AGRIVYAAN

Index

INTRODUCTION

- **1. Project Title**

  Smart Agriculture System

- **2. Problem Statement**

  Traditional irrigation methods result in inefficient water use and require constant human intervention. This project addresses the need for automated, efficient, and precise irrigation to save water and improve crop health.

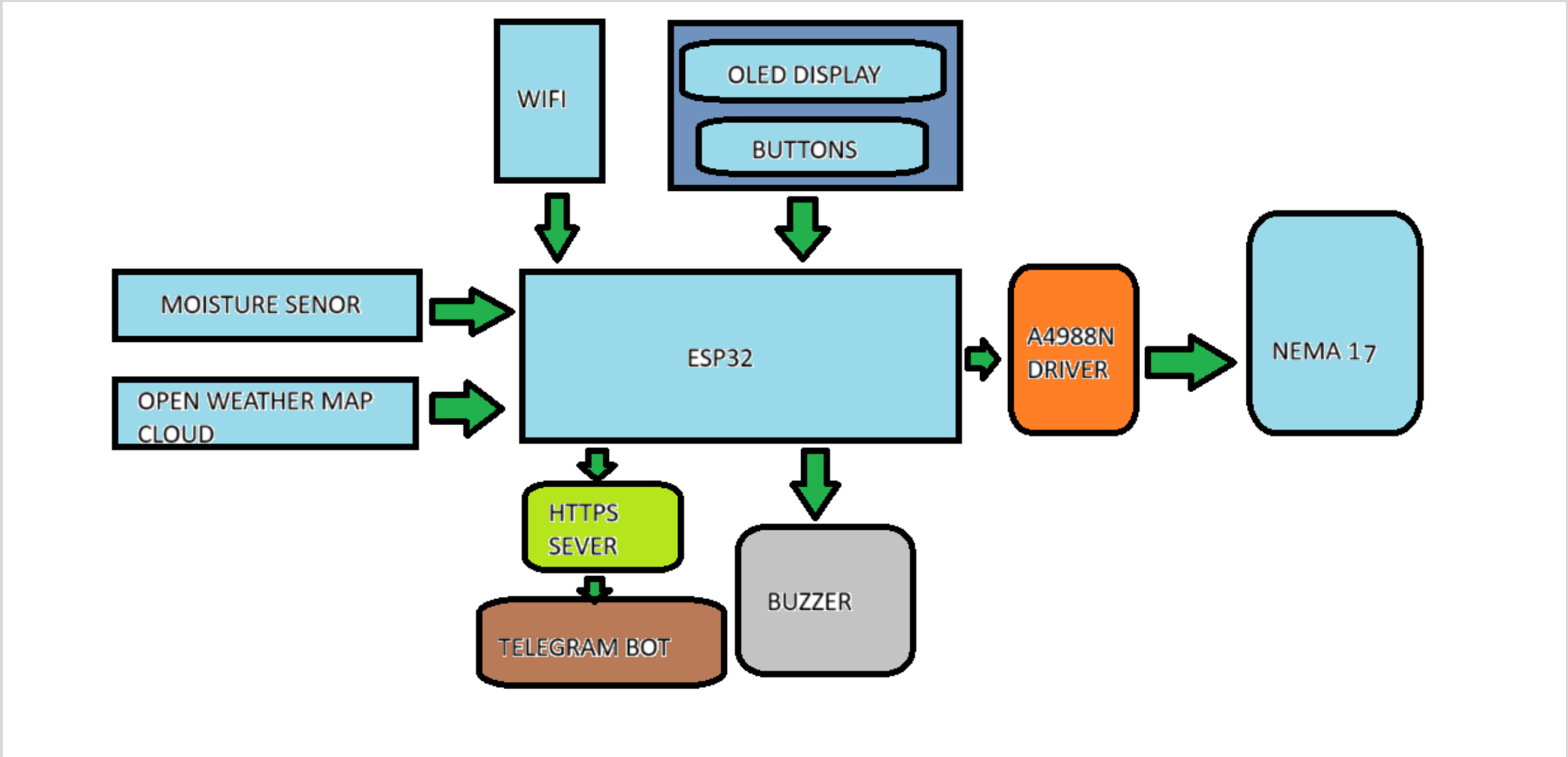
- **3. Objective**

  To develop a smart system for monitoring soil moisture, weather conditions, and pest control, automating irrigation processes and ensuring effective water management.

PROJECT OVERVIEW

- **1. Project Description**
- The system uses an ESP32 microcontroller, OLED display, moisture sensors, and a stepper motor to monitor soil moisture, display weather conditions, automate irrigation, and control pests using a buzzer alarm. It features Telegram integration for remote data monitoring.
- **2. Block Diagram**

- **Input:** Moisture sensors, temperature sensor, buttons.
- **Processing:** ESP32 handles inputs and controls devices.
- **Output:** OLED display, stepper motor, buzzer, and pest alarm.
  Diagram to be included in the presentation.

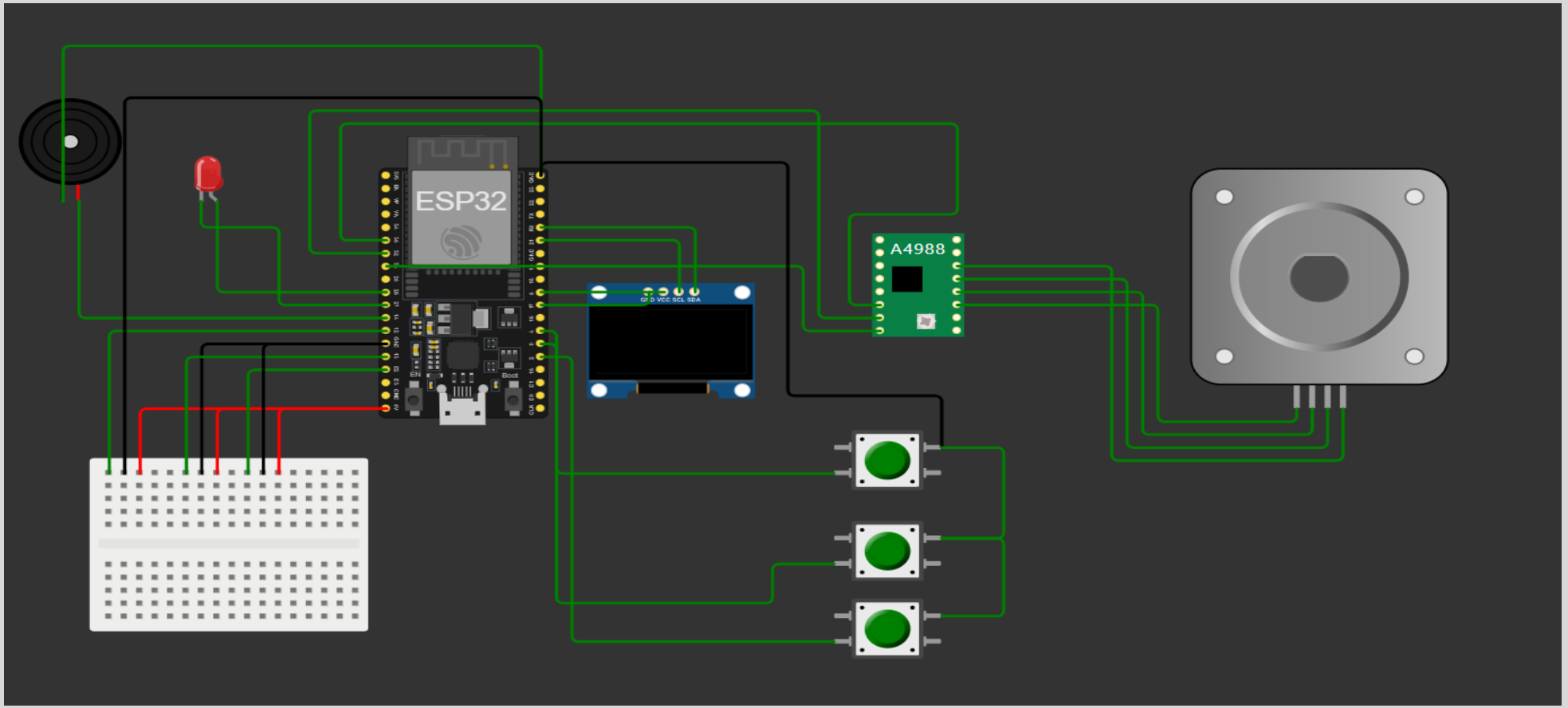

WIFI
OLED DISPLAY
BUTTONS
MOISTURE SENOR
OPEN WEATHER MAP CLOUD
ESP32
A4988N DRIVER
NEMA 17
HTTPS SEVER
TELEGRAM BOT
BUZZER

HARDWARE COMPONENTS

Component	Specification
ESP32	Microcontroller
OLED Display	128x32 pixels
Moisture Sensors	Analog (3 units)
Stepper Motor (NEMA17)	1.8° step angle
a4988n Driver	Stepper motor control
Buzzer	General-purpose

HARDWARE COMPONENTS

Circuit diagram



HARDWARE COMPONENTS

Hardware Description

- **ESP32:** Processes sensor data and controls devices.
- **OLED Display:** Visual interface for data and menu.
- **Moisture Sensors:** Monitor soil moisture levels.
- **Stepper Motor & DRV8825:** Position irrigation system.
- **Buzzer:** Feedback for actions and pest control.

SOFTWARE COMPONENTS

- **1. Programming Language**
- Arduino C++
- **2. openweather api**
- For monitor the weather condition and rain status with clous informations
- **3. woki**
- Software circuit designing
- **4. telegram**
- For monitoring the moisture,weather,ecomode using http protocol

SOFTWARE COMPONENTS

- **2. Software Description**

- Sensor data is read and processed by ESP32.
- Menu navigation is implemented for controlling features like Eco Mode, Weather display, and Pest Alarm.
- Telegram bot integration enables remote monitoring.

- **3. Algorithm**

1. Initialize hardware and connect to Wi-Fi.
2. Read sensor data periodically.
3. Trigger motor for irrigation if moisture is below threshold.
4. Display weather data on OLED.
5. Allow user interaction via buttons.
6. Provide remote data via Telegram API.
7. Finding the rotational status by esp32

PROJECT IMPLEMENTATION

- **1. Implementation Steps**

1. Set up hardware connections and initialize components.
2. Implement Wi-Fi and Telegram connectivity.
3. Develop menu navigation and control logic.
4. Test soil moisture-based motor movement.
5. Integrate all modules and test system functionality.

- **2. Challenges Faced**

- Wi-Fi connection issues resolved by optimizing code.
- Stepper motor calibration for precise movement.
- Issues of motor driver

- **3. Testing and Debugging**

- Conducted functional testing for each module.
- Debugged sensor readings and motor control logic.

RESULTS AND DISCUSSION

- **1. Results**
- Automatic irrigation based on soil moisture.
- Real-time weather and soil data displayed on OLED.
- Remote monitoring using Telegram.
- **2. Discussion**
- Efficient water management achieved.
- Reliable performance with some limitations in rain detection accuracy.

3. Comparison with Existing Solutions

Aspect	Existing Solutions	Proposed System
Automation	Partial	Fully automated
Remote Monitoring	Limited	Telegram Integration
Cost-Effectiveness	Moderate	Affordable

CONCLUSION

- **1. Summary**
- The project focuses on creating a **Smart Agriculture System** designed to optimize farming operations through automation and monitoring. Key components include an ESP32 microcontroller, sensors (moisture, temperature), an OLED display, buttons for navigation, a NEMA17 stepper motor, a DRV8825 motor driver, and a water pump .

1. **Moisture Monitoring and Watering**: Measures soil moisture levels, triggers alarms for dry conditions, and automatically waters the soil.
2. **Weather Display**: Shows temperature and weather conditions, including a rain check.
3. **Eco Mode**: Automates motor movements for efficient watering based on a user-set timer.
4. **Pest Control**: Activates a buzzer to repel pests and animals.
5. **User Interaction**: Menu navigation via buttons, feedback through buzzers, and data visualization on an OLED and all this user can monitor through telgram

CONCLUSION

- **future scope**
- **Extend Telegram Bot Functionality**: Add more commands to control and monitor other aspects of the system like adjusting irrigation levels, scheduling automated tasks, or integrating additional sensors.
- **Integrate with Cloud**: Send sensor data and alerts to a cloud service like Firebase or ThingSpeak for historical tracking and analysis.
- **Mobile App for Remote Control**: Develop a dedicated mobile app for easier management of the system, with custom notifications and settings.
- **Acknowledgments**
- **Mentors**: For their guidance and advice throughout the project.
- **Peers**: For collaboration, testing, or feedback.
- **Libraries and Platforms**: Acknowledge contributions like the Adafruit libraries, Telegram API, and Wi-Fi connectivity setups.

REFERENCES

- Libraries
  - <Adafruit_GFX.h>
  - <Adafruit_SSD1306.h>
  - <WiFi.h>
  - <HTTPClient.h>
  - <ArduinoJson.h>
  - <UniversalTelegramBot.h>
  - <WiFiClientSecure.h
- OpenWeather API
- Arduino and ESP32 documentation
- Telegram documentation
- Woki platform(oled meniu designing)
- Tinkercad

APPENDICES

- **Pin Definitions:**

1. **BUTTON_UP**: Pin 15
2. **BUTTON_SELECT**: Pin 4
3. **BUTTON_BACK**: Pin 5
4. **BUZZER**: Pin 23
5. **LED_PIN**: Pin 13
6. **DIR_PIN**: Pin 27 (Stepper motor direction pin)
7. **STEP_PIN**: Pin 26 (Stepper motor step pin)
8. **ENABLE_PIN**: Pin 25 (Stepper motor enable pin)
9. **PEST_ALARM_PIN**: Pin 12 (for pest alarm buzzer)

- **Moisture Sensor Pins:**

1. **MOISTURE_SENSOR_1**: Pin 34
2. **MOISTURE_SENSOR_2**: Pin 35
3. **MOISTURE_SENSOR_3**: Pin 32

- **Display:**

- **OLED Display**: I2C communication (SDA, SCL)
    - **SCL**: Pin 21
    - **SDA**: Pin 21
    - **Address**: 0x3C

APPENDICES

- **Network Configuration:**
- **SSID**: "offline"
- **Password**: "raziq2025"
- **City**: "Thrissur" (for weather fetching)
- **API Key**: "9a73e2999f2ac696e9e8ddb256c1ab4f"
- **Bot Token**: "7516663328:AAE0BMRYvVAu-AdDYzEthbOLJ11rhG1zhsM"
- **Chat ID**: "5310005938"
- **System Configuration:**
- **STEP_PER_REV**: 200 (Steps per revolution for the stepper motor)
- **MICROSTEPS**: 16 (Microsteps for stepper motor)
- **LEAD_SCREW_PITCH**: 2.0 (Pitch of the lead screw for stepper motor)
- **TOTAL_LENGTH_CM**: 21.0 (Total length to be moved by the motor)
- **PARTITION_COUNT**: 3 (Number of partitions)
- **MOISTURE_THRESHOLD**: 30 (Threshold for moisture level triggering action)
- **Functions and Actions:**
- **Moisture Sensors**: Monitor soil moisture levels and trigger action if below the threshold (e.g., water spraying).
- **Eco Mode**: Execute eco-friendly cycles, moving the stepper motor to different partitions repeatedly for the set number of cycles.
- **Weather Display**: Fetch weather data (temperature, conditions) from OpenWeather API.
- **Pest Alarm**: Control a buzzer for pest detection or as a deterrent.
- **Telegram Bot**: Allow remote monitoring and control via Telegram commands.

- To calculate the rotation of the **NEMA 17** stepper motor using an **A4988** stepper driver, you'll need to know a few key parameters and apply some simple formulas.
- **Parameters to Consider:**

1. **Step Angle of the Motor**: This is typically **1.8° per step** for a NEMA 17 motor, but you should check the datasheet of your specific motor for the exact value. This means it takes 200 full steps to complete one full revolution (360°).
2. **Microstepping**: The A4988 driver can be set to various microstep resolutions. The most common ones are:
    1. **Full step** (1 step per motor step)
    2. **Half step** (2 microsteps per motor step)
    3. **Quarter step** (4 microsteps per motor step)
    4. **Eighth step** (8 microsteps per motor step)
    5. **Sixteenth step** (16 microsteps per motor step)
3. **Lead Screw Pitch**: For the **lead screw**, the distance moved per step is dependent on the pitch of the lead screw and the microstepping setting. You have **2.0 mm per lead screw rotation** as the pitch in your code.

- **Steps to Calculate Rotation:**

1. **Determine the Number of Steps for One Full Revolution**: The NEMA 17 motor has a **1.8° step angle** (for a full step), meaning it takes **200 steps** to complete one full revolution.
2. For microstepping:
    1. **Full step**: 200 steps per revolution.
    2. **Half step**: 400 steps per revolution.
    3. **Quarter step**: 800 steps per revolution.
    4. **Eighth step**: 1600 steps per revolution.
    5. **Sixteenth step**: 3200 steps per revolution.
3. **Calculate the Distance Moved per Step**: The lead screw has a pitch of **2.0 mm**. This means the nut on the lead screw moves **2 mm** for every full rotation (360°) of the lead screw.
4. The distance moved per step (for full step mode) is:
5. Distance per step=Lead screw pitchSteps per revolution\text{Distance per step} = \frac{\text{Lead screw pitch}}{\text{Steps per revolution}}Distance per step=Steps per revolutionLead screw pitch Distance per step=2.0 mm200=0.01 mm per step (for full step mode)\text{Distance per step} = \frac{2.0 \, \text{mm}}{200} = 0.01 \, \text{mm per step (for full step mode)}}Distance per step=2002.0mm=0.01mm per step (for full step mode)For microstepping, the distance moved per step will be even smaller. For example:
    1. **Half step**: 0.01 mm / 2 = **0.005 mm per step**
    2. **Quarter step**: 0.01 mm / 4 = **0.0025 mm per step**
    3. **Eighth step**: 0.01 mm / 8 = **0.00125 mm per step**
    4. **Sixteenth step**: 0.01 mm / 16 = **0.000625 mm per step**

•Distance per step=Steps per revolutionLead screw pitch
Distance per step=2.0 mm200=0.01 mm per step (for full step mode)\text{Distance per step} = \frac{2.0 \, \text{mm}}{200} = 0.01 \, \text{mm per step (for full step mode)}Distance per step=2002.0mm=0.01mm per step (for full step mode)
For microstepping, the distance moved per step will be even smaller. For example:
•**Half step**: 0.01 mm / 2 = **0.005 mm per step**
•**Quarter step**: 0.01 mm / 4 = **0.0025 mm per step**
•**Eighth step**: 0.01 mm / 8 = **0.00125 mm per step**
•**Sixteenth step**: 0.01 mm / 16 = **0.000625 mm per step**
•**Calculate the Number of Steps to Move a Given Distance**: Suppose you want to move the stepper motor by a certain distance in millimeters, the number of steps required can be calculated by:
Number of steps=Distance to moveDistance per step\text{Number of steps} = \frac{\text{Distance to move}}{\text{Distance per step}}Number of steps=Distance per stepDistance to move
For example, to move **10 mm**:
•**Full step mode**: Number of steps=10 mm0.01 mm per step=1000 steps\text{Number of steps} = \frac{10 \, \text{mm}}{0.01 \, \text{mm per step}} = 1000 \, \text{steps}Number of steps=0.01mm per step10mm=1000steps
•**Half step mode**: Number of steps=10 mm0.005 mm per step=2000 steps\text{Number of steps} = \frac{10 \, \text{mm}}{0.005 \, \text{mm per step}} = 2000 \, \text{steps}Number of steps=0.005mm per step10mm=2000steps
•**Quarter step mode**: Number of steps=10 mm0.0025 mm per step=4000 steps\text{Number of steps} = \frac{10 \, \text{mm}}{0.0025 \, \text{mm per step}} = 4000 \, \text{steps}Number of steps=0.0025mm per step10mm=4000steps

- Motor

THANK YOU