Greenhouse CO2 Controller Project Report

Group 05

Olga Sharma
Upeksha Samarawickrama Liyanage
Lihini Hewage
Shamila Thennakoon

Third-year IOT Project
School of ICT
Metropolia University of Applied Sciences
12 October 2025

Content

1. Introduction

The Greenhouse CO_2 Controller project focused on designing an automated system with the capacity to regulate desirable levels of carbon dioxide in a greenhouse environment. The system is designed to help the plants to grow under the most optimal conditions by keeping CO_2 level in safe and productive range.The controller is programmed to automatically change both the ventilation fan and CO_2 injection valve to keep the CO_2 level within user-defined limits. Based on a Raspberry Pi Pico W microcontroller, the system reads the environmental data from attached sensors, stores configuration data in EEPROM, and communicates with the cloud through a RESTful API for real time monitoring and remote control.

1.1 Goals of the project

The primary objectives of this project are:

- Hold a user-specified target level of CO_2 concentration (max. 1500 ppm).
- Increase ventilation if the CO_2 level exceeds the safety threshold (2000 ppm).
- Store configuration data like CO_2 setpoint and network credentials in EEPROM.
- Ensure both local control via a user interface as well as remote monitoring through the ThingSpeak cloud platform.
- Implement safely, efficiently, and reliably both for miniature test systems and actual greenhouse conditions.

This document presents the theoretical basis, design procedure, hardware and software implementation, and testing of the Greenhouse CO_2 Controller system.

2. Theoretical Background

In a greenhouse environment, carbon dioxide (CO_2) supports photosynthesis and plant growth under greenhouse conditions[2]. The CO_2 concentration in the air is about 400 ppm under normal environmental conditions, but in most cases fails to achieve the maximum possible plant yield. CO2 concentrations in greenhouses of up to 1000–1200 ppm have been found to greatly enhance plant growth when temperature, humidity, and light intensity are maintained as constant factors [1].

A CO_2 controller maintains concentration within an optimal level. If CO_2 level drops too low, plant growth slows down and if levels exceed 2000 ppm, it can negatively affect both plants and human operators.Thus, it is important to keep CO_2 level in equilibrium with automation in greenhouses[3].

The system uses a closed-loop control process in which the sensor information is continuously compared with the target CO_2 level. The controller adjusts the speed of the fan and operates the CO_2 injection valve (in on/off manner) to maintain the CO_2

concentration in the green house ensuring safe and efficient operation while managing communication with the user interface, network and EEPROM as required based on the feedback[4].The project integrates both Modbus and I²C communication protocols for controlling the devices and for reading sensor values[5]. The sensor data is transmitted to the ThingSpeak cloud using an HTTP-based RESTful API for real time monitoring and remote command execution[6].

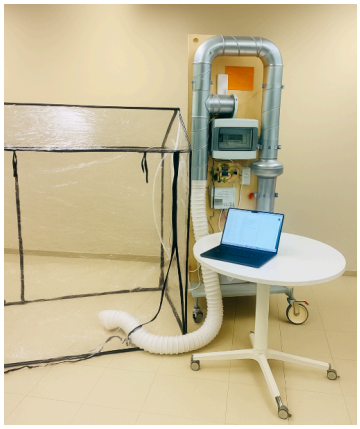
Lastly, the system uses EEPROM based non volatile memory to remember settings during power cycling, and FreeRTOS as the operating system for supporting multitask operations in order to provide efficient and reliable performance[7].

3. Methods and Materials

3.1 Hardware Requirements

The Greenhouse CO_2 Controller hardware is built around the Raspberry Pi Pico W microcontroller, which serves as the central control unit managing sensor readings, actuator control, user interaction, and cloud communication. The system integrates both Modbus RTU and I²C communication interfaces to interface with a variety of environmental sensors and control modules.[7]

- **Main setup**

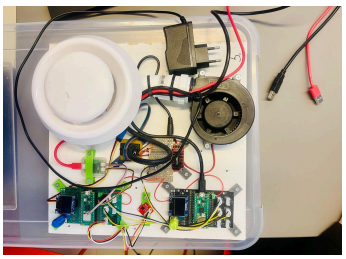


The hardware configuration consists of sensors, a ventilation control module, a CO_2 injection valve, an EEPROM and an OLED display.

The Pico W communicates with Modbus devices (fan controller, CO_2, and RH/T sensors) via the RS-485 interface, while the I²C bus is used for the EEPROM, and OLED display

Figure 1: Greenhouse CO2 controller setup

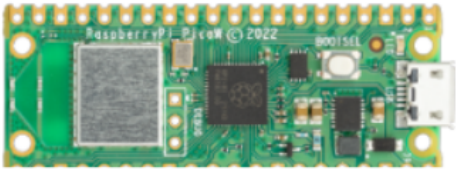
- **Miniature test setup**



We use a small test setup with a second microcontroller that simulates the real sensors, fan controller and valve. This lets us safely test the system's logic and communication before deploying it in a real greenhouse.

Figure 2: Miniature test setup

- **Raspberry Pi Pico W**



Main microcontroller for task coordination, networking, and logic execution[8].

Figure 3: The Raspberry Pi Pico W Rev3 board.

- **Produal MIO 12-V Module**

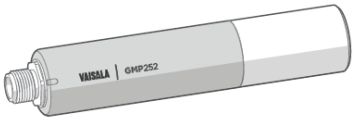
Ventilation control is achieved through the Produal MIO 12-V module, a Modbus-controlled I/O device that regulates fan speed based on CO_2 concentration levels.

- Output : 0-10 V analog signal (0V = 0% (off) ,10V=100% (maximum speed))
- Input **:** Digital pulse counter for fan speed feedback

Safety Response: If CO_2 concentration exceeds 2000 ppm, the controller automatically increases ventilation speed to maximum to rapidly reduce CO_2 level returning to normal control mode.[9]

Figure 4: Produal MIO 12-V

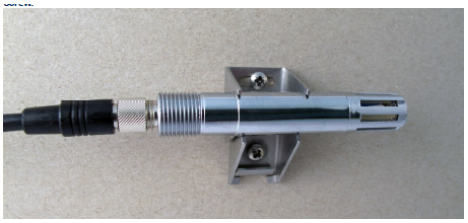
- **Vaisala GMP252 CO_2 Sensor**



Measures CO_2 concentration via Modbus RTU (address 240) [5]

Figure 5: GMP252 CO_2 Sensor

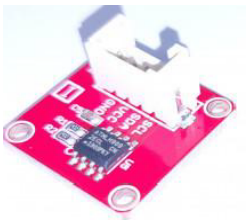
- **Vaisala HMP60 Sensor**



Measures temperature and relative humidity (address 241).[10]

Figure 6: HMP60 Relative Humidity and Temperature Sensor

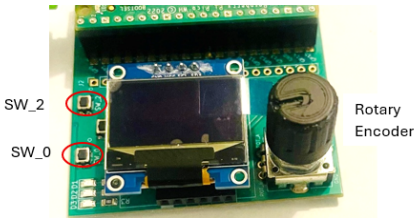
- **EEPROM (I²C)**



Writes, stores and reads back system configuration data and user defined parameters that must be held between power cycles such as CO_2 Set Point, Network Configuration.

Figure 7: EEPROM

- **OLED Display and Rotary Encoder**



Provides a local user interface.

Figure 9 :OLED Display and Rotary Encoder

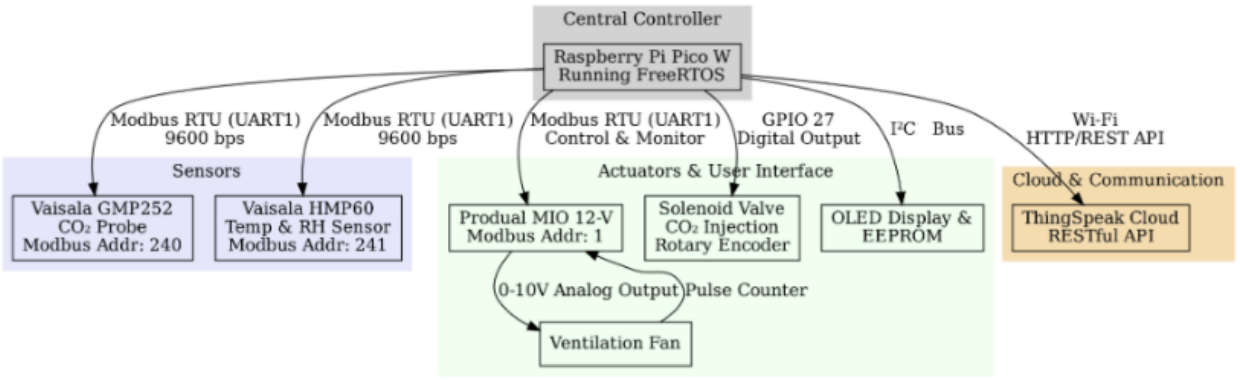
3.2 Software Design

This software runs on FreeRTOS and is written in C and C++ on the RP2040 platform. It separates and performs functions such as performance control, user interface, network communication, and security monitoring.

Task	Function	Key Features
Controller Task	“Brain” of the system, sends/receives updates from other tasks	Closed-loop CO_2 control, safety mechanism, fan control, data communication, memory handling.
UI Task	Manage OLED and encoder input	Menu navigation, value adjustment,SSID/password
Network Task	Handles Wi-Fi and cloud upload, receives user input from Cloud	RESTful API communication
EEPROM Task	Read and wites persistent configuration data	store and update configuration parameters
Hardware Task	Handles communication with greenhouse sensors and actuators	Reads sensor data, sends actuator commands ,givesl feedback to the controller.

Table 1: Software Architecture Diagram

3.3 Block Diagram



4.Cloud Monitoring and Control Interface

4.1 Local User Interface and Data Transmission

The system uses **ThingSpeak** for cloud-based data storage and visualization, allowing users to monitor and manage data online.

4.2 Monitored Parameters

The system sends five specific data fields to the cloud platform:

Field	Parameter	Unit / Description
Field 1	CO_2 Level	ppm(parts per million)
Field 2	Relative Humidity	%
Field 3	Temperature	°C
Field 4	Ventilation Fan Speed	%
Field 5	CO_2 Set Point	ppm(parts per million)

4.3 ThingSpeak Cloud Integration and Access

The ThingSpeak platform enables comprehensive data management and visualization.

4.3.1 Data Access

Users can log in to https://thingspeak.com to view dashboards, sensor readings, and system data.

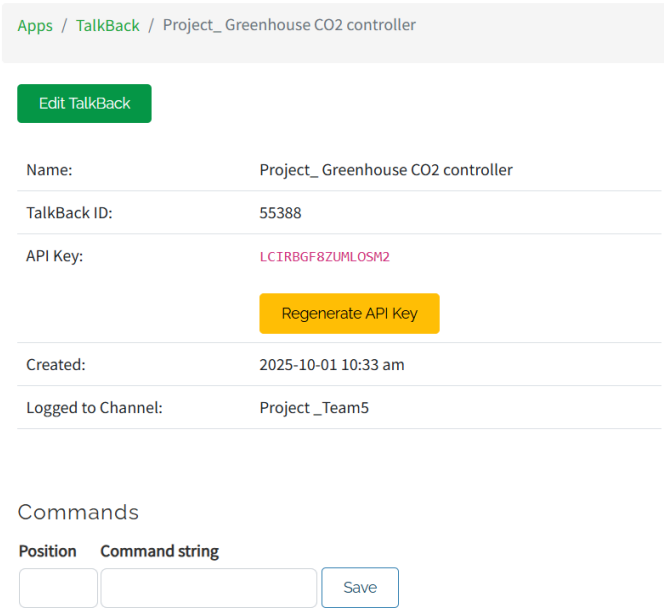
4.3.2 Visualization and Alerts

ThingSpeak displays real-time and historical data through customizable charts. It also supports alerts that notify users when readings exceed normal limits.

4.4 Remote Control (Advanced Feature)

The system's advanced feature enables remote adjustment of the CO_2 setpoint, allowing users to make changes from any location via the cloud interface.

1. Users must Access the ThingSpeak **TalkBack** feature.
2. A command containing the new setpoint value is sent.
3. The system checks for new commands periodically.
4. The new setpoint is applied and confirmed in the next data update.



5. Implementation

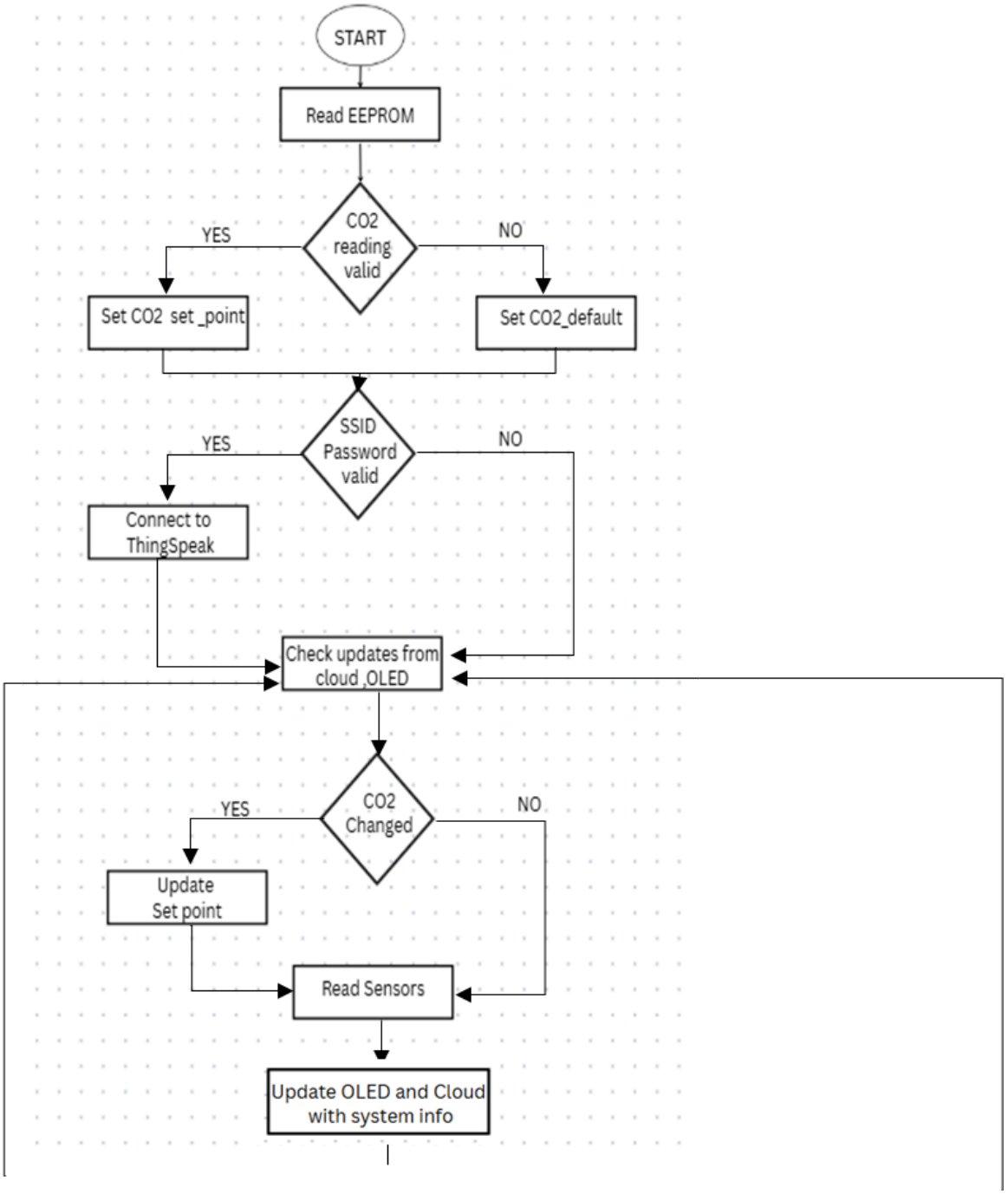
5.1 Operating Principles of the Software

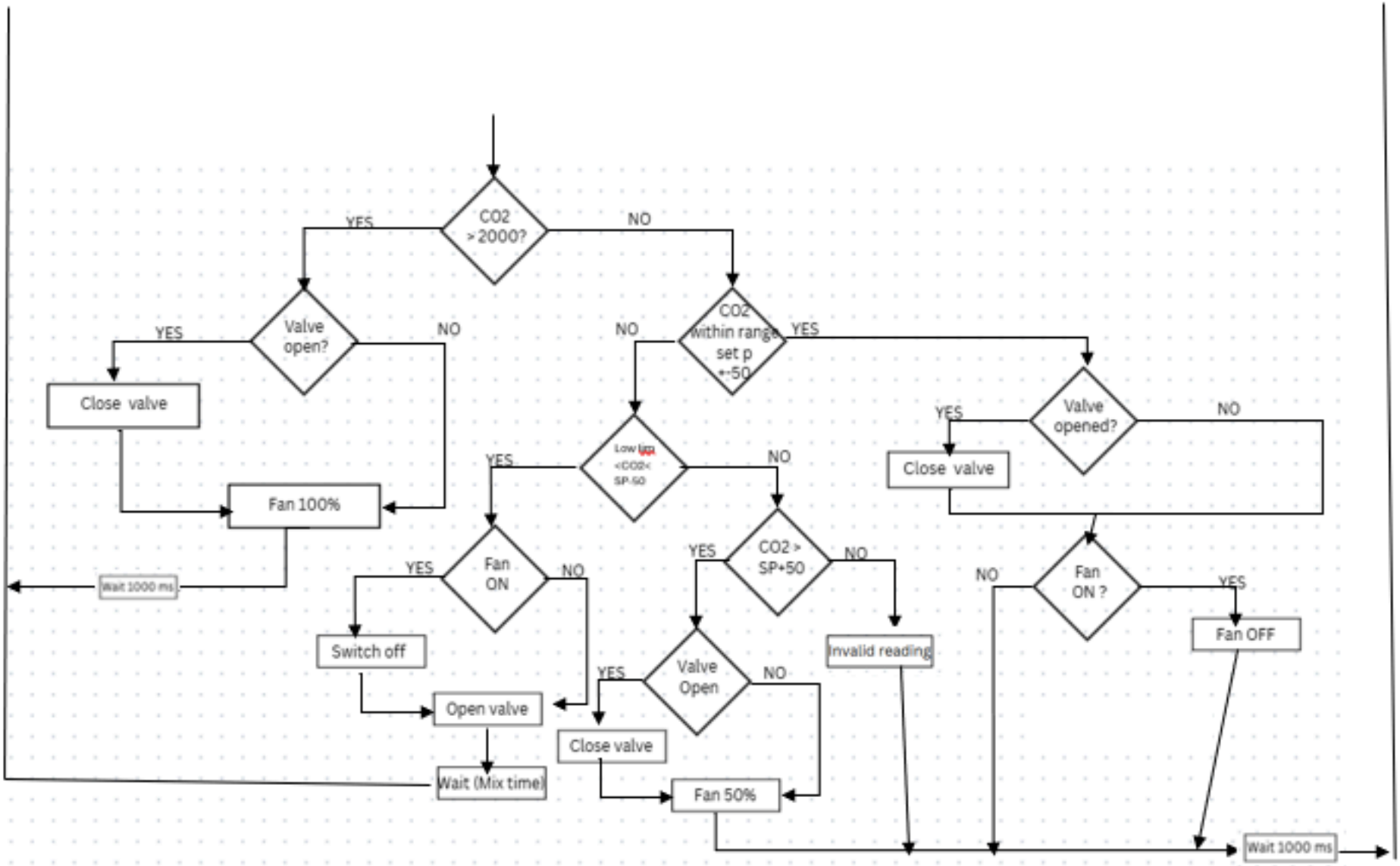
The software operates on a multitasking system built on FreeRTOS, where Independent tasks run simultaneously to manage specific system functions. It continuously checks the CO_2 sensor data, makes control decisions and adjusts actuators to maintain the CO_2 level.

- Real-Time Sensor Monitoring - Reads CO_2 values, temperature and humidity continuously. And the controller task always has access to the latest readings for the continuing process.
- Control Logic - The Controller task is implemented based on the current CO_2 reading and the user defined set point. If the value is too low, it opens the CO_2 injection valve and if the level is too high, activates the ventilation fan to disperse the excess gas.
- Permanent Configuration Storage - All user configurations and system states such as CO_2 setpoint and network credentials are saved and updated to EEPROM. This provides persistence on power failure or reboot.
- User Interface - Shows sensor values and receives setpoint adjustment via a rotary encoder.
- Cloud communication - Logs environmental data transmission to the ThingSpeak cloud as a background process. Sensor readings and system status are sent regular intervals via HTTP POST

- Inter Task Communication - Various tasks share data with the controller task using dedicated message queues. This method provides smooth coordination, avoids data collisions, and ensures consistency of data between all tasks.

5.2 Flow Chart




CO2 > 2000?
YES
NO
Valve open?
YES
NO
Close valve
Fan 100%
Wait 1000 ms
CO2 within range set p +-50
NO
YES
Low lim <CO2< SP-50
YES
NO
Fan ON
YES
NO
Switch off
Open valve
Wait (Mix time)
CO2 > SP+50
YES
NO
Valve Open
YES
NO
Close valve
Fan 50%
Invalid reading
Valve opened?
YES
NO
Close valve
Fan ON ?
NO
YES
Fan OFF
Wait 1000 ms

5.3 Software Component

5.3.1 Essential Libraries and Dependencies

Library	Purpose	Usage in Project
FreeRTOS-Kernel	Real-time operating system	Task scheduling and synchronization
pico_cyw43_arch_lwip	WiFi and TCP/IP stack	Cloud connectivity and HTTP communication
pico_mbedtls	Encryption and security	Secure TLS connections to ThingSpeak
hardware_i2c	I²C communication	OLED display and EEPROM communication
pico_stdlib	RP2040 hardware abstraction	GPIO, UART, and timer operations
Modbus Library	Industrial communication	Sensor data reading via RS-485

Table 2: Key Software Libraries

5.3.2 Core Functions

Function	Task	Description
read_ppm()	CO2Sensor	Reads CO_2 concentration
read_rh() / read_t()	RHTSensor	Reads environment data (temperature and humidity)
setSpeed()	Fan	Controls fan speed (0-100%)
open() / close()	Co2Valve	Controls CO_2 injection solenoid valve
eeprom_write() / eeprom_read()	EEPROM	Stores and retrieve CO_2 setpoint and Network credentials
send_to_thingspeak()	Network	Uploads sensor data via API
Encoder_ISR()	UI	Handles rotary encoder input interrupts

Table 3: Key System Functions

5.3.3 Class Architecture

Class	Responsibility	Key Methods
CO2Sensor	CO_2 concentration sensing	read_ppm()
RHTSensor	Environmental monitoring	read_rh(), read_t()
Fan	Ventilation control	setSpeed(), isRunning()
Co2Valve	CO_2 injection control	open(), close(), pulse()
EEPROM	Configuration storage	eeprom_write_co2(), eeprom_read_string()
PicoI2C	I^2C communication	write(), read(), transaction()
ModbusClient	Industrial communication	Sensor register access

Table 4: Main Class Structure

5.4 Main Program Description

The main() function initializes the greenhouse CO_2 control system, establishing its FreeRTOS-based multitasking architecture. At startup, it enables the necessary hardware interfaces and creates the communication queues required for data exchange between system components.

The program runs by launching the following specific tasks:

- Hardware task – managing sensor readings and actuator control.
- Controller task – implementing CO_2 regulation logic.
- UI task – updating the OLED display and processing user input via the rotary encoder.
- EEPROM task – storing and managing system configuration.
- Network task – managing Wi-Fi connectivity and cloud data communication.

Each task is allocated the necessary resources to maintain a good balance between system stability and real-time response. The main control tasks are given higher priority than background network communication tasks.
Once all tasks are created, control is transferred to the FreeRTOS scheduler, which manages task execution and synchronization. The main() function then enters an infinite loop, ensuring that the system runs in real time.
This architecture is designed to reliably and simultaneously operate environmental monitoring, user interaction, and cloud connectivity, without interrupting the CO_2 control algorithm that is essential for maintaining the greenhouse environmental conditions.

6. Testing and Results

Testing was conducted using both the miniature simulator setup and the actual hardware. Various operational and safety override responses were evaluated. Data retention in EEPROM were investigated

Test Case	**Expected Outcome**	**Result**
CO_2 Control Response	Stabilizes within ±50 ppm of setpoint	Pass
Safety Override	Activates above 2000 ppm	Pass
EEPROM retention	Restores settings after reboot	Pass
Cloud Update	Data visible on ThingSpeak dashboard	Pass

Table 5: Sample Test Results

7. Conclusion

The Greenhouse CO_2 Controller project illustrated how IoT devices and embedded systems could be integrated easily to manage the environment in agriculture systems autonomously. The system was able to regulate the CO_2 level and performed evenly when tested, with stable cloud connectivity through the ThingSpeak platform.

Although some irregularities were noted due to the simplified simulation setup, overall results confirmed that the controller met all relevant design specifications. It maintained stable CO_2 regulation, ensured safety through automated ventilation, control and maintained consistent data storage for reliable operation.

In the future, certain advances would also make it even more practical for real greenhouse applications. Certain future potential advances would include the application of advanced PID control for smoother CO_2 control, field testing in an actual greenhouse, and creating an app for remote operation. Utilizing AI-based environmental prediction could also be feasible for additional energy efficiency as well as optimal plant growth.

Overall, the project is a strong foundation for future smart agriculture solutions that integrate automation, data intelligence, and sustainability.

8. References

1. Government of Manitoba, *Greenhouse CO_2 Supplement.* Manitoba Agriculture. [Online]. Available: https://www.gov.mb.ca/agriculture/crops/crop-management/co2-supplement.html (Accessed: 13 Oct 2025)
2. University of California Agriculture and Natural Resources (UC ANR), *Carbon Dioxide Enrichment of Greenhouses.* [Online]. Available: https://ucanr.edu/blog/nursery-and-flower-grower/article/carbon-dioxide-enrichment-greenhouses (Accessed: 13 Oct 2025)
3. B. Dunn and M. Poudel, *Greenhouse Carbon Dioxide Supplementation.* Oklahoma State University Extension, 2023. [Online]. Available: https://extension.okstate.edu/fact-sheets/greenhouse-carbon-dioxide-supplementation.html (Accessed: 13 Oct 2025)
4. D. Walczuch, T. Nitzsche, T. Seidel, and J. Schöning, "Overview of Closed-Loop Control Systems and Artificial Intelligence Utilization in Greenhouse Farming," in Proc. 2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS), 2022. doi:10.1109/COINS54846.2022.9854938. Available: https://doi.org/10.1109/COINS54846.2022.9854938 (Accessed: 19 Oct 2025)
5. Vaisala Oy, *GMP252 Carbon Dioxide Probe: Technical Datasheet,* 2024. [Online]. Available: https://www.vaisala.com/en/products/instruments-sensors-and-other-measurement-devices/carbon-dioxide-probes/gmp252 (Accessed: 13 Oct 2025)
6. MathWorks, *ThingSpeak API Reference: Write Data and Execute TalkBack Commands,* 2024. [Online]. Available: https://se.mathworks.com/help/thingspeak/writedataandexecutetalkbackcommand.html (Accessed: 13 Oct 2025)
7. Greenhouse CO_2 Controller Specification, Metropolia University of Applied Sciences, 2025. https://oma.metropolia.fi/tyotilat?p_p_id=WorkspacePortlet_WAR_workspaceportlet&p_p_lifecycle=0&p_p_state=normal&p_p_mode=view&_WorkspacePortlet_WAR_workspaceportlet_struts.portlet.action=%2Fworkspace%2Fdocuments%2Findex&_WorkspacePortlet_WAR_workspaceportlet_struts.portlet.mode=view&workspace.id=340049536&workspace.name=ARM%20Processors%20and%20Embedded%20Operating%20Systems%20TX00EX86-3003&currentFolder=10437733
8. Raspberry Pi Ltd., *Raspberry Pi Pico W Datasheet*, May 2, 2024. [Online]. Available: https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf (Accessed: 13 Oct 2025)
9. Produal Oy, *MIO 12-V User Guide*, Version 1.1.4, Oct. 24, 2018. Produal Oy, Keltakalliontie 18, 48770 Kotka, Finland. [Online]. Available: https://www.produal.com. (Accessed: 15 Oct 2025).
10. Vaisala Oyj, *HMP60 Humidity and Temperature Probe — Datasheet (B210851EN-K)*, Rev. K, 2024. [Online]. Available: https://docs.vaisala.com/v/u/B210851EN-K/en-US. (Accessed: 15 Oct. 2025).
11. Microchip Technology Inc., *24LC256 EEPROM — Datasheet*, DS20001203V, 2022. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/24AA256-24LC256-24FC256-Data-Sheet-20001203V.pdf. (Accessed: 15 Oct 2025).