Expertise
and insight
for the future

ABB Ventilation Control System

Metropolia University of Applied Sciences

Bachelor of Engineering

Internet of Things TX00CI63-3006

Daniil Marinec
Dongbin Yang
Long Pham
Kendrick Kwong

30.10.2022

Contents

1 Project Description

The project is combination of a Ventilation Fan System with Vaisala's GMP252 Carbon Dioxide probe, HMP60 Temperature and Humidity sensor. Fan speed inside the duct can be controlled with Produal MIO 12V Actuator, while pressure difference between the inside and outside of the duct will be measured with Sensirion's SDP600 differential pressure sensor.

2 User Manual

The idea of the system is to measure surrounding environments attributes, specifically CO2 level, relative humidity and temperature, then using the collected information as feedback for the ventilation fan to adjust power level so as to create the ideal state in terms of air conditions. The system operates in 2 separate modes:

2.1 Operation

- Automatic Mode:

  Automatic mode lets the user decide the desired parameter of pressure inside the ventilation duct, either through an LCD user interface implemented physically on the system or through a web interface.
  Controller measures the pressure difference between pressure level in the duct and in the surrounding environment, adjust the fan speed to keep it at a required level.

- Manual Mode:

  User is free to control the fan speed by themselves during the operation in manual mode, giving them full access to modify the fan speed level arbitrarily. This will change the pressure difference between the outer space and inside the duct accordingly.

2.2 Interface

The ventilation system supports two Interfaces – local LCD interfaces and remote interface through internet. It allows user to manipulate the system efficiently. The system can be controlled not only with LCD display and buttons on device, but also it provides the possibility to control it through internet on web interface.

- Web Interface

User is required to log in using their credentials. There is a page where users can create their account if they have not had one. A page of auto mode will be shown on default after user have logged in, which displays the current fan speed inside the duct. In addition, the pressure level will be shown and the user can modify it through an input field, and the fan speed will adjust until it reaches the desired pressure level. There is a switch in the page to change the operation mode of the system to manual and vice versa. Manual mode tab contains a chart that displays the current pressure value of the system. User can change the fan speed arbitrarily and the other components of the system shall act accordingly. User can inspect statistical data through the **See Full Statistics** button, where it will display the graph of pressure data, fan speed data and the user data. These graphs can be displayed in a better solution, separately.
For user's activities analytics, it will show the amount of time that individual user spent on the page in a scale of 24 hours.

- LCD Interface

The LCD screen is physically connected to the system and can be controlled with *4 buttons*. Due to lacking of physical buttons, 3 out of those 4 buttons can be combined with a control button to perform an alternative task. **A1** is the control button which will control how other buttons behave. **A2** and **A3**, in order, work as *UP* and *DOWN* buttons when user is browsing through the menu, and as *CONFIRM* and *BACK* when combined with the A1. **A4** performs the task of switching operation mode and will control whether the program should make connections with sensors and fan retries. **LCD** will display the values of the system measured in **both** operation modes. In auto

mode, users can only configure the desired parameter of pressure inside the ventilation duct, the system will adjust the fan speed of the system and the LCD screen will display the updated value of other data. On the other hand, user can only configure the fan speed in the manual mode, in which the LCD screen will also continuously updating the remaining measurement results to the screen. Every other state on the LCD menu are *read-only* values and cannot be configured.

Note: *If the operation mode is modified when the user is setting a parameter of the system, all pending changes will not be saved and the parameter will be reset back to its original value.*

A word on the interface flags. First flag to appear after the screen name is **!** - it marks if the desired pressure level cannot be achieved by the fan in automatic mode. Second flag is ***** - it appears on the screen when device will not respond to any commands, while it performs some task. Third flag is **U** and **D** - meaning that current selected sensor or actuator corresponding to the screen is up or down accordingly. Fourth flag is **M** and **A** - meaning the device operates in manual or automatic mode.

3 Documentation

The system is controlled by an LPC1549 Microcontroller. The LPC1549 is an ARM-Cortex M3 based microcontroller with multiple peripherals and lower power consumption. It contains built-in Nested Vectored Interrupt Controller (NVIC), Systick Timer, up to 72MHz in CPU clock speed, and a flash memory of 256kB. This magic board also supports a various type of protocols, with UART, I2C and Modbus being the most noteworthy ones. In general, it is powerful and power-efficient, suitable for embedded devices.

3.1 Sensors

CO2 and RH&T Sensors are read with an interval of 0.5 seconds continuously to avoid causing system overloading and freezing, while Pressure Differential Sensor measurements are taken every 13 milliseconds.

These are estimates on theory and will not work exactly as each Modbus/I2C transaction takes around 20 milliseconds. In case a sensor went down, it would be mark as D instead of U on the LCD UI and the last value measured will be used until the next valid read lands.

3.1.1 GMP252 CO2 Sensor

GMP252 CO2 Sensor communicates with LPC1549 through Modbus RTU over RS-845 interface. The Modbus address for the sensor is 240. The measurement range is 0 <= co2 <= 10000 ppmCO2 (can reach up to 30000 ppmCO2 with reduced accuracy). It can occasionally take the most recent relative humidity and pressure values as reference to produce extra precision in readings.

3.1.2 HMP60 Relative Humidity and Temperature Sensor

HMP60 RH&T Sensor also uses Modbus RTU over RS-845 interface to do transactions with the main control board, having a Modbus address of 241. The limit for temperature measurements lies in range of −40 °C <= t <= - 60 °C and 0% <= rh <= 100% for relative humidity reads.

3.1.3 SDP600 Differential Pressure Sensor

SDP600 make use of I2C communication to exchange data with the MCU, with an I2C address of 0x40. The limit range for SDP600 measurements in standard operation is -500Pa <= p <= 500Pa. The value read from the sensor is a two's complement integer and must be converted with a suitable scale factor, in our case 240 Pa^-1.

3.2 Actuator

The Produal MIO 12-V IO device is attached to the system as the only actuator. Its usage is to control the operation inside the duct, adjusting the fan speed to serve the wanted conditions to end user. The device uses Modbus with an address of 0x01. The fan produces pulses that is being monitored by MIO which reports the status of the fan

(if it is on spinning or not). It uses an output level of 0-10V with 0V being no output at all and 10V setting the fan to the highest power.

3.3 Communication Protocols

3.3.1 Message Queuing Telemetry Transport (MQTT)

MQTT was used as a bridge for communication purposes between the Internet part and the Things part of the system. Whenever a change occurs, either the modification was made through LCD or Web Interface, a JSON-formatted message is published to MQTT server's topic, to which either the LPC1549 or the webpage subscribe. Both parties should receive messages from the MQTT Broker, react and work in sync.

3.3.2 Modbus Remote Terminal Unit (Modbus RTU)

Most of the devices we were working with communicates through Modbus RTU. Although Modbus was not super difficult to use, the process of checking control registers and reading floating point values was challenging.

3.3.3 Inter-Integrated Circuit (I2C)

I2C is made use of by the SDP600 Pressure Sensor.

3.4 Website

3.4.1 Front-End User Interface

The front end is a basic page using EJS (Embedded JavaScript) Templating Language along with CSS. In order to use the Web Interface, a user must be authenticated. The login authenticated flag is kept in check by cookies shared by the client side with the server. Live data being displayed in charts (with the help of ChartJs) are fetched through a data route every 5 seconds using AJAX calls. Most views have logout buttons for the sake of convenience for end users. Session data is also managed to auto-

matically log a user out if they do not perform the log out manually. The input fields, intended for posting data to server requires confirmation with a button click/Enter press to prevent accidental touches/keyboard scraps.

3.4.2 Back-End

- Server

The language choice for backend server was NodeJs along with Express. The server was implemented with simplicity in mind, so it only contains most basic routes for login, signup, logout, posting data and sending views. The authentication system used PBKDF2 for password encryption. MQTT messages were not being polled to reduce overload potential of the server. The server has access to cookies and it takes part in managing the state of the website.

- Database

MongoDB was the option for this project due to ease of use and its compatibility with Nodejs and its popularity among the web devs community. The app uses 2 collections, users and stats. Users collection, obviously contain user credentials, activities history. Stats collection store data collected from the devices and received from MQTT's server. User's ID can be implemented but with Mongo's built-in auto ID assignment it is not necessary for now.

4 Further Development

The system requires more further development in order to provide complete functionalities and bridge the interaction from web interface to the embedded end. The further modifications to system concern several topics:

- Implementing MQTT to work properly on the embedded side is a foreseeable challenge waiting to be solved.
- Categorizing account type for admin and casual user will also be considered carefully.
- Selenium Robot Framework Test should be written for web testing purposes as it could be very problematic to test the whole website manually.
- The embedded side has some inconsistence, data readings could very well be wrong and requires small delays to retry the reading turn.

- User's usage time chart on the web contains minor cases with inconsistent behaviours if the user uses it from one day to another (through the night at the 12AM mark).

- After posting data on the web, the site will be reloaded as a default action, which could be changed to provide a better user's experience.

- Error handing has not been implemented for the web side. If no data is received from the physical devices, an error message should be displayed.

References

Device Manuals

- Project specifications
- GMP252 CO2 Sensor Manual
- HMP60 Temperature and Humidity Probe Manual
- Produal MIO 12V Manual
- SDP600 Differential Pressure Sensor Manual
- LPC1549 Manual

Contributors

- Daniil Marinec
- Dongbin Yang
- Kendrick Kwong
- Long Pham