Greenhouse CO_2 controller specification

Introduction

The goal of the project is to implement a CO_2 fertilization controller system for a greenhouse. The controller keeps the CO_2 level at the level that is set from the local user interface. The upper limit for CO2 setting is 1500ppm. The settings are stored in EEPROM to make them persistent. For safety reasons if the CO2 level exceeds 2000ppm at any time ventilation is increased to reduce the CO2 level to the requested level.

Controller connects to the cloud using restful API and sends the measured CO_2 level and other environmental information to the cloud for display and/or further processing. The network parameters needed for connection can be set from the local user interface and are stored in the EEPROM.

Hardware platform

Fan speed is controlled by Produal MIO 12-V, which is a Modbus controlled IO-device that has its 0 – 10V output connected to the ventilation fan control input. MIO output voltage determines the speed of the fan: 0V is off (0%) and 10V is maximum speed (100%). Ventilation fan has a pulse output that gives pulses when the fan is turning, and it is connected digital pulse counter input of MIO. The counter is self-clearing, which means that the counter is reset to zero after the count is read through Modbus. When read periodically if the value is zero after two reads it means that the fan is stopped.

Produal MIO 12-V

- Modbus address: 1
- 0-10V output (AO1) controls the ventilation fan speed
- Digital input (AI1 digital counter) counts rotation pulses from the fan. Counter value is used to check if the fan is running. Counter is self-clearing

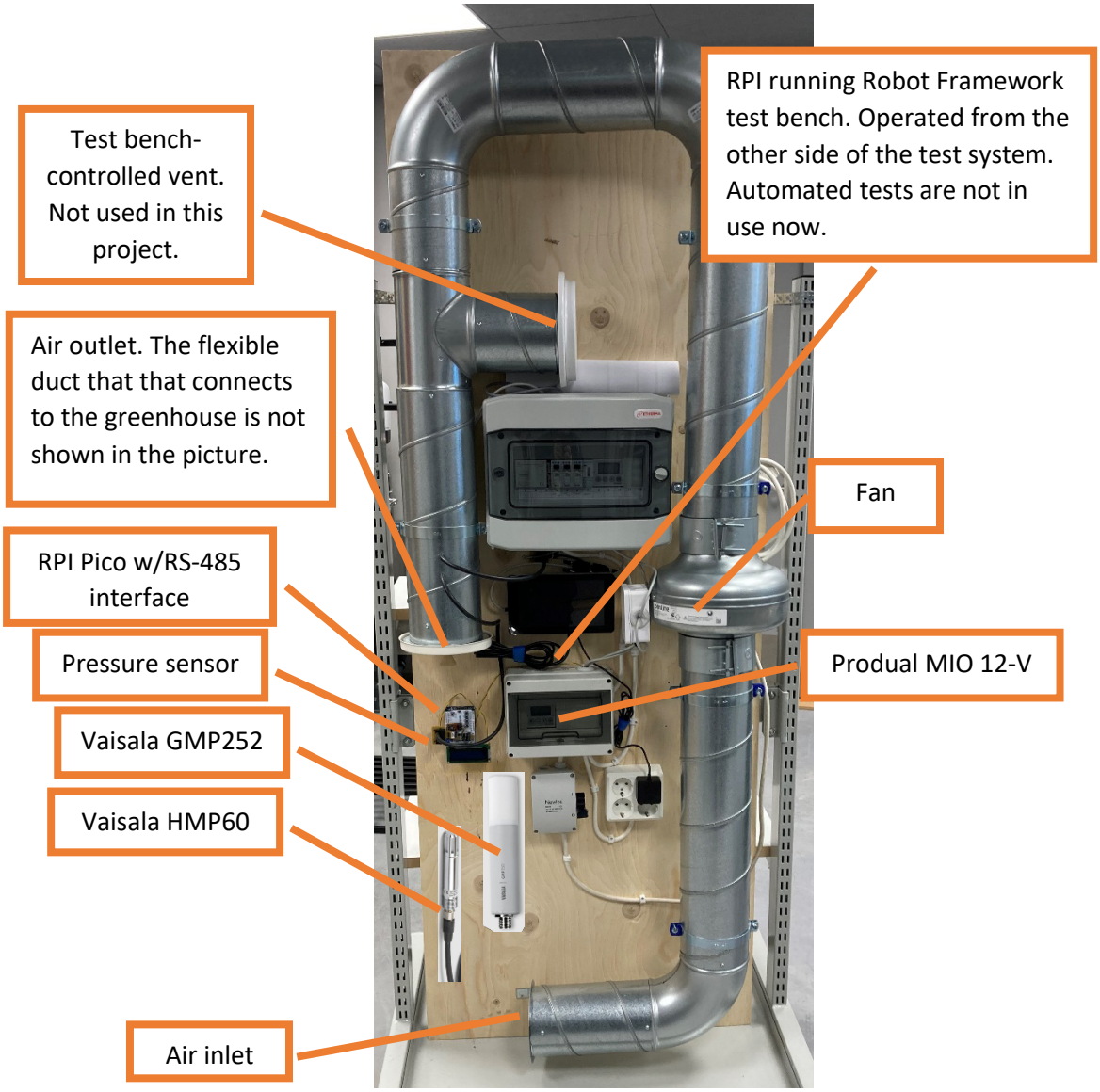
The system is equipped with three sensors. All modbus devices operate at 9600 bps speed. See data sheets of the sensors in the course workspace for more information.

- Vaisala GMP252 CO2 probe
    - Modbus address: 240
- Vaisala HMP60 relative humidity and temperature sensor
    - Modbus address: 241
- Sensirion SDP610 – 125Pa pressure sensor
    - I^2C bus
        - Address: 0x40
        - SCL → Pin 15
        - SDA → Pin 14
    - Differential pressure sensor which connected to measure pressure difference between room and the ventilation duct
    - Sensor returns a signed 16 bit value
    - The value must be converted to physical value (pascals)
        - Scale factor (see data sheet chapter 2)

- Altitude correction (see data sheet chapter 5)
- Hose length compensation (see data sheet chapter 8) – not needed for hose length up to 1 m

The relay that controls CO2 injection valve is connected to GPIO27. The test greenhouse is very small and opening the valve even for a few seconds may cause CO2 level to exceed acceptable limits. In the real greenhouse the valve should be opened for no more than 2 seconds at a time followed by wait period to let CO2 disperse in the greenhouse and for the sensor reading to stabilize.

Test system



Miniature test system

Miniature test system consists of similar components as the “big” test system with the exception that some of the components are simulated by another Pico. The physical parts are much smaller which has an impact on the performance of the miniature test system. For example, the miniature system’s fan is not able to produce as high a pressure difference as the actual system which may have an impact on the controller performance on miniature system and vice versa.

However, from the Pico software perspective the two test systems are identical, and the software should run on both systems without any modification.

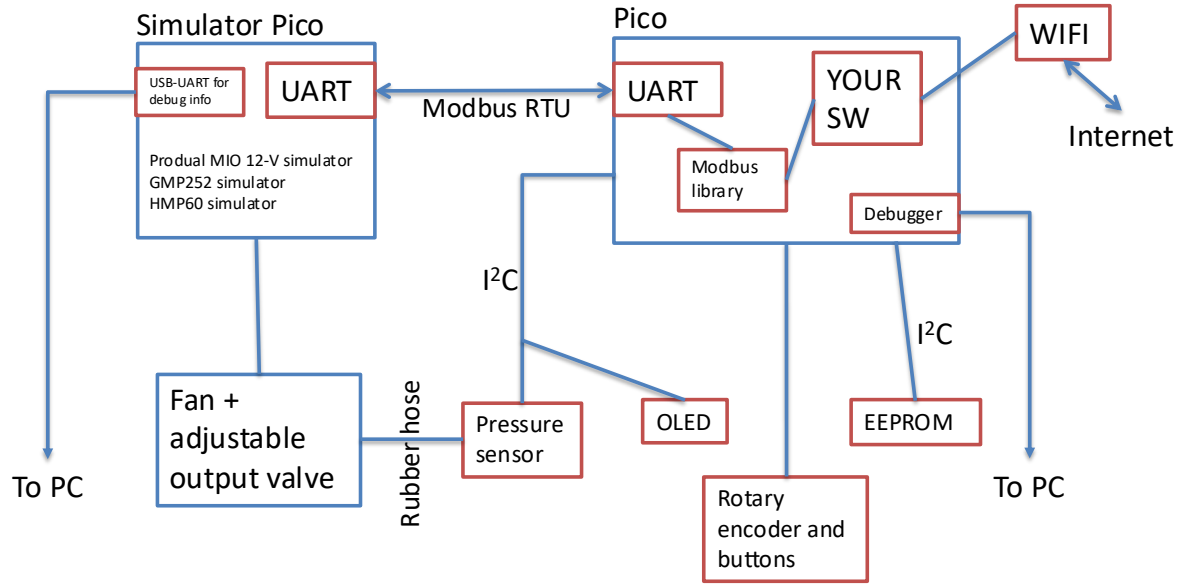
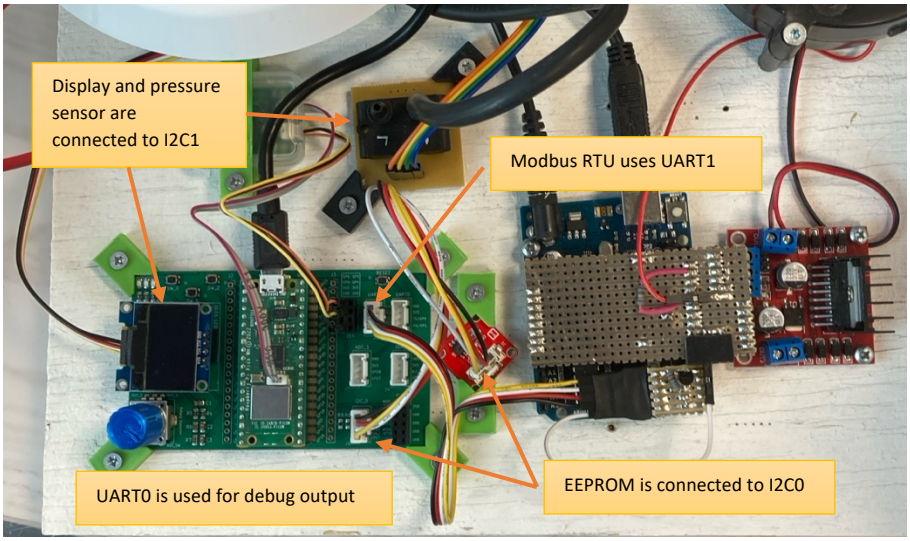


Figure 1 Miniature test system diagram



Cloud connection

The measured data and some control parameters are sent to ThingSpeak cloud service using RESTful API. The API is also used to receive commands to change the set point.

The following information is to be sent to the cloud at regular intervals:

1. CO_2 level (ppm)
2. Relative humidity
3. Temperature
4. Ventilation fan speed (0-100%)
5. CO_2 set point (ppm)

The data format is simple: fields are numbered as above, and data fields are sent in a HTTP POST request. The format is :

<name of the field>=<field value>&<name of the field>=<value>&…

Field names are: fieldX, where X is a number between 1 and 8. For example, CO_2 level of 450 ppm is sent as “field1=450”. All fields should be sent in the same message. There are also additional named fields, for example time stamp or status, that can be sent along with the measurement data.

See: https://se.mathworks.com/help/thingspeak/writedataandexecutetalkbackcommand.html for more details about the data format.

Credentials that are needed to send data and receive commands are provided in the workspace.

Pico sensor/IO simulator

Pico software simulates Vaisala sensors and simulates dissipation of CO_2 only to some extent. The most important simulation parameters are displayed on the simulator OLED screen.

It should be noted that simulator has some shortcomings:

- CO_2 dissipation is modelled with a simple ad-hoc formula that does not match real world behaviour to any extent. It is just to help with rudimentary testing of keeping CO_2 level stable.
- Increased ventilation does not reduce the CO2 level at the same rate as it does in the real greenhouse.

You can open simulator Pico’s serial port with speed 115200 bps to see simulator status messages and to set values to sensors. There is no local echo, the characters you type are not echoed back to you, so you either have enable local echo on your terminal or have steady fingers. Vaisala sensor values can be set by simple commands: `co2` (or `ppm`) , `rh`, and `t` followed by value you want the sensor to report. The value that is read from the command is written to the terminal when you press enter after the command.

CO_2 dissipation is controlled with command `dec`. The command sets CO_2 decrement per second in ppm/s. Value of 2.5 means 2.5 ppm/s, 1 is 1 ppm/s etc. CO_2 level is never decremented below 200 ppm.

CO_2 injected in to the green house by opening a solenoid valve that when open lets CO_2 in the green house at the rate of 17l/min. This valve is simulated by increasing CO_2 level 1 ppm/s when the valve is open. It has not been verified that the rate matches the real-world case – most likely it does not.

Project template

Project template can be found here:

https://gitlab.metropolia.fi/lansk/rp2040-freertos.git

Minimum requirements

To have the project approved you need to implement at least the local UI with persistent CO_2 level setting. All sensors on the system must be read and displayed on the local UI. You also need to provide the following documentation:

- User manual
    - How to operate your system through the UI
    - How to operate the system remotely (if implemented)
- Program documentation
    - Class diagrams, block diagrams, flow charts etc.
    - Implementation principles

The greenhouse controller should be divided into tasks with clear responsibilities. For example: network task, controller task, UI task, etc.

Advanced features

Reports measured values to Thingspeak

Can be controlled with commands that are sent using Thingspeak talkback queue.