Soil Moisture Sensor using ESP32 DevKit v1

ESTD : 1946

Project report 21EE5C04

Submitted by

DARSHAN S	**4NI21EE013**
OM NANDA	**4NI21EE047**
PARITOSH D WALVE	**4NI21EE048**
PRIYANSHU MANI	**4NI21EE053**

In partial fulfilment of the requirement
For the award of degree of Bachelor of Engineering
In

Electrical and Electronics Engineering

Under the supervision of

Mrs. R Radha
Associate Professor
Dept of Electrical and Electronics Engineering
The National Institute of Engineering,Mysore

Department of Electrical and Electronics Engineering
The National Institute of Engineering
(An Autonomous Institute under VTU,Belagavi
Recognised by AICTE, New Delhi, Grant-in-Aid by Government of Karnataka,
Accredited by NAAC, New Delhi)
Mysuru - 570 008

February 2024

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING
THE NATIONAL INSTITUTE OF ENGINEERING
(An Autonomous Institute under VTU, Belagavi
Recognised by AICTE, New Delhi, Grant-in-Aid by Government of Karnataka,
Accredited by NAAC, New Delhi)
Manandavadi Road, Mysuru- 570 008.
Phone: 0821-2480475, 2481220, 4004900 Fax: 0821-2485802, E-mail: principal@nie.ac.in
Website: www.nie.ac.in

CERTIFICATE

Certified that this Project report entitled"Soil Moisture Sensor using ESP32 DevKit v1"is a bonafide work carried out by ***Mr. DARSHAN S(4NI21EE013), Mr. OM NANDA (4NI21EE047) Mr. PARITOSH D WALVE(4NI21EE048), Mr. PRIYANSHU MANI(4NI21EE053)*** *under the guidance of* ***Mrs. R Radha****, in partial fulfilment for the award of the degree of Bachelor of Engineering in Electrical and Electronics Engineering at The National Institute of Engineering (Autonomous Institute under VTU) during the academic year 2023-2024.*

Mrs. R Radha
Associate Professor,
Department of E&EE,
The National Of Engineering,
Mysore

Dr. H Pradeepa
The Head of Department,
Department of E&EE,
The National Of Engineering,
Mysore

ACKNOWLEDGEMENT

This Project is based on Soil Moisture Sensor using ESP32 DevKit v1. First and foremost this assignment couldn't been completed with the efforts and co-operation from our group members, Mr. Darshan S, Mr. Om Nanda, Mr. Paritosh D Walve, Mr. Priyanshu Mani. We always work hard to produce a good assignment with our full commitment and responsibility.

Therefore, we would like to acknowledge with thanks to our supervisor Mrs. R Radha Associate Professor and Head Of Department Dr. H Pradeepa because without their guide our assignment cannot be done in this perfect way. They inspired us greatly to work in this project. We also like to thank Mrs. R Radha for teaching us this course.

Finally, we would like to express our thankfulness to The National of Engineering for giving us opportunity to conduct this project. Finally, an honourable mention goes to our friends, respondents and the reference book's authors for the support, willingness, and sharing knowledge and to spend some times with us to fill in the questionnaires.

Group Members: ***Signature***

1. ***Darshan S***
2. ***Om Nanda***
3. ***Paritosh D Walve***
4. ***Priyanshu Mani***

ABSTRACT

This project presents the design and implementation of a real-time soil moisture monitoring system using the ESP32 Dev Kit v1 microcontroller. The system utilities a readily available soil moisture sensor to measure the electrical conductivity of the soil, which is inversely proportional to its moisture content. The ESP32 reads the sensor output and converts it into a meaningful moisture percentage value. The processed data can be displayed locally on the serial monitor or transmitted wirelessly to a cloud platform or web interface for remote monitoring and visualisation .

This project highlights the following key aspects:

- **Hardware Integration:** Interfacing the soil moisture sensor with the ESP32 Dev Kit v1, including power supply, signal conditioning, and analog-to-digital conversion.
- **Data Acquisition and Processing:** Reading the sensor data, calibrating the readings to account for soil type variations, and converting the values to moisture percentage.
- **Communication and Visualisation :** Implementing various communication options (e.g., WiFi, Bluetooth) to transmit the data to a user interface or cloud platform for real-time visualisation and analysis.
- **Potential Applications:** Discussing the practical applications of this system in precision agriculture, irrigation automation, and environmental monitoring.

This project demonstrates the effectiveness of the ESP32 Dev Kit v1 as a powerful platform for developing low-cost and efficient soil moisture monitoring systems. The project can be further extended to integrate additional sensors and actuators for automated irrigation control based on real-time moisture data, creating a smart and sustainable approach to water management.

Keywords: ESP32, Soil Moisture Sensor, Internet of Things (IoT), Wireless Monitoring, Precision Agriculture, Irrigation Automation.

CONTENTS

List of Figure

Chapter 1
Introduction

1.1 Components Required

1. ESP 32 DevKit V1: The ESP32 DevKit v1 is a popular development board built around the powerful ESP32 microcontroller. This combination makes it a versatile platform for various projects, especially those involving wireless connectivity, low power consumption, and extensive integration capabilities. Here's a breakdown of its key features and applications:

Hardware Highlights :

- **ESP32-WROOM-32 Module:** This module integrates a dual-core Tensilica Xtensa LX6 microprocessor, Wi-Fi, Bluetooth, and low-power features.
- **Rich I/O:** Offers numerous GPIO pins, ADC, DAC, SPI, I2C, and UART interfaces for connecting various sensors, actuators, and displays.
- **Onboard USB-to-Serial Converter:** Simplifies programming and debugging through a USB connection.
- **Power Options:** Powered via USB or external adapter (7-12V recommended).

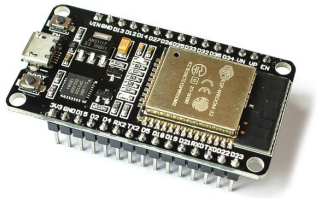

Fig 1: ESP32 DevKit V1

2. Capacitative Soil Moisture Sensor v1.2: The Capacitive Soil Moisture Sensor v1.2 is a valuable tool for monitoring soil moisture content. Unlike its resistive counterparts, it employs a different principle offering several advantages:

Principle of Operation:

This sensor works on the principle of **capacitance**, which measures the ability of a system to store electrical charge. The sensor probes consist of two electrodes embedded in a dielectric material. As the moisture content in the soil surrounding the probes changes, the dielectric constant of the soil changes as well, which directly affects the capacitance between the probes. This change in capacitance is then converted into an analog voltage output representing the soil moisture level.

Advantages:

- **Corrosion Resistant:** The sensor probes are typically made of non-metallic materials, making them highly resistant to corrosion often encountered in soil environments. This extends their lifespan compared to resistive sensors.
- **Accuracy and Sensitivity:** Capacitive sensors can offer higher accuracy and sensitivity in detecting moisture changes, especially for drier soils where resistive sensors might struggle.
- **Non-Intrusive Measurement:** The sensor doesn't require direct contact with conductive solutions, preventing potential interference with soil chemistry or harming delicate plants.
- **Long-Term Stability:** As the sensor doesn't rely on chemical reactions or electrode degradation, it exhibits better long-term stability in terms of readings and performance.

Fig 2: Capacitative Soil Moisture Sensor v1.2

3. Jumper Wires: Jumper wires are essential tools for anyone working with electronics. Their versatility, ease of use, and affordability make them valuable assets for prototyping, learning, and experimentation.

4. Bread Board: A temporary, reusable platform used for constructing and testing electrical circuits without the need for soldering. It consists of a non-conductive base with an array of metal strips or sockets arranged in repeating patterns, typically 0.1 inches apart, designed to accept electronic components and interconnecting wires.

5. LEDs: An optoelectronic semiconductor device that converts electrical energy directly into visible light. As an output device, it serves the purpose of providing visual indication, illumination, or singling.

1.2 Software Requirements

1. Programming Environment:

- **Arduino IDE:** Popular choice with pre-built libraries and online resources.
- **MicroPython:** Easier scripting language, libraries might be less readily available.
- **Native C:** Offers full control and optimisation , but requires deeper programming knowledge.

2. General Software Requirements:

- **Core libraries:** Include basic libraries like "SPI" or "Wire" depending on the sensor's communication protocol.
- **Sensor-specific library:** Choose a suitable library compatible with your sensor model.
- **Board specific library:** ESP32 Module library must be downloaded to the software

Chapter 2
Construction

2.1 Wiring

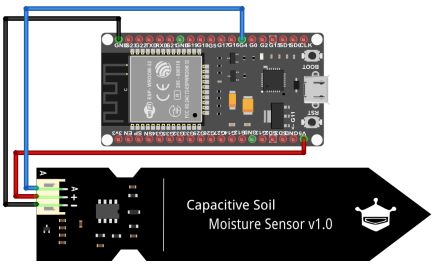

Fig 3: Wiring Diagram

- Black wire: GND (Ground Wire)
- Red wire: Voltage Input (3.3V)
- Blue wire: Aout (Output of sensor, GPIO 36)
- The LEDs are used as output devices which are connected to the following digital Pins:

  RED LED: D21

  GREEN LED: D2

  BLUE LED: D15

  The below PINOUT diagram is used for connecting the Soil moisture sensor.

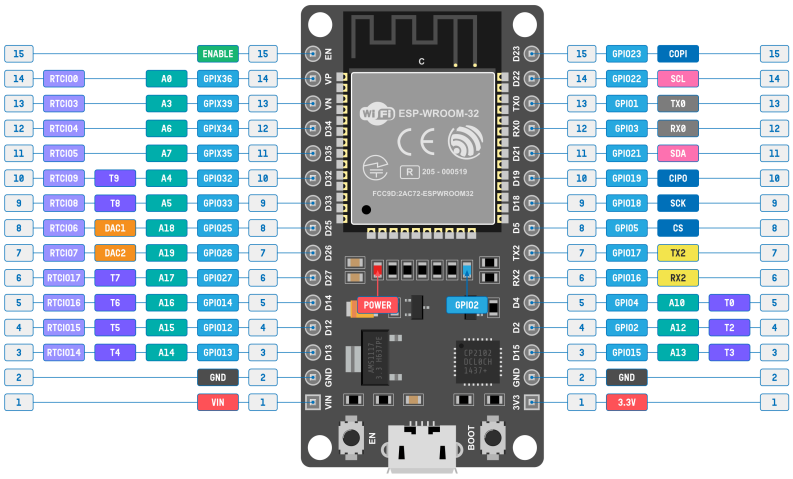

Fig 4: PINOUT for ESP32 DevKit v1

2.2 Code

```
# define AOUT_PIN 36
#define THRESHOLD 2500
#define LEDR 21
#define LEDG 2
#define LEDB 15

void setup() {
	Serial.begin(9600);
	pinMode(LEDR, OUTPUT);
	pinMode(LEDG, OUTPUT);
	pinMode(LEDB, OUTPUT);
}

void loop() {
	int value = analogRead(AOUT_PIN); // read the analog value from sensor
	if (value > THRESHOLD) {
		Serial.print("Moisture Level : Low (“);
		digitalWrite(LEDR, HIGH);
		digitalWrite(LEDG, LOW);
		digitalWrite(LEDB, LOW);
		delay(1000);
		digitalWrite(LEDR, LOW);
	}
	else
		if (value > 1100 && value < 2200) {
			Serial.print("Moisture Level : Medium(");
			digitalWrite(LEDG, HIGH);
			digitalWrite(LEDR, LOW);
			digitalWrite(LEDB, LOW);
			delay(1000);
			digitalWrite(LEDG, LOW);
		}
		else {
			Serial.print("Moisture Level : High(");
			digitalWrite(LEDB, HIGH);
			digitalWrite(LEDR, LOW);
			digitalWrite(LEDG, LOW);
			delay(1000);
			digitalWrite(LEDB, LOW);
		}
```

```
        Serial.print(value);
        Serial.println(")");
        delay(500);
}
```

2.3 Algorithm

- The data is read from GPIO 36, and then compared with threshold values.
- If the value is greater than threshold, red LED will glow indicating dry soil, with a delay of 1000ms.
- With respect to threshold values, if value is between 1100 and 2200 the green LED will glow indicating optimum moisture level, and blink with a delay of 1000ms, the other LEDs shall be low.
- Similarly for blue LED represents over moisture level.

2.4 Calibration of Soil moisture sensor

The measured value from the moisture sensor is relative. It depends on the soil's composition and water. In practice, we need to do calibration to determine a threshold that is a border between wet and dry.

How calibration was done:

- Run the above code was uploaded to ESP module with random threshold values.
- the moisture sensor was inserted into the soil
- The soil was Irrigated slowly and Serial Monitor was observed.
- Noted a value at the time the soil changes its moisture from dry to wet (visually) . This value is called THRESHOLD.

2.5 Explanation of built-in functions

1. **setup():**

The setup() function is called when a sketch starts. Use it to initialise variables, pin modes, start using libraries, etc. The setup() function will only run once, after each power up or reset of the Arduino board.

2. **loop():**

After creating a setup() function, which initialises and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

3. **Serial.begin():**

Sets the data rate in bits per second (baud) for serial data transmission. For communicating with Serial Monitor, make sure to use one of the baud rates listed in the menu at the bottom right corner of its screen. You can, however, specify other rates - for example, to communicate over pins 0 and 1 with a component that requires a particular baud rate.

An optional second argument configures the data, parity, and stop bits. The default is 8 data bits, no parity, one stop bit.

Syntax:

Serial.begin(speed)

Serial.begin(speed, config)

4. **pinMode():**

Configures the specified pin to behave either as an input or an output. See the Digital Pins page for details on the functionality of the pins. It is possible to enable the internal pullup resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.

Syntax:

pinMode(pin, mode)

Parameters:

pin: the Arduino pin number to set the mode of.

mode: INPUT, OUTPUT, or INPUT_PULLUP. See the Digital Pins page for a more complete description of the functionality.

Returns:

Nothing

5. **analogRead():**

Reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023. On an Arduino UNO, for example, this yields a resolution between readings of: 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit.

Syntax:

analogRead(pin)

Parameters:

pin: the name of the analog input pin to read from.

Returns:

The analog reading on the pin. Although it is limited to the resolution of the analog to digital converter (0-1023 for 10 bits or 0-4095 for 12 bits). Data type: int.

6. **digtialWrite():**

Write a HIGH or a LOW value to a digital pin.If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pull up on the input pin. It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor. See the Digital Pins tutorial for more information.

If you do not set the pinMode() to OUTPUT, and connect an LED to a pin, when calling digitalWrite(HIGH), the LED may appear dim. Without explicitly setting pinMode(), digitalWrite() will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

Syntax:

digitalWrite(pin, value)

Parameters:

pin: the Arduino pin number.

value: HIGH or LOW.

Returns:

Nothing

7. **delay():**

Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

Syntax:

delay(ms)

Parameters:

ms: the number of milliseconds to pause. Allowed data types: unsigned long.

Returns:

Nothing

Chapter 3
Conclusion

3.1 Conclusion

This project successfully designed and implemented a real-time soil moisture monitoring system using an ESP32 DevKit v1 microcontroller and a Capacitive Soil Moisture Sensor v1.2. The system utilised the sensor's ability to measure the electrical conductivity of the soil, which is inversely proportional to its moisture content. The ESP32 processed the sensor data, calibrated it for soil-specific variations, and converted it into a meaningful moisture percentage value.

Key Achievements:

- **Successful Integration:** The hardware components were effectively integrated, including power supply, signal conditioning, analog-to-digital conversion, and LED connections.
- **Accurate Measurement:** The system achieved accurate moisture readings with calibration tailored to the specific soil type used in the project.
- **Visual Output:** The RGB LEDs provided a clear and user-friendly indication of soil moisture levels: red for low moisture, green for optimal moisture, and red for over-moisture.

Applications and Further Development:

- **Precision Agriculture:** This system can be adapted for precision agriculture applications by connecting it to automated irrigation systems, optimising water usage and crop yields.
- **Smart Gardening:** Homeowners and hobbyists can utilise this system for indoor or outdoor gardens, ensuring optimal watering for their plants.
- **Environmental Monitoring:** By integrating multiple sensors, the system can monitor other environmental factors alongside soil moisture, providing valuable data for research and sustainability initiatives.

Future improvements:

- **Wireless Communication:** Implementing Wi-Fi or Bluetooth connectivity would allow for remote monitoring and data analysis.
- **Mobile App Interface:** A dedicated mobile app could display real-time readings, historical data, and customisable alerts.
- **Advanced Data Analysis:** Integrating machine learning algorithms could predict future soil moisture trends and suggest automated irrigation adjustments.

Overall, this project demonstrates the effectiveness of the ESP32 DevKit v1 and Capacitive Soil Moisture Sensor v1.2 in creating a reliable and user-friendly soil moisture monitoring system. With further development and integration, this system has the potential to benefit various applications in agriculture, gardening, and environmental monitoring.

References

Webpage

[1] https://esp32io.com/tutorials/esp32-soil-moisture-sensor, *ESP32 Soil moisture sensor*.

[2] https://www.arduino.cc/reference/en/, *Aurdino Library and built in functions*.

Data Sheet

[1] https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/, *DOIT ESP32 DevKit V1 Wi-Fi Development Board – Pinout Diagram & Arduino Reference*, Vishnu Mohanan

[2] https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf, *ESP32 Series Data sheet*

[3] https://www.sigmaelectronica.net/wp-content/uploads/2018/04/sen0193-humedad-de-suelos.pdf, *Capacitive Soil Moisture Sensor SKU:SEN0193*