PYQT ventilator project uses these following three classes:

1. **Breathworker** - This class is used for setting up the GPIO's, **Controlling the PWM and controlling the flow control** based on the given input parameters.

2. **Backendworker -** This class is used to get the live **Sensor data**.

3. **App -** This class is used for **UI**.

Important parameters:

- **Pressure –** pressure inside the lungs.
- **Volume –** Volume of air delivered to the lungs
- **PIP-** Highest pressure during the Inhalation.
- **Peep-** Pressure during the Exhalation
- **BPM-** Breath Per Minute
- **VTi-** Volume during the Inhalation
- **VTe-** Volume during the Exhalation
- **RR-**Respiratory Ratio
- **Trigger-** Patient triggering the breathe(During Inhalation)

Modes:

VC:

VC is abbreviated as Volume Control. In VC mode, the ventilator will be controlled based on the volume that was given from the input parameters. For example if we set 400 volume in the UI, the ventilator will send only 400 volume.

PC:

PC is abbreviated as Pressure Control. In PC mode, the ventilator will be controlled based on the pressure that was given from the input parameters. For example if we set 20 pressure in the UI, the ventilator will send only 20 pressure.

Spotaneous:

spontaneous mode works similar as the PC mode, this mode will also control the ventilator based on the pressure that was given from the input parameters. For the first 15 seconds the ventilator will be waiting for the patient to trigger the breathe, if the patient triggered the breathe, the pressure will reach to the given input pressure and after reaching the pressure, exhalation is started. If the patient doesn't triggered the breathe, automatically it changes to the PC mode.

HFONC:

In HFONC mode there will no exhalation and once this mode started, it will continuously gives pressure.

BreathWorker:

Update_pwm_data:

This function collects the input parameters data, that has been set by UI and process the data for calculation and calculate the "pressure cycle value".

Later this "pressure cycle value" was used to set duty cycle for controlling the flow control.

Pwm_in:

This function is used to control the flow control during the inhalation and this function behaves according to different modes

Pwm_out:

This function is used to control the flow control during the Exhalation and this function behaves according to different modes

Readpressurevalues:

This function read the values for duty cycle according to the pressure value from the input and this function read duty cycle value from the pressure.json

Readvolumevalues:

This function read the values for duty cycle according to the volume value from the input and this function read duty cycle value from the volume.json

Backendworker:

This class contains a dictionary “dataDict”, which is used update the live pressure and oxygen values to the UI.

***GetAndProcessMode*:**

This function is used to get the sensor data from the ADC and update the “dataDict” dictionary.

App:

createGridLayout:

This function creates the create the grid layout for all the buttons and labels in the UI

Comp_warning:

This function generates a Qmessage box is there is no compressor is connected to the ventilator. This function generates the compressor waring based on the global flag comp_on

Graph:

This function will generates the pyqt graph for pressure and start timer to update for the functions update_plot_data and alarm_data

Update_plot_data:

This function will update the labels and pressure graph based on the senor values. This function will receive the sensor value from the dataDict dictionary.

Alram_data:

This function will trigger the alarm based on the flags

Value_set:

This function will generate the slider, to set the values for pressure,volume,bpm,fi02.