Documentation eVent IST Firmware

Table of Contents

1 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

2 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

3 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.

4 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.

5 Calculations

5.1 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.

$$Volume[mL] = Steps * 8 - 190 \quad | \, for\ Steps < 60$$

$$Volume[mL] = Steps * 12 - 420 \quad | \, for\ Steps > 60$$

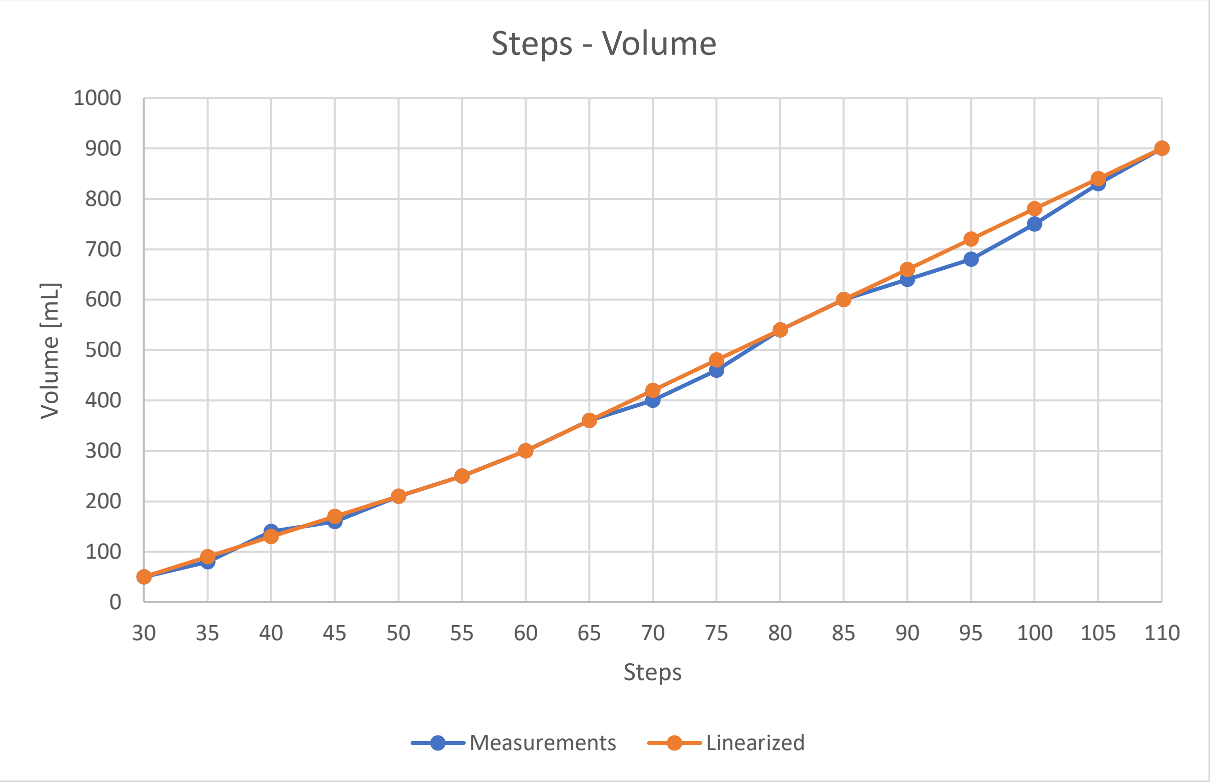

Figure 1 – Pumped air volume per step

5.2 Inhale/Exhale Time

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

$$T_{single\ breath}[ms] = \frac{60000}{breath\ rate} - T_{hold} \quad | \, T_{hold}\ \dots break\ between\ I\ and\ E$$

$$T_{inhale} = T_{single\ breath} * \frac{1}{1+\ I:E} \quad | \, Example\ I:E\ = 1:\mathbf{2} \rightarrow\ T_{single\ breath} * \frac{1}{1+\mathbf{2}}$$

$$T_{exhale} = T_{single\ breath} - \ T_{inhale}$$

5.3 Velocity

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

$$velocity\left[\frac{T}{s}\right] = Ticks\left[\frac{T}{P}\right] * \frac{1000}{P[ms]}$$

5.4 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.

$$flow[\frac{ml}{s}] = velocity\left[\frac{T}{s}\right] * 8 \quad or\ flow = velocity * 12\ , depending\ on\ position$$

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.

$$flow_{mean} = \frac{Volume[T]}{Time_{inhale}[s]} * 12$$

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.

$$flow = flow_{mean} * (0.2 + \frac{1.1 * \mathrm{CC}}{0.5 * CC_{max}}) \mid \mathrm{CC} \rightarrow \mathrm{Control\ Cycle} < \frac{CC_{max}}{2}$$

$$flow = flow_{mean} * 1.3 \mid \mathrm{CC} \rightarrow \mathrm{Control\ Cycle} > \frac{CC_{max}}{2}$$

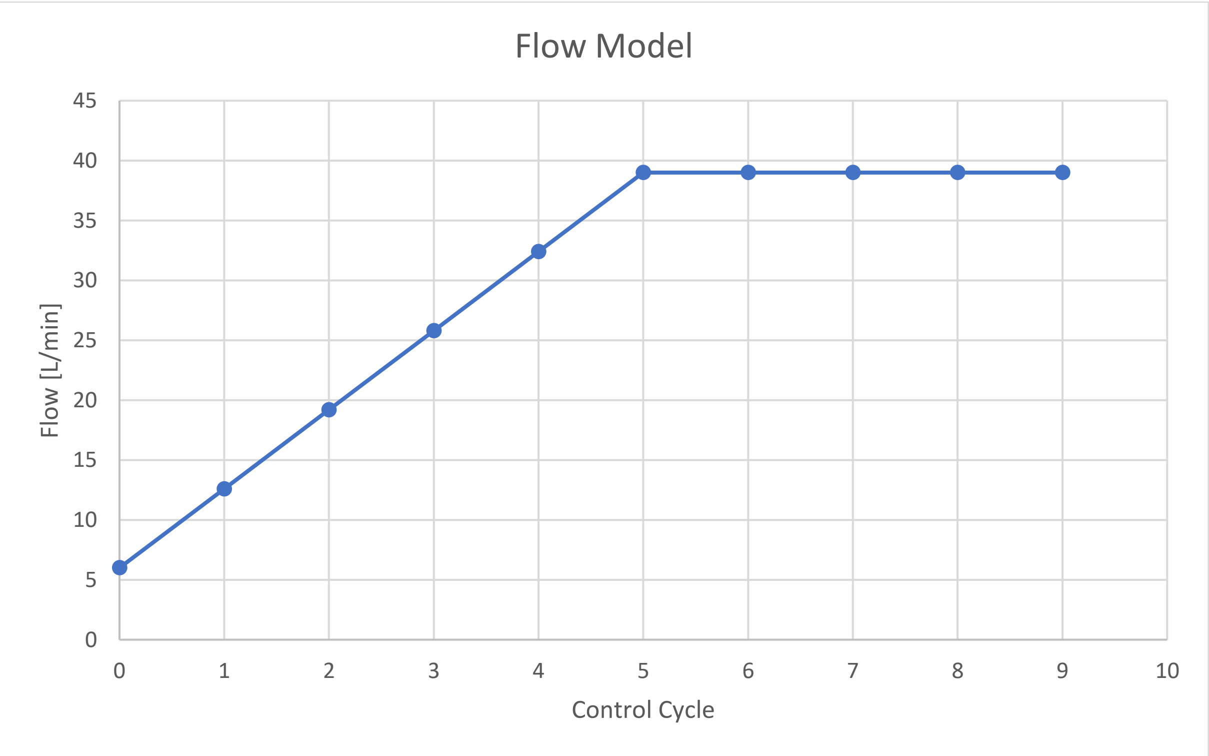

Figure 2 – Flow model

(f_{mean} = 30 L/min)

6 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.

7 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

8 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).

9 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.

9.1 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

9.2 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.