Algorithm 1: Read User Input

Result: Reads Buttons and Sets user specified parameters (every 50ms / 20Hz) (Parameters: Required Tidal Volume (ml/kg), Required Pressure(cmH2O), Required Respiratory Rate (BPM), Required FiO2, Ventilation Mode (Volume Controlled / Pressure Controlled), Patient Weight, PEEP, Expiratory Ratio)

```
//to be added in an always executing loop;
if sampling period has elapsed then
    read Ok Button Status;
    read Next/Previous Menu Item Button;
    read Value Increment Decrement Button;
    Set selected value;
    convert value to engineering unit and store;
    limit values to within max/min range
    save settings to EEPROM if Ok button is pressed
end
```

Algorithm 2: Read Sensors

Result: Reads and converts sensor values (every 50ms / 20Hz)

```
//to be added in an always executing loop;
if sampling period has elapsed then
    read pressure value;
    read flow value;
    read O2 Sensor value;
    filter all value with low pass filter;
    convert all values to engineering unit;
end
```

Algorithm 3: Volume Controlled Breath Cycle (Inspiratory Cycle) (For Stepper Motor)

Result: Sets the maximum speed and acceleration of stepper for inspiratory cycle (every 50ms / 20Hz)

//to be added in an always executing loop;

// /*********** MOTOR PARAMETERS **********

* These values will be highly dependent on mechanical design.

* Wrong values can cause unpredictables moves and motor stalls.

motorSpeed = Speed for 1 liter/second
motorAcceleration = Acceleration for 1 liter / second (inverse square of flow)
motorVolumeRatio = Ratio of distance in steps to air volume in step per mL.

$$* BreathPeriod(ms) = \frac{60000}{BPM}$$

breathPhase = 0;

if breathe period has elapsed **then**

breathPhase = 1;

calculate Inspiration cycle time; (needs improvement regarding snchronisation with patient's breathing cycle; if required)

$$InspirationTime(ms) = BreathPeriod * \frac{1}{1 + ReqExpRatio}$$

calculate Breathe In Speed (volume per millisecond);

$$BreatheInSpeed = \frac{Volume}{InspirationTime}$$

set maximum stepper speed as;

$$StepperMaxSpeed = motorSpeed * BreatheInSpeed$$

set maximum stepper acceleration as;

$$StepperAccel = motorAccelration * BreatheInSpeed^{2}$$

move stepper to desired position;

$$StepperMoveTo = motorVolumeRatio * ReqVolume$$

calculate tidal volume delivered;

$$TidalVolumeDelivered += MeasuredFlow * Timestep$$

calculate Minute Ventilation

$$MinuteVentilation += TidalVolumeDelivered$$

end

if oneMinuteCount elapsed **then**
calculate Total Minute Ventilation
$TotalMinuteVentilation = TidalVolumeDelivered$;
oneMinuteCount = 0;
end

Algorithm 4: Volume Controlled Breath Cycle (Expiratory Cycle) (For Stepper Motor)

Result: Sets the maximum speed and acceleration of stepper for Expiratory cycle (every 50ms / 20Hz)

//to be added in an always executing loop;

// /********** MOTOR PARAMETERS **********

* These values will be highly dependent on mechanical design.

* Wrong values can cause unpredictable moves and motor stalls.

motorSpeed = Speed for 1 liter/second
motorAcceleration = Acceleration for 1 liter / second (inverse square of flow)
motorVolumeRatio = Ratio of distance in steps to air volume in step per mL.

$$* BreathPeriod(ms) = \frac{60000}{BPM}$$

if InspirationTime has elapsed **then**

breathPhase = 2;

calculate Expiration cycle time;

$$ExpirationTime(ms) = BreathPeriod - InspirationTime$$

calculate Breathe Out Speed (volume per millisecond);

$$BreatheOutSpeed = \frac{Volume}{ExpirationTime}$$

set maximum stepper speed as;

$$StepperMaxSpeed = motorSpeed * BreatheOutSpeed$$

set maximum stepper acceleration as;

$$StepperAccel = motorAccelration * BreatheOutSpeed^{2}$$

move stepper to 0 position;

end

Algorithm 5: Alarm

Result: Sets the Alarm in case of abnormality

//to be added in an always executing loop;

//All Alarms are counter based, i.e. triggered after *XX* continuous samples

//Also Set corresponding Alarm Flag

if breathPhase = 1 **then**
  if Rel Pressure <= InspiratoryPressureLimit **then**
    $Alarm\ Beep = 1$;
  end
  if FiO2 <= FiO2Limit **then**
    $Alarm\ Beep = 1$;
  end
end
if breathPhase = 2 **then**
  if Rel Pressure < PEEP **then**
    $Alarm\ Beep = 1$;
  end
  if Rel Pressure >= ExpiratoryPressureLimit **then**
    $Alarm\ Beep = 1$;
  end
  if Rel Press < (0.1 *ReqdPress) **and** Peak Press < (0.1 *ReqdPress) **then**
    BeginMandatoryBreathingCycle = 1;
  end
  if TidalVolumeDelivered < (RequiredTidalVolume – Tolerance) **then**
    $Alarm\ Beep = 1$;
  end
  if TidalVolumeDelivered > (RequiredTidalVolume + Tolerance) **then**
    $Alarm\ Beep = 1$;
  end
end
if breathPhase = 0 **then**
  if Rel Pressure < PEEP **then**
    $Alarm\ Beep = 1$;
  end
  if Rel Pressure >= ExpiratoryPressureLimit **then**
    $Alarm\ Beep = 1$;
  end
  if Rel Press < (0.1 * ReqdPress) **and** Peak Press < (0.1 * ReqdPress) **then**
    BeginMandatoryBreathingCycle = 1;
  end
end
if TotalMinuteVentilation is outside ReqMinuteVentilation Bound **then**
  $Alarm\ Beep = 1$;
end

Algorithm 6: Oxygen Valve Control (FSO2)

Result: Controls the FSO2 (Level)

//to be added in an always executing loop;

if sampling period has elapsed **then**

calculate error value;

$$Error = ReqdFiO2 - MeasuredFiO2$$

$$Integral = Integral + Error$$

$$Derivative = Error - Pre\_Error$$

$$Pre\_Error = Error$$

$$\text{output} = (\text{kp} * \text{error}) + (\text{ki} * \text{integral}) + (\text{kd} * \text{derivative})$$

if output greater than zero **then**

open valve;

else if output less than zero **then**

close valve;

else

do nothing;

end

end

Algorithm 7: Load Saved Setting on Startup

Result: Load Saved Settings from EEPROM

//to be called once on startup
load all saved settings from eeprom;
perform limit check on all loaded values;
load default value in case limit check is failed;
end

Algorithm 8: Update Display

Result: Display Update

//Standard Update Function

End

Algorithm 9: Pressure Controlled Breath Cycle (Inspiratory Cycle) (For Stepper Motor)

Result: Sets the maximum speed and acceleration of stepper for inspiratory cycle (every 50ms / 20Hz)

//to be added in an always executing loop;

// /*********** MOTOR PARAMETERS **********

* These values will be highly dependent on mechanical design.

* Wrong values can cause unpredictable moves and motor stalls.

motorSpeedForPC = Speed for 1 kPa/second
motorAccelForPC = Acceleration for 1kPa / second
motorPressRatio = Ratio of distance in steps to air pressure in steps per kPa.

$$* BreathPeriod(ms) = \frac{60000}{BPM}$$

breathPhase = 0;

if breathe period has elapsed **then**

breathPhase = 1;

calculate Inspiration cycle time; (needs improvement regarding synchronization with patient's breathing cycle; if required)

$$InspirationTime(ms) = BreathPeriod * \frac{1}{1 + ReqExpRatio}$$

calculate Breathe-in Speed;

$$BreatheInSpeed = \frac{ReqdPress - (StepperCurrPos/motorPressRatio)}{InspirationTime}$$

set maximum stepper speed as;

$$StepperMaxSpeed = \text{motorSpeedForPC} * BreatheInSpeed$$

set maximum stepper acceleration as;

$$StepperAccel = motorAccelForPC * BreatheInSpeed^2$$

move stepper to desired position;

$$StepperMoveTo = motorPressRatio * ReqPressure$$

calculate tidal volume delivered;

$$TidalVolumeDelivered \mathrel{+}= MeasuredFlow * Timestep$$

calculate Minute Ventilation

$$MinuteVentilation \mathrel{+}= TidalVolumeDelivered$$

end

if oneMinuteCount elapsed **then**
calculate Total Minute Ventilation
$TotalMinuteVentilation = TidalVolumeDelivered$;
oneMinuteCount = 0;
end

Algorithm 10: Pressure Controlled Breath Cycle (Expiratory Cycle) (For Stepper Motor)

Result: Sets the maximum speed and acceleration of stepper for Expiratory cycle (every 50ms / 20Hz)

//to be added in an always executing loop;

// /*************** MOTOR PARAMETERS ***************

* These values will be highly dependent on mechanical design.

* Wrong values can cause unpredictable moves and motor stalls.

motorSpeedForPC = Speed for 1 kPa/second

motorAccelForPC = Acceleration for 1kPa / second

motorPressRatio = Ratio of distance in steps to air pressure in steps per kPa.

$$* BreathPeriod(ms) = \frac{60000}{BPM}$$

if InspirationTime has elapsed **then**

breathPhase = 2;

calculate Expiration cycle time;

$$ExpirationTime(ms) = BreathPeriod - InspirationTime$$

calculate Breathe Out Speed;

$$BreatheOutSpeed = \frac{ReqdPressure - PEEP}{ExpirationTime}$$

set maximum stepper speed as;

$$StepperMaxSpeed = \text{motorSpeedForPC} * BreatheOutSpeed$$

set maximum stepper acceleration as;

$$StepperMaxAccel = \text{motorAccelForPC} * BreatheOutSpeed^{2}$$

move stepper to PEEP position;

$$StepperMoveTo = motorPressRatio * PEEP$$

end

Algorithm 11: Self-Checks on Startup

Result: Perform Self-test and sensor calibration on startup.

//To be called once on startup
Wait for warmup time
Initialize input outputs
Check for default sensors values,
Reset valves to preset positions.
Do not start until start button is pressed.
Estimate default motor parameters of your design. (If Required)
end