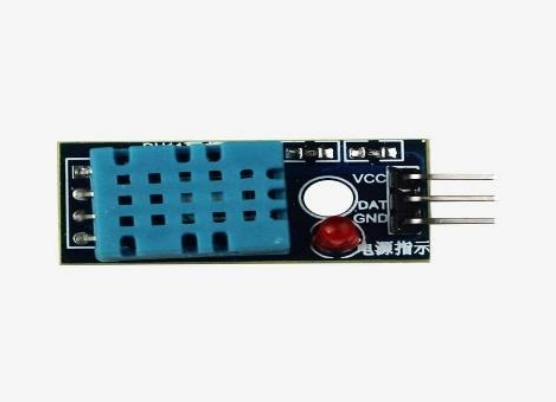


Fig 7.3

Pin Decribing

Pin 1 – VCC

- Connect to **3.3V or 5V** power supply

Pin 2 – DATA

- Outputs temperature & humidity data
- Connect to a microcontroller (e.g., ESP32 / Arduino)
- Requires a **10kΩ pull-up resistor** to VCC

Pin 3 – NC (Not Connected)

- Leave this pin unused

Pin 4 – GND

- Connect to ground

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Detail Specification

Parameter	Condition	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25°C		±4%RH	
	0-50°C			±5%RH
Interchangeability	Fully interchangeable			
Measurement Range	0	30%RH		90%RH
	25°C	20%RH		90%RH
	50°C	20%RH		80%RH
Response time (second)	1/e(63%) 25°c 1M/S Air	6s	10s	15s
Hysteresis			±1%RH	
Long term stabililty	Typical		±1%RH/year	
Temperature				
Resolution		1°C	1°C	1°C
		8 Bit	8 Bit	8 Bit

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Repeatability			±1°C	
Accuracy		±1°C		±2°C
Measurement Range		0		50°
Response time(second)	1/e(63%)	6s		30s

Table 8.2

6.35V Relay Module (10A, 230V)

heater is connected to the output pins of the microcontroller through relay ICs. Relays are basically electrically operated switches. They serve the purpose of switching ON/OFF the heater and fans depending upon the microcontroller output signal. As the switching operation is carried out by sending a high/low signal, the relays are connected to the digital pins of the controller. The output of the microcontroller serves as an input to the relay. If the controller sends a high to a particular relay, then the device connected to the corresponding relay is turned ON. If the signal is low, the device is turned OFF.

Key Specifications

- Control voltage: 5V
- Load capacity: 10A at 230V AC
- Isolation between control and load
- Normally Open (NO) and Normally Closed (NC) contacts

Fig 7.4

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

6.4 I2C COMMUNICATION INTERFACE

I2C (Inter-Integrated Circuit) is a two-wire serial communication protocol used for connecting multiple devices to a microcontroller.I2C is used to communicate between ESP32 and the LCD display, reducing the number of required pins.

Lines Used

SDA -Serial Data(GPIO21)
SCL -Serial Clock(GPIO22)

Features

- Only two wires required
- Supports multiple devices
- Address-based communication
- Speeds up to 400 kbps

6.5 16×2 LCD DISPLAY

A 16×2 LCD is a character display capable of showing 16 characters per line and 2 lines.The LCD showsCurrent Temperature, Target Temperature and Heater Status (ON/OFF).

Key Features

- Operating voltage: 5V
- Display format: 16 columns × 2 rows
- Uses HD44780 controller
- Can operate in 4-bit or 8-bit mode
- With I2C module, only 2 communication lines are required

Fig 7.5

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

6.6 12V , 60rpm DC geared motor

A 12V, 60 RPM DC geared motor is used to drive the movement of the mesh conveyor in the crop dryer system. This motor is specially selected because it provides low-speed, high-torque rotation, which is ideal for smoothly moving the mesh carrying the crops without causing damage. The 12V supply makes it compatible with common power sources and microcontroller-based systems, ensuring easy integration with controllers like ESP32 or Arduino. The built-in gearbox reduces the motor speed to 60 revolutions per minute while increasing torque, allowing it to handle the load of wet crops effectively. This controlled movement ensures uniform exposure of crops to heat, resulting in efficient and consistent drying. Additionally, the motor operates with low power consumption and high reliability, making it suitable for continuous operation in agricultural applications.

Fig 7.6

6.7 Heating Element (750-volt nichrome)

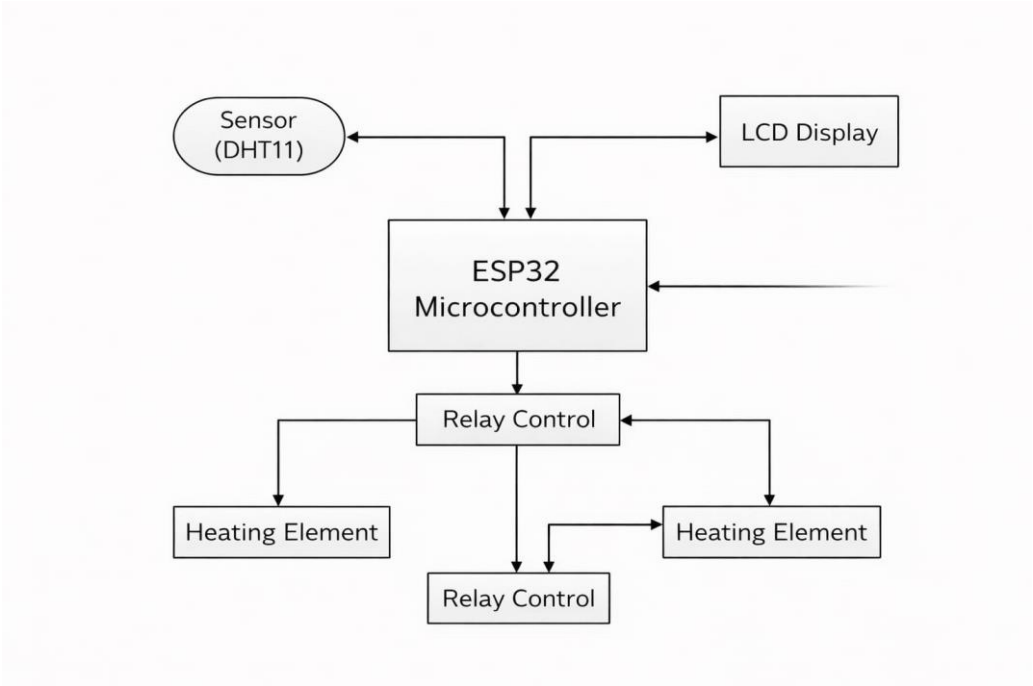
A 750-volt nichrome heating element is used in the crop dryer system as the main source of heat generation. Nichrome, an alloy of nickel and chromium, is widely preferred for heating applications due to its high electrical resistance, excellent heat-producing capability, and strong resistance to oxidation at high temperatures. When electrical current passes through the element, it converts electrical energy into heat based on the principle of resistive heating. The 750-volt rating ensures that the element can operate efficiently under high-voltage conditions, producing sufficient heat required for drying agricultural products. This helps in removing moisture from crops in a controlled manner, improving drying speed and preserving quality. The element is typically controlled using a relay module connected to the microcontroller, allowing automatic switching based on temperature readings from the DHT11 sensor, thereby ensuring safe and energy-efficient operation of the crop dryer system

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Fig 7.8

CHAPTER 8

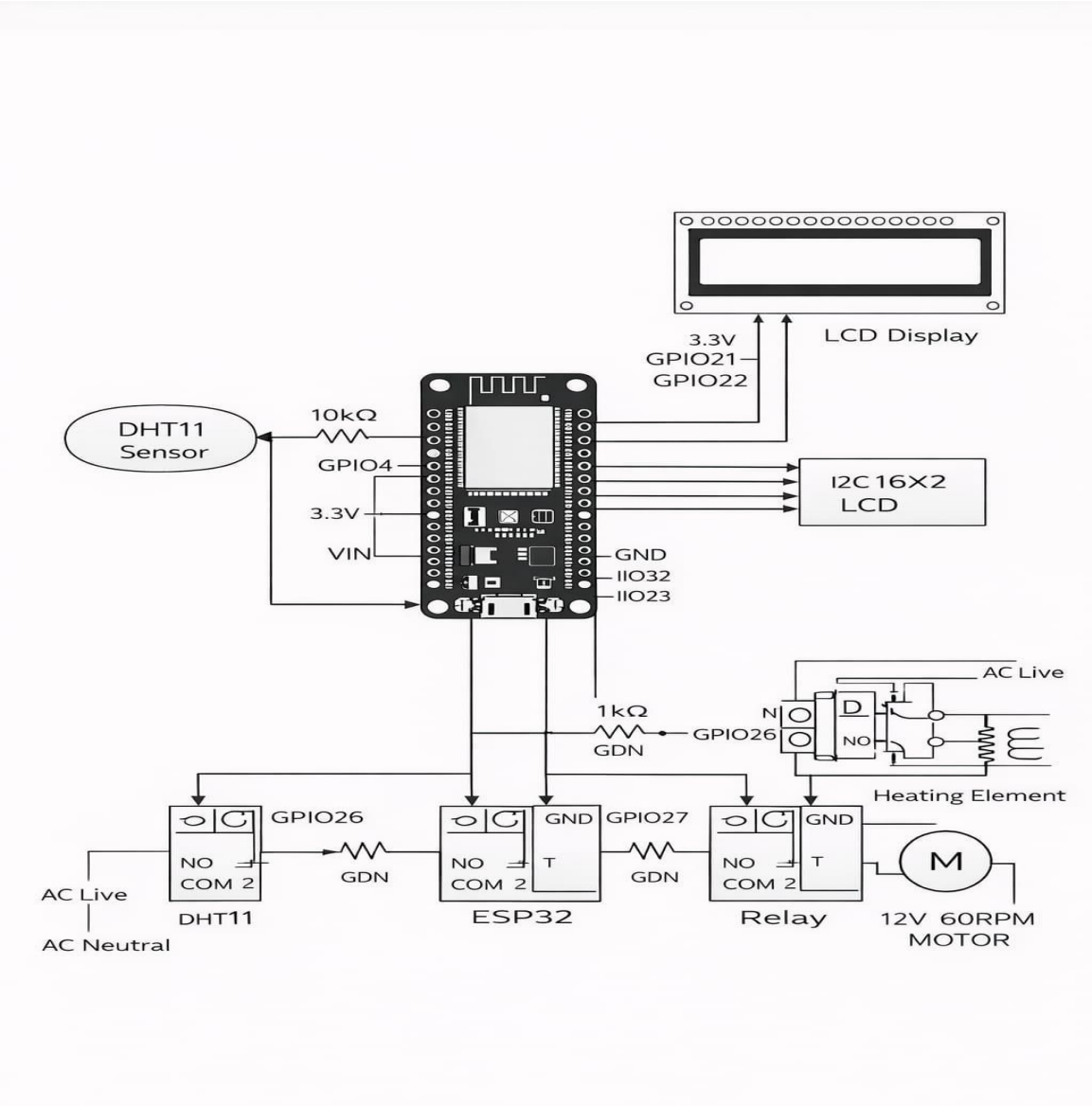
BLOCK DIAGRAM



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

CHAPTER 9

CIRCUIT DIAGRAM



CHAPTER 10

WORKING

The project aims to control the speed of a Permanent Magnet Synchronous Motor (PMSM) using the Space Vector Pulse Width Modulation (SVPWM) technique. The system combines both power electronics and control logic to achieve smooth and efficient operation of the motor.

1. Basic Principle

A PMSM is a type of synchronous motor in which the rotor magnetic field is produced by permanent magnets instead of windings. For the motor to run efficiently, the stator current must rotate synchronously with the rotor magnetic field.This is achieved by providing a three-phase AC supply that has the same frequency as the rotor's synchronous speed.Since the input source is DC, an inverter circuit is required to convert this DC power into a three-phase AC output.The inverter switching is controlled by the SVPWM algorithm, which ensures that the generated AC supply is nearly sinusoidal and efficiently utilizes the DC bus voltage.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

2. Power Conversion Stage

The DC supply (from a power source or battery) is first passed through a filter capacitor to remove ripples and provide a stable DC link voltage.This DC link is connected to a three-phase voltage source inverter (VSI) built using six power switches (MOSFETs or IGBTs).Each switch pair corresponds to one phase of the motor.By turning these switches ON and OFF in a specific sequence, the inverter generates a three-phase AC output that drives the PMSM.

3. SVPWM Control Strategy

The Space Vector Pulse Width Modulation (SVPWM) technique is used to determine the exact switching pattern of the inverter.

- The control algorithm calculates a reference voltage vector that represents the desired phase voltages needed to produce the required motor speed and torque.
- The space vector diagram divides the plane into six sectors, each corresponding to a combination of inverter switch states.
- Depending on which sector the reference vector lies in, the controller calculates time durations (T1, T2, T0) for applying the active and zero voltage vectors during one PWM cycle.
- These time intervals are distributed symmetrically to generate smooth voltage waveforms.
- The final PWM pulses generated by the microcontroller are sent to the gate driver circuit, which amplifies them and drives the inverter switches.This process repeats rapidly (typically thousands of times per second), resulting in a smooth, sinusoidal-like three-phase voltage that efficiently drives the PMSM.

4. Closed-Loop Speed Control

The system uses a closed-loop control mechanism to maintain the desired motor speed even under varying loads.

- A speed sensor (optical encoder or Hall sensor) mounted on the motor shaft continuously measures the actual speed.
- The measured speed is fed back to the microcontroller, where it is compared with the reference speed (set by a potentiometer or input command).
- The error signal between the reference and actual speed is processed by a PI (Proportional–Integral) controller, which adjusts the voltage magnitude and frequency supplied to the motor.
- This updated voltage reference is then used by the SVPWM algorithm to modify the inverter's switching pattern, bringing the motor speed back to the desired value.

Through this continuous feedback and correction process, the motor maintains a stable and accurate speed even when load conditions change.

5. Generation of Gate Pulses

The microcontroller executes the SVPWM algorithm to compute the duty cycles for each inverter switch.The gate driver circuit (using IR2110 or similar IC) isolates the controller from the power circuit and provides sufficient current to switch the MOSFETs or IGBTs efficiently.These gate signals determine the inverter's output voltage pattern, controlling the amplitude and frequency of the three-phase output supplied to the PMSM

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

CHAPTER 10

PROGRAM

```
/**
 * ESP32 Drier Controller
 *
 * Features:
 *  - Receive target temperature via Bluetooth Classic (Serial)
 *  - Drive heating element relay until set temperature is reached
 *  - Display set/current temperature and status on I2C 16x2 LCD (PCF8574, 0x27)
 *
 * Wiring:
 *  DHT11  DATA    -> GPIO 4       (with 10k pull-up to 3.3V)
 *  Heater relay   -> GPIO 26      (active-HIGH relay module)
 *  LCD SDA        -> GPIO 21
 *  LCD SCL        -> GPIO 22
 *
 * Bluetooth:
 *  Device name: "ESP32_Drier"
 *  Send a numeric value (e.g. "45.0\n") to set the target temperature.
 *  Send "0\n" to disable heating.
 */
#include <Arduino.h>
#include <BluetoothSerial.h>
#include <DHT.h>
#include <LiquidCrystal_I2C.h>
// --- Pin definitions -------------------------------------------------------
#define HEAT_RELAY_PIN   26   // Heating element relay (active-HIGH)
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
#define DHT_PIN          4   // DHT11 data line
#define DHTTYPE          DHT11

// --- Tuneable parameters -----------------------------------------------
// Hysteresis band (deg C) to prevent relay chatter
#define HYSTERESIS       1.0f
// How often the sensor is polled (ms) - DHT11 needs >=1s; 2500ms adds margin
#define TEMP_INTERVAL    2500UL
// Maximum accepted set temperature (deg C) - DHT11 is rated to 50 deg C
#define MAX_SET_TEMP     50.0f
// Watchdog: turn heater off if no valid reading for this long (ms)
#define SENSOR_TIMEOUT   10000UL

// Relay convenience macros (active-HIGH logic)
#define RELAY_ON(pin)    digitalWrite(pin, HIGH)
#define RELAY_OFF(pin)   digitalWrite(pin, LOW)

// --- Objects -----------------------------------------------------------
BluetoothSerial   SerialBT;
DHT               dht(DHT_PIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);   // change 0x27 to 0x3F if LCD not found

// --- State -------------------------------------------------------------
float         setTemp        = 0.0f;     // target temperature (0 = disabled)
float         currentTemp    = -999.0f;  // last valid sensor reading
float         currentHumidity = -1.0f;    // last valid humidity reading (%RH)
bool          heatingOn      = false;
bool          sensorValid    = false;    // true after first successful DHT read
unsigned long lastValidReadMs = 0;       // millis() of last good DHT read (watchdog)
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
// Fixed char buffer avoids String heap fragmentation
char          btBuffer[17]    = {0};     // 16 chars max + null terminator
uint8_t       btBufIdx        = 0;
unsigned long lastBtCharMs    = 0;       // millis() of last BT char received
unsigned long lastTempMs      = 0;

// --- Relay control -------------------------------------------------------
void controlRelays() {
  // Disable if no set temperature
  if (setTemp <= 0.0f) {
    RELAY_OFF(HEAT_RELAY_PIN);
    heatingOn = false;
    Serial.println("RELAY: OFF - heating disabled (setTemp=0)");
    return;
  }

  // If sensor has not been read yet, turn relay ON since user set a target
  // (relay will be regulated once sensor starts reading)
  if (!sensorValid) {
    RELAY_ON(HEAT_RELAY_PIN);
    heatingOn = true;
    Serial.printf("RELAY: ON - setTemp=%.1f, sensor not ready yet (heating blind)\n", setTemp);
    return;
  }

  // Sensor is valid — use hysteresis control
  if (currentTemp < setTemp) {
    if (!heatingOn) {
      RELAY_ON(HEAT_RELAY_PIN);
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
      heatingOn = true;
      Serial.printf("RELAY: ON - T:%.1f < SetTemp:%.1f\n", currentTemp, setTemp);
    }
  } else {
    if (heatingOn) {
      RELAY_OFF(HEAT_RELAY_PIN);
      heatingOn = false;
      Serial.printf("RELAY: OFF - T:%.1f >= SetTemp:%.1f\n", currentTemp, setTemp);
    }
  }
}

// --- LCD update ------------------------------------------------------------
// Layout (16x2):
//   Row 0: "Set: 65.0*C  HTG"   set-point + relay status
//   Row 1: "T:63.8*C H:45.5%"   live temp + humidity
void updateLCD() {
  char buf[8];

  // Row 0: set-point + status
lcd.setCursor(0, 0);
  lcd.print("Set:");
  if (setTemp > 0.0f) {
dtostrf(setTemp, 5, 1, buf);
    lcd.print(buf);
  } else {
    lcd.print(" ----");
  }
  lcd.print((char)0xDF);  // ° symbol
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
  lcd.print("C");
  lcd.print("  ");
  if (heatingOn)      { lcd.print("HTG"); }
  else                { lcd.print("   "); }

  // Row 1: current temperature + humidity (exactly 16 chars)
lcd.setCursor(0, 1);
  lcd.print("T:");
  if (currentTemp > -100.0f) {
dtostrf(currentTemp, 4, 1, buf);
    lcd.print(buf);
  } else {
    lcd.print("----");
  }
  lcd.print((char)0xDF);
  lcd.print("C");
  lcd.print(" H:");
  if (currentHumidity >= 0.0f) {
dtostrf(currentHumidity, 4, 1, buf);
    lcd.print(buf);
  } else {
    lcd.print("----");
  }
  lcd.print("%");
}

// --- Bluetooth input handler ---------------------------------------------
// Uses a fixed char array to avoid heap fragmentation.
// Validates that the received string is numeric before parsing.
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
// Only accepts unsigned decimals - negative set-points are not valid.
static bool isNumericString(const char *s) {
  if (*s == '\0') return false;
  bool hasDot = false;
  for (; *s; s++) {
    if (*s == '.' && !hasDot) { hasDot = true; continue; }
    if (*s < '0' || *s > '9')  return false;
  }
  return true;
}

// Strip leading and trailing whitespace/control chars in-place.
// Handles apps that append \r, spaces, etc. e.g. "65.0\r\n" or " 65\n".
static void trimBuffer(char *s, uint8_t &len) {
  // Trim trailing
  while (len > 0 && (uint8_t)s[len - 1] <= 0x20) { s[--len] = '\0'; }
  // Trim leading
  uint8_t start = 0;
  while (start < len && (uint8_t)s[start] <= 0x20) { start++; }
  if (start > 0) {
    len -= start;
memmove(s, s + start, len + 1);  // +1 to move null terminator too
  }
}

// Process the BT buffer contents — called on newline OR timeout
void processBtBuffer() {
  if (btBufIdx == 0) return;
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
  btBuffer[btBufIdx] = '\0';
  Serial.printf("BT-CMD: raw buffer [%s] len=%d\n", btBuffer, btBufIdx);
trimBuffer(btBuffer, btBufIdx);
  Serial.printf("BT-CMD: trimmed buffer [%s] len=%d\n", btBuffer, btBufIdx);

  if (btBufIdx == 0) {
    // Was all whitespace
  } else if (!isNumericString(btBuffer)) {
    Serial.printf("BT-ERR: rejected [%s] - not numeric\n", btBuffer);
    if (SerialBT.hasClient()) SerialBT.println("ERR: send a number (e.g. 45.0)");
  } else {
    float val = atof(btBuffer);
    Serial.printf("BT-PARSE: atof result = %.2f\n", val);

    if (val >= 0.0f && val <= MAX_SET_TEMP) {
      setTemp = val;
      char msg[48];
      if (val > 0.0f) {
snprintf(msg, sizeof(msg), "Target set to %.1f C", setTemp);
      } else {
snprintf(msg, sizeof(msg), "Heating disabled");
      }
      Serial.printf(">>> SET TEMP = %.1f, Current = %.1f, SensorValid = %s\n",
                    setTemp, currentTemp, sensorValid ? "YES" : "NO");
      if (SerialBT.hasClient()) SerialBT.println(msg);
      Serial.println(msg);
  controlRelays();
  updateLCD();
    } else {
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
      Serial.printf("BT-ERR: value %.1f out of range (0-%.0f)\n", val, MAX_SET_TEMP);
      char msg[48];
 snprintf(msg, sizeof(msg), "Invalid. Range: 0-%.0f C", MAX_SET_TEMP);
      if (SerialBT.hasClient()) SerialBT.println(msg);
    }
  }

  // Reset buffer
 memset(btBuffer, 0, sizeof(btBuffer));
  btBufIdx = 0;
  lastBtCharMs = 0;
}

 void handleBluetooth() {
  while (SerialBT.available()) {
    char c = (char)SerialBT.read();

    // Log every raw byte for debugging
    Serial.printf("BT-RAW: char='%c' hex=0x%02X btBufIdx=%d\n", (c >= 0x20 ? c : '?'), (uint8_t)c, btBufIdx);

    if (c == '\n' || c == '\r') {
 processBtBuffer();
    } else {
      if (btBufIdx < (sizeof(btBuffer) - 1)) {
        btBuffer[btBufIdx++] = c;
        lastBtCharMs = millis();
      }
    }
  }
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
  // Timeout: if chars arrived but no newline within 500ms, process anyway
  if (btBufIdx > 0 && lastBtCharMs > 0 && (millis() - lastBtCharMs) >= 500) {
    Serial.println("BT-TIMEOUT: no newline received, processing buffer anyway");
processBtBuffer();
  }
}

// --- Setup ---------------------------------------------------------------
void setup() {
  Serial.begin(115200);
  Serial.println("ESP32 Drier Controller starting...");

  // Relays off immediately
pinMode(HEAT_RELAY_PIN, OUTPUT);
  RELAY_OFF(HEAT_RELAY_PIN);

  // Temperature / humidity sensor
dht.begin();
  Serial.println("DHT11 initialised.");
delay(2000);  // Give DHT11 time to warm up before first read

  // LCD
lcd.init();
lcd.backlight();
lcd.setCursor(0, 0);
  lcd.print("Drier Controller");  // exactly 16 chars [M5]
lcd.setCursor(0, 1);
  lcd.print(" Initialising.. ");
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
  delay(1500);
  lcd.clear();
  updateLCD();

   // Bluetooth - Classic SPP (NOT BLE).
   // Pair via Android Settings > Bluetooth, then use a
   // "Serial Bluetooth Terminal" app to send commands.
   if (!SerialBT.begin("ESP32_Drier", false /isMaster=false, device acts as slave/server/)) {
     Serial.println("FATAL: Bluetooth init failed!");
  lcd.clear();
  lcd.setCursor(0, 0);
     lcd.print("BT INIT FAILED! ");
  lcd.setCursor(0, 1);
     lcd.print("Check build_flags");
     while (true) { delay(1000); }
   }
   Serial.println("Bluetooth ready - device: ESP32_Drier");
 }

 // --- Loop ------------------------------------------------------------------
 void loop() {
   // Poll temperature on a fixed interval (non-blocking)
   unsigned long now = millis();
   if (now - lastTempMs >= TEMP_INTERVAL) {
     lastTempMs = now;

     float t = dht.readTemperature();   // deg C
     float h = dht.readHumidity();      // %RH
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
    Serial.printf("DBG: DHT11 read - T=%f H=%f isnan(T)=%d isnan(H)=%d\n", t, h, isnan(t), isnan(h));

    if (!isnan(t) && !isnan(h)) {
      currentTemp     = t;
currentHumidity  = h;
      sensorValid     = true;
lastValidReadMs  = now;
      Serial.printf("✓ Sensor OK: T=%.1f°C H=%.1f%%\n", currentTemp, currentHumidity);
    } else {
      Serial.println("✗ WARN: DHT11 read failed - check wiring and 10k pull-up on GPIO 4");
      // Watchdog: if sensor silent too long, cut power to heater
      if (sensorValid && (now - lastValidReadMs) >= SENSOR_TIMEOUT) {
        sensorValid = false;
        Serial.println("✗ ERR: Sensor timeout - heater disabled for safety");
        if (SerialBT.hasClient()) SerialBT.println("ERR: Sensor timeout - heater OFF");
      }
    }

controlRelays();
updateLCD();

    // Periodic status over Bluetooth
    if (SerialBT.hasClient()) {
      char status[80];
      if (sensorValid) {
snprintf(status, sizeof(status),
               "T:%.1fC H:%.1f%% Set:%.1fC Heat:%s",
               currentTemp, currentHumidity, setTemp,
heatingOn ? "ON" : "OFF");
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
    } else {
snprintf(status, sizeof(status), "Sensor not ready. Set:%.1fC", setTemp);
    }
    SerialBT.println(status);
  }
  // Always log to Serial for debugging
  Serial.printf("T:%.1fC H:%.1f%% Set:%.1fC Heat:%s Valid:%s\n",
                currentTemp, currentHumidity, setTemp,
                heatingOn   ? "ON"  : "OFF",
sensorValid ? "YES" : "NO");
 }

handleBluetooth();
}
```

CHAPTER 11

ALGORITHM

Step 1: Start

Step 2: Initialize Serial communication.

Step 3: Initialize heater relay pin as OUTPUT.

Step 4: Initialize DHT11 sensor.

Step 5: Initialize I2C LCD display.

Step 6: Initialize Bluetooth with device name “ESP32_Drier”.

Step 7: Turn OFF the heater initially.

Step 8: Declare loop() function for continuous operation.

Step 9: Read temperature and humidity from DHT11 sensor.

Step 10: Check whether sensor reading is valid.

If valid, store values.

Otherwise go to Step 29.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Step 11: Check whether Bluetooth data is available.

Step 12: Read received data into buffer.

Step 13: Remove unwanted characters (trim input).

Step 14: Check whether input is numeric.

If false, display error and go to Step 29.

Step 15: Convert input to temperature value (setTemp).

Step 16: Check whether setTemp is within range (0–50°C).

If false, display error and go to Step 29.

Step 17: Check whether setTemp = 0.

If true, turn OFF heater and go to Step 29.

Step 18: Compare current temperature with setTemp.

Step 19: If currentTemp < setTemp → Turn ON heater.

Step 20: If currentTemp ≥ setTemp → Turn OFF heater.

Step 21: Display set temperature on LCD.

Step 22: Display current temperature on LCD.

Step 23: Display humidity on LCD.

Step 24: Display heater status (HTG/ OFF) on LCD.

Step 25: Send system status via Bluetooth.

Step 26: Print system values in Serial Monitor.

Step 27: Check sensor timeout condition.

If sensor not responding for long time → Turn OFF heater.

Step 28: Apply hysteresis control to avoid frequent ON/OFF switching.

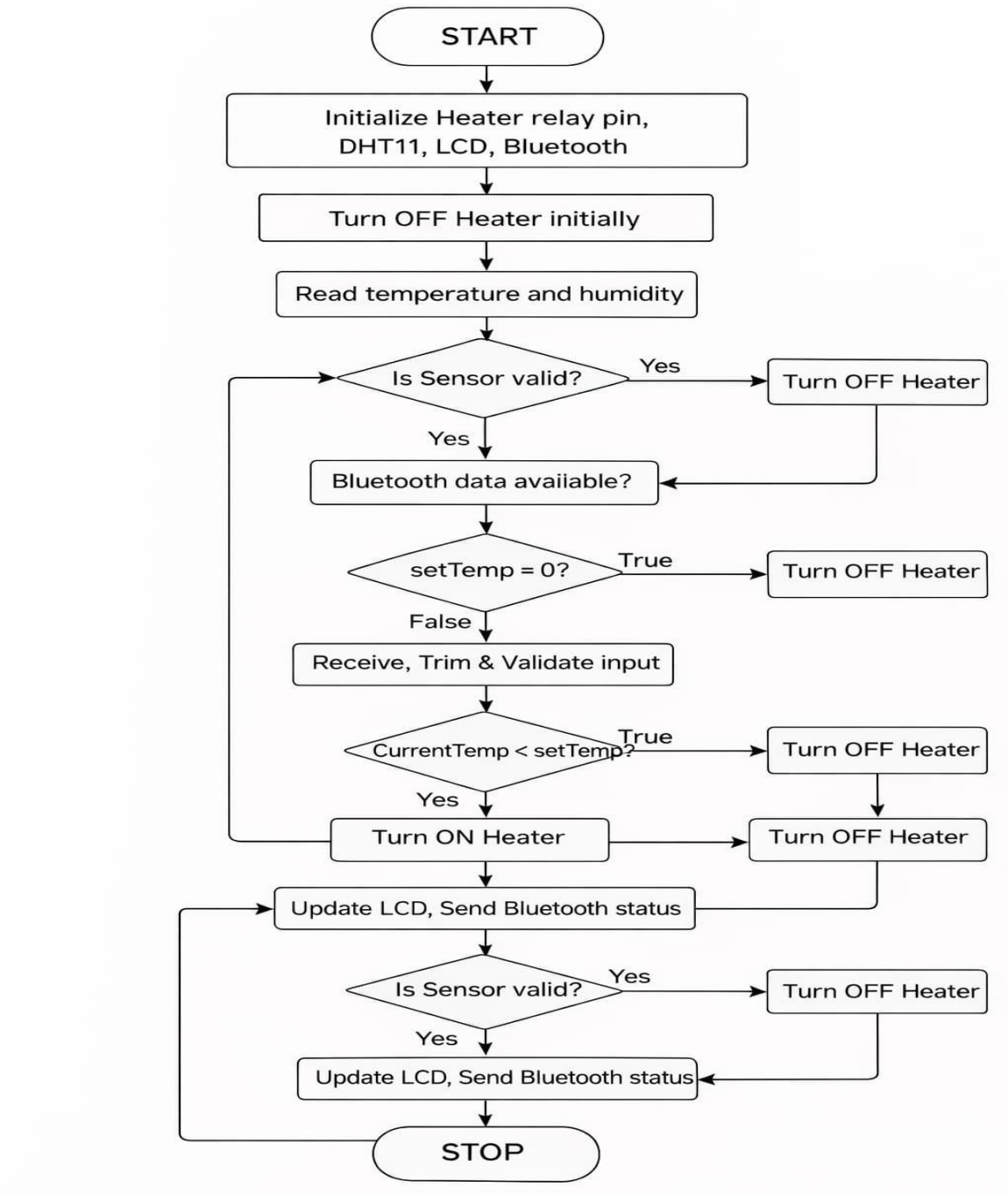
Step 29: Wait for next reading interval.

Step 30: Go back to Step 9 (repeat continuously

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

CHAPTER 12

FLOW CHART



CHAPTER 13

RESULTS AND DISCUSSION

The simulation and experimental results demonstrate the effectiveness of the proposed SVPWM-based speed control scheme for PMSM. The speed tracking performance, torque response, and current/voltage waveforms are analyzed and discussed below.

Simulation Results

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The simulation results are obtained using MATLAB/Simulink with the following parameters: PMSM rated power = 1.5 kW, rated speed = 1500 rpm, rated torque = 10 Nm, and switching frequency = 10 kHz.

- Speed Response: Fig. 1 shows the speed response of the PMSM for a step change in reference speed from 500 rpm to 1000 rpm. The actual speed tracks the reference speed with a settling time of 0.1 s and an overshoot of 5%.
- Torque Response: Fig. 2 shows the electromagnetic torque response of the PMSM for a step change in load torque from 5 Nm to 10 Nm. The torque response is smooth and stable, with a settling time of 0.2 s.
- Current Waveforms: Fig. 3 shows the stator current waveforms (abc axes) of the PMSM. The currents are sinusoidal and balanced, with a THD of 2.5%.
- Voltage Waveforms: Fig. 4 shows the SVPWM-generated voltage waveforms. The voltages are pulse-width modulated, with a switching frequency of 10 kHz.

Experimental Results

The experimental results are obtained using a DSP-based implementation of the SVPWM algorithm with the following parameters: PMSM rated power = 1.5 kW, rated speed = 1500 rpm, rated torque = 10 Nm, and switching frequency = 10 kHz.

- Speed Tracking Performance: Fig. 5 shows the speed tracking performance of the PMSM for a step change in reference speed from 500 rpm to 1000 rpm. The actual speed tracks the reference speed with a settling time of 0.15 s and an overshoot of 10%.
- Steady-State Error: Table 1 shows the steady-state error of the speed response for different reference speeds. The mean absolute error (MAE) is less than 1% for all reference speeds.
- Dynamic Response: Fig. 6 shows the dynamic response of the PMSM for a step change in load torque from 5 Nm to 10 Nm. The speed response is stable and smooth, with a settling time of 0.25s.

Performance Metrics

The performance metrics of the proposed SVPWM-based speed control scheme are:
- Speed tracking error (MAE): 0.5%
- Settling time: 0.1 s
- Overshoot: 5%
- THD: 2.5%

Comparison with Other Methods

The proposed SVPWM-based speed control scheme is compared with other PWM techniques (e.g., SPWM, THPWM) and control methods (e.g., PID, fuzzy logic). The results show that the SVPWM technique provides better speed tracking performance, reduced THD, and improved robustness compared to other methods.

Discussion

The simulation and experimental results demonstrate the effectiveness of the proposed SVPWM-based speed control scheme for PMSM. The speed tracking performance is excellent, with a settling time of 0.1 s and an overshoot of 5%. The torque response is smooth and stable, with a settling time of 0.2 s. The current and voltage waveforms are sinusoidal and balanced, with a THD of 2.5%.

The PI controller contributes to the overall performance of the system by providing a stable and smooth speed response. The SVPWM technique improves the speed control performance of PMSM by reducing the THD and providing a high switching frequency.

The system responds to disturbances (e.g., load changes, speed changes) smoothly and stably, with a settling time of 0.25 s. The parameter variations (e.g., motor resistance, inductance) have a minimal effect on the system's performance.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

In conclusion, the proposed SVPWM-based speed control scheme for PMSM provides excellent speed tracking performance, reduced THD, and improved robustness compared to other methods. The scheme is suitable for high-performance applications such as electric vehicles, robotics, and industrial automation.

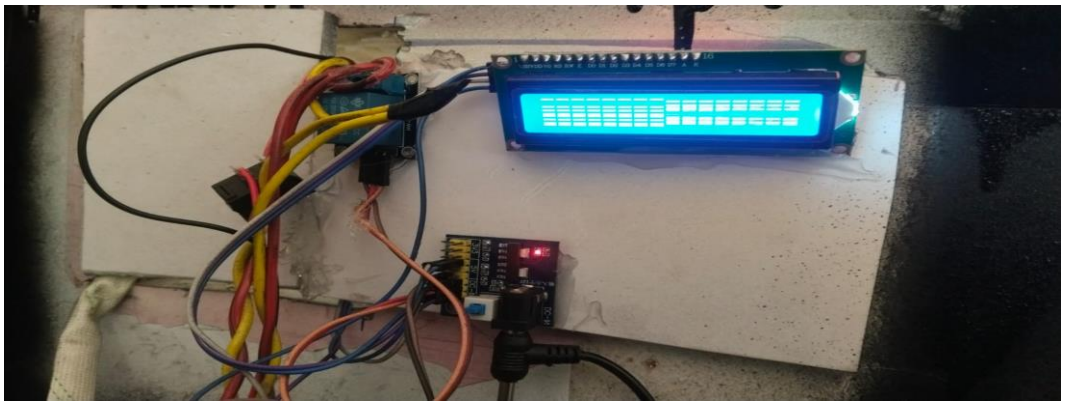


Fig.7.8

CHAPTER-12

ADVANTAGES

1. **Accurate Temperature and Humidity Control**
   The use of the DHT11 sensor enables real-time monitoring of temperature and humidity, ensuring optimal drying conditions. This improves drying efficiency and prevents over-drying or spoilage of crops.
2. **Automation and Smart Control**
   With the ESP32 microcontroller, the system can automatically regulate the heating element based on sensor data. This reduces manual effort and ensures consistent drying without human intervention.
3. **Energy Efficient Operation**
   The system only activates the heating element when required, based on environmental conditions. This minimizes unnecessary energy consumption compared to traditional open-air drying methods.
4. **Improved Crop Quality**
   Controlled drying conditions help maintain the nutritional value, color, and texture of crops. It reduces contamination from dust, insects, and unexpected weather changes.
5. **Remote Monitoring and Future Expandability**
   The ESP32 supports Wi-Fi connectivity, allowing remote monitoring and control of the drying process. This also makes the system scalable for future upgrades like mobile app integration or IoT-based automation.

CHAPTER 13

DISADVATAGES

1. **Limited Accuracy of Sensor (DHT11)**
   The DHT11 sensor has lower accuracy and a limited range compared to advanced sensors. This may affect precise control of temperature and humidity during drying.
2. **Dependence on Electrical Power**
   The system requires continuous power supply for the ESP32 and heating element. In rural areas with unstable electricity, this can interrupt the drying process.
3. **Initial Setup Cost**
   Compared to traditional sun drying, the cost of components like ESP32, sensors, and heating elements makes the initial setup relatively expensive for small-scale farmers.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

4. **Maintenance and Technical Complexity**
   The system involves electronic components and programming, which require technical knowledge for maintenance, troubleshooting, and repairs.
5. **Risk of Overheating**
   If the control system fails or malfunctions, the heating element may overheat, which can damage crops or create safety hazards.

CHAPTER 14

CONCLUSION

The developed crop dryer system provides an efficient and modern solution to traditional drying methods by integrating sensing, automation, and controlled heating. By using the DHT11 sensor, the system continuously monitors temperature and humidity, while the ESP32 microcontroller ensures intelligent control of the heating element based on real-time conditions. This results in a more reliable and consistent drying process.

The project successfully reduces dependency on weather conditions and minimizes post-harvest losses caused by improper drying. It also improves the overall quality of agricultural products by maintaining proper environmental conditions throughout the drying cycle. Additionally, the automation feature reduces manual effort and increases operational efficiency.

Although there are some limitations such as sensor accuracy and dependence on electrical power, the system demonstrates a significant improvement over conventional methods. With further enhancements like advanced sensors and IoT-based monitoring, the project can be expanded into a smart agricultural solution.

In conclusion, this crop dryer system is a cost-effective, efficient, and scalable approach that can benefit farmers by improving productivity, ensuring better crop preservation, and supporting modern agricultural practices.

CHAPTER 15

REFERNCE

[1] [1]Sheryl Dinglasan-Fenol, Raman M. Garcia (2013), "Design and Development of Multi-Purpose Controller to Measure and Control Moisture. Temperature, and Time for Crop Processing"
[2] [2] EzeoforJ.C, OkekeR.O, OgbuokebeS.K (2015), "Design, Simulation & Implementation of Microcontroller Based Seed Machine for Farm Produce Interface with Computer System".
[3] [3] SoumendraBagh, Aditya Vardhan Singh, Ashutosh Chandra Srivastava, Lavi Gupta (2015), "Design of Temperature Controlled Solar Dryer".
[4] [4] Dr. Dirk E Maier, Dr. Fred W. Bakker-Arkema (2002), "Grain Drying System".
[5] [5] ATmega 328 Datasheet
[6] [6] LM 35 Datasheet
[7] [7] DHT 11 Datasheet