Documentation eVent IST Firmware

Table of Contents

[1 Modes 2](#_Toc38292030)

[2 Pinout 2](#_Toc38292031)

[3 States 3](#_Toc38292032)

[4 Variables 3](#_Toc38292033)

[5 Calculations 5](#_Toc38292034)

[5.1 Volume 5](#_Toc38292035)

[5.2 Inhale/Exhale Time 5](#_Toc38292036)

[5.3 Velocity 5](#_Toc38292037)

[5.4 Flow 6](#_Toc38292038)

[6 Controller 7](#_Toc38292039)

[7 State Machine 7](#_Toc38292040)

[8 Alarms 8](#_Toc38292041)

[9 Installation Guide 8](#_Toc38292042)

[9.1 Arduino IDE 8](#_Toc38292043)

[9.2 Atom IDE 8](#_Toc38292044)

# Modes

Automated operation of ventilation bags in two Modes:

* Volume controlled ventilation with adjustable Parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Range** | **Unit** |
| Volume | 300 - 900 | mL |
| Breath rate | 8 – 40 | Breaths per minute |
| Inspiratory : Expiratory ratio | 1:1 – 1:4 |  |

Table 1 - volume control parameters

* Pressure controlled ventilation with adjustable Parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Range** | **Unit** |
| Pressure | 10 - 30 | mBar |
| Breath rate | 8 – 40 | Breaths per minute |
| Inspiratory : Expiratory ratio | 1:1 – 1:4 |  |

Table 2 - pressure control parameters

# Pinout

The firmware is designed for an Arduino Mega. Nevertheless, the hardware should be compatible with the MIT project, which uses an Arduino Uno, because all mandatory functions have been mapped to Uno pins. The used pins are defined in the *pins.h* file.

|  |  |  |  |
| --- | --- | --- | --- |
| **Definition** | **Pin** | **Configuration** | **Description** |
| LCD\_D7 | 0 | digital output | LCD data line |
| LCD\_D5 | 1 | digital output | LCD data line |
| ENC\_A | 2 | digital input (INT)° | encoder impulse signal A |
| MOTOR\_PWM | 3 | PWM output | defines speed of motor |
| LCD\_D4 | 4 | digital output | LCD data line |
| LCD\_ENABLE | 5 | digital output | LCD control signal |
| LCD\_RS | 6 | digital output | LCD control signal |
| LCD\_D6 | 7 | digital output | LCD data line |
| CONFIRM\_BUTTON | 8 | digital input (DB\*) | momentary button to confirm |
| BUZZER | 9 | PWM output | piezo buzzer |
| ENC\_B | 10 | digital input | encoder impulse signal B |
| SILENCE\_BUTTON | 11 | digital input (DB\*) | momentary button to mute alarms |
| MOTOR\_DIR | 12 | digital output | defines direction of motor |
| LIMIT\_SWITCH | 13 | digital input | state of the limit switch, defines home |
| VOL\_POT | A0 | analog input | sets volume setpoint |
| RATE\_POT | A1 | analog input | sets breath rate setpoint |
| I2E\_POT | A2 | analog input | sets inspiration/expiration ratio setpoint |
| PRS\_POT | A3 | analog input | Sets maximum pressure setpoint |
| MODE\_SWITCH | A4 | analog input | three states: VCC | VCC/2 | GND |
| PRESSURE | A5 | analog input | signal of pressure sensor 0.2-4.7V |
| LCD\_BACKLIGHT | 14 | digital output | LCD backlight on/off via transistor |
| LED\_POWER | 15 | digital output | LED to indicate that Arduino booted |
| LED\_GREEN | 16 | digital output | LED to indicate active mode |
| LED\_RED | 17 | digital output | LED to indicate error |
| ENC\_I | 18 | digital input | not implemented |
| MOTOR\_SENSE | A8 | analog input | measure motor current (1.65V/A) |
| MOTOR\_BRAKE | A9 | digital output | not implemented |

Table 3 - pinout

\*debouncing necessary °Interrupt

# States

In the *states.h* file, the states of switches, buttons and the motor are defined to provide easy adjustment to different electrical implementation. For example, one might use external pull-down resistors instead of the internal pull-ups, then it would be necessary to change the “BUTTON\_PRESSED” state to “HIGH”. Or normally-closed buttons or switches are used instead of the normally-open switches we used.

# Variables

A number of constants in *main.cpp*are defined. They are used to adjust the firmware to parameters of the electro-mechanical construction. Moreover, some clinical process-related constants are set.

|  |  |  |
| --- | --- | --- |
| **Name** | **Unit** | **Description** |
| maxMotorCurrent | mA | if this current is exceeded the motor will be stopped |
| diffBagPosition | T\* | ticks between home and fingers touching the bag |
| timeHold | ms | time between inhale and exhale |
| pressurePlateauMax | mbar | maximum pressure between inhale and exhale |
| velocityHoming | DCS° | defines the motor speed while homing |
| controlPeriod | ms | period time of the velocity controller |

Table 4 - constants

The following tables describe the used variables separated into functional groups.

|  |  |  |
| --- | --- | --- |
| **Name** | **Unit** | **Description** |
| lastMode |  | the mode which was active before the current mode |
| mode |  | the current mode |
| nextMode |  | the mode which should be active after the current mode |
| state |  | the current state in the state machine |

Table 5 - state machine variables

|  |  |  |
| --- | --- | --- |
| **Name** | **Unit** | **Description** |
| homePosition | T\* | encoder value at the position of the limit switch |
| currentPosition | T\* | last encoder reading |
| bagPosition | T\* | position where fingers touching the bag |
| inhaleEndPosition | T\* | position at the end of the inspiratory phase |
| prePeriodPosition | T\* | ticks before the current velocity control period |

Table 6 - positioon variables

|  |  |  |
| --- | --- | --- |
| **Name** | **Unit** | **Description** |
| volumeTidalSet | mL | unconfirmed set value for volume tidal |
| breathsPerMinuteSet | 1/min | unconfirmed set value for the breath rate |
| I2E\_ratioSet |  | unconfirmed set value for the I:E ratio |
| limitPressureSet | mBar | unconfirmed set value for the pressure limit |
| lastVolumeTidalSet | mL | last unconfirmed set value for volume tidal |
| lastBreathsPerMinuteSet | 1/min | last unconfirmed set value for the breath rate |
| lastI2E\_ratioSet |  | last unconfirmed set value for the I:E ratio |
| lastLimitPressureSet | mBar | last unconfirmed set value for the pressure limit |

Table 7 - set - potentiometer variables

|  |  |  |
| --- | --- | --- |
| **Name** | **Unit** | **Description** |
| breathsPerMinute | 1/min | breaths per minute, also called respiratory rate (RR) |
| I2E\_ratio |  | ratio of inhale to exhale time |

Table 8 - clincal variabels

|  |  |  |
| --- | --- | --- |
| **Name** | **Unit** | **Description** |
| targetTimeInhale | ms | calculated length of the inspiratory phase |
| targetTimeExhale | ms | calculated length of the expiratory phase |
| volumeTidal | mL | total volume delivered to the patient |
| volumeTicks | T\* | encoder ticks necessary to deliver the volume |

Table 9 - volume mode variables

|  |  |  |
| --- | --- | --- |
| **Name** | **Unit** | **Description** |
| currentFlow | ml/s | calculated flow in last controller period |
| targetFlow | ml/s | flow which should be reached according to calculations |
| meanFlow | ml/s | mean flow which should be reached over whole inspiration |
| targetInhaleVelocity | T\*/s | target input of controller for inspiratory phase |
| targetExhaleVelocity | T\*/s | target input of controller for expiratory phase |
| currentVelocity | T\*/s | measure input of controller |
| dutyCycleOutput | DCS° | output of controller, scaled from 0-255 for PWM Output |
| integral |  | integral value for I- controller |
| iTerm |  | I factor of controller |
| pTerm |  | P factor of controller |

Table 10 - controller variables

|  |  |  |
| --- | --- | --- |
| **Name** | **Unit** | **Description** |
| pressurePlateau | mbar | Last measured plateau pressure (between I and E phase) |
| limitPressure | mbar | set limit pressure |

Table 11 - pressure variables

\* Ticks °Duty Cycle Steps

Additionally, several *timer variables* are used to handle non-blocking timing in the state machine.

# Calculations

## Volume

To identify the relation between the encoder steps moved, and the volume delivered to the patient, a test was performed. While varying the encoder distance, the delivered volume was measured with a lung simulator “Dräger LS800”. A “Laerdal bag adult” and an encoder with a resolution of 500 steps per rotation was used.

As the results were very close to linear, a model of two linear functions was considered reasonable to define the relation.

Figure 1 – Pumped air volume per step

## Inhale/Exhale Time

The inhale/exhale time is calculated with the user defined parameters breath rate and I:E ratio.

## Velocity

To be able to control the velocity, the current velocity is calculated with the covered encoder steps in the last time period.

## Flow

Despite the flow is not measured by the system, it should be controlled to provide an accelerating behaviour at the inspiration phase of the volume control ventilation. For that reason, it is calculated based on the calculated volume over a known time period.

The flow should be slowly increased and then stay constant till the end of the inspiration. However, the mean flow over one inspiration phase has to be met to attain the required inhale time.

It was defined that the inspiration should start with 20% of the mean flow. It then increase linear till half the inhale time is reached. To accomplish that, the following model is used to calculate the current target flow.

Figure 2 – Flow model

(fmean = 30 L/min)

# Controller

To control the velocity and pressure, a PI controller was implemented. To achieve that, the function “*controlInhaleVelocity”* (or exhale) and “controlPressure” is called after every control period. The control period is defined by the constant *“controlPeriod”* and is hardcoded to be 50ms.

The controller calculates the control difference and adjusts the PWM output, which controls the speed of the motor, depending on the *“pTerm”* and *“iTerm”* settings. The preconfigured values for the control terms have been chosen by testing with a lung simulator “Dräger LS800”. They may vary depending on the used motor. More information about PI controllers can be found on <https://en.wikipedia.org/wiki/PID_controller#PI_controller>.

# State Machine

The firmware differs between 4 modes:

* Idle
* Homing
* Volume
* Pressure

The user can select between idle, volume-controlled ventilation and pressure-controlled ventilation. Every ventilation mode starts with a homing sequence to calibrate the positions.

The state machine is mainly designed after the concept of MIT which, at the moment of writing, can be found at: <https://e-vent.mit.edu/controls/high-level-controls/>

|  |  |  |
| --- | --- | --- |
| **State** | **Action** | **Condition** |
| 0 | Move fingers outwards | Limit switch reached? |
| 1 | Move fingers inwards, set home position | Limit switch not reached? |
| 2 | Move to bag position | Bag position reached? |

Table 12- Homing states

|  |  |  |
| --- | --- | --- |
| **State** | **Action** | **Condition** |
| 0 | start homing | homing done? |
| 1 | move fingers inhale, control velocity | Inhale end position reached? Start timer |
| 2 | reset control parameters for exhale | Time hold reached? |
| 3 | move fingers exhale, control velocity | bag position reached? |
| 4 | stop, wait for rest of single breath time | Breath time reached? |

Table 13 - Volume states

|  |  |  |
| --- | --- | --- |
| **State** | **Action** | **Condition** |
| 0 | start homing | homing done? |
| 1 | move fingers inhale, control pressure | Inhale time or close position reached? |
| 2 | reset control parameters for exhale | Time hold reached? |
| 3 | move fingers exhale, control velocity | bag position reached? |
| 4 | stop, wait for rest of single breath time | Breath time reached? |

Table 14 - Pressure states

# Alarms

|  |  |
| --- | --- |
| **Alarm** | **Reason** |
| LOW VOLUME | The minute-volume in pressure mode fell below the hardcoded minute-volume minimum. (minVolumeperMinute in alarms.cpp, def: 3Liters) |
| MOTOR ERR | Either the motor is unplugged or blocked. Alarm occurs if the current exceeds the hardcoded maximum motor current or the motor was not able to move for 2 seconds. (maxMotorCurrent in alarms.cpp, def: 2000mA) |
| PRES ERR | The pressure is either too high or too low. High pressure would indicate a bad compliance of the lung, low pressure would indicate leakage. (pressurePlateauMax, pressureMax, pressureMin in alarms.cpp) |

To make the alarm louder, simply replace the “tone()” functions through “digitalWrite(BUZZER, HIGH) and “noTone()” to digitalWrite(BUZZER, LOW).

# Installation Guide

There are two different software versions included in the project. First, an Arduino IDE project file (.ino). You can use this file, if you have a similar electromechanical setup and just want to flash your Arduino without editing the project. The advantage of the Arduino IDE is, that it is very easy to get started. The disadvantage is, that the structure of an Arduino .ino file does support separating the code into different files. Therefore, the structure of the code is very confusing, and it is hard to do any changes.

The Atom IDE project is more complicated to install but provides a more structured view of the code.

## Arduino IDE

1. Download and install Arduino IDE: <https://www.arduino.cc/en/main/software>
2. Open the .ino file with Arduino IDE.
3. Connect the Arduino to the PC via the USB cable.
4. In the IDE under “Tools” select the port with your Arduino.
5. Click Upload

## Atom IDE

1. Download and install Atom IDE: <https://atom.io/> (you will need to install Python 2.7 as well)
2. In Atom, open the package manager (File/Setting/Install) and install Platformio-ide
3. When restarting Atom, the platform io home screen should appear. Press Open Project and find the eVent Folder on your PC.
4. Connect the Arduino to the PC via the USB cable.
5. Select Upload from toolbar on the left side.