Homtec

Generated by Doxygen 1.8.18

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 CiclosEPROM Struct Reference

esta estructura conserva los ciclos que lleva funcionando el respirador y la cantidad de escrituras en EPROM realizadas.

`#include <defaults.h>`

Public Attributes

- unsigned long **ciclos**
- unsigned int **escriturasEPROM**

3.1.1 Detailed Description

esta estructura conserva los ciclos que lleva funcionando el respirador y la cantidad de escrituras en EPROM realizadas.

The documentation for this struct was generated from the following file:

- defaults.h

3.2 COEFFS Struct Reference

Estructura con coeficientes para la calibración de la bolsa ambu de adulto.

`#include <defaults.h>`

3.2.1 Detailed Description

Estructura con coeficientes para la calibración de la bolsa ambu de adulto.

The documentation for this struct was generated from the following file:

- defaults.h

3.3 Configuration_t Struct Reference

Estructura de configuración.

`#include <MechVentilation.h>`

Public Attributes

- float **pip**
- unsigned short **timeoutIns**

3.3.1 Detailed Description

Estructura de configuración.

Parameters

pip	Peak Inspiratory Pressure timeoutIns: Tiempo total de inspiración
pip	

Peak Inspiratory Pressure

Parameters

timeoutIns	Tiempo total de inspiración

The documentation for this struct was generated from the following file:

- MechVentilation.h

3.4 Mech Class Reference

Contiene las variables y definiciones de los métodos para controlar la ventilación mecánica.

3.4.1 Detailed Description

Contiene las variables y definiciones de los métodos para controlar la ventilación mecánica.

The documentation for this class was generated from the following file:

- MechVentilation.h

3.5 MechVentilation Class Reference

Clase "mech Ventilation". Contiene las variables y definiciones de los métodos para controlar la ventilación mecánica.

`#include <MechVentilation.h>`

Public Member Functions

- MechVentilation (FlexyStepper *stepper, Sensors *sensors, AutoPID *pid, VentilationOptions_t options)

  Construct a new Mech Ventilation object.
- boolean **getStartWasTriggeredByPatient** ()
- void **setVentilationCyle_WaitTime** (float speedExsufflation)
- void start (void)
- void stopMech (void)
- uint8_t **getRunning** (void)
- void evaluatePressure (void)
- void **evaluate220V** (void)
- void update (void)
- void activateRecruitment (void)
- void **deactivateRecruitment** (void)
- int getAlarms (void)
- bool **getSensorErrorDetected** ()
- uint8_t **getRPM** (void)
- short **getExsuflationTime** (void)
- short **getInsuflationTime** (void)
- short **getPeakInspiratoryPressure** (void)
- short **getPeakEspiratoryPressure** (void)
- short **getVolTidalPorcen** (void)
- short **getVolTidalByArray** (void)
- short **getIer** (void)
- short **getTh** (void)
- uint8_t **getPause** (void)
- uint8_t **getFrReal** (void)
- State **getState** (void)
- void setAlarms (int alarm_value)
- void **resetAlarms** (int alarm_value)
- void setInsuflationTime (uint8_t value)

  Función de seteo de la variable _tHoldIn.
- float getInsuflationTimeApp (void)

  Función que permite ver el valor de la variable _timeoutIns para sensor de presion MPX.
- void **setRPM** (uint8_t rpm)
- void **setPeakInspiratoryPressure** (uint8_t pip)
- void **setPeakEspiratoryPressure** (uint8_t peep)
- void **setPause** (uint8_t pause)
- void **setVolTidalInStep** (uint8_t vt)
- void **setVolTidalByArray** (uint8_t vt)
- void **setIer** (uint8_t ier)
- void **setTh** (uint8_t th)
- void **goToPositionByDur** (FlexyStepper *stepper, int goal_pos, int cur_pos, int dur)
- void **setState** (State state)
- uint8_t **GetPlateau** (void)
- unsigned long **GetCiclos** (void)
- void **ResetCiclos** (void)

3.5.1 Detailed Description

Clase "mech Ventilation". Contiene las variables y definiciones de los métodos para controlar la ventilación mecánica.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 MechVentilation()

```
MechVentilation::MechVentilation (
            FlexyStepper * stepper,
            Sensors * sensors,
            AutoPID * pid,
            VentilationOptions_t options )
```

Construct a new Mech Ventilation object.

Parameters

stepper	
sensors	
pid	
options	

3.5.3 Member Function Documentation

3.5.3.1 activateRecruitment()

```
void MechVentilation::activateRecruitment (
            void  )
```

Recruitment

3.5.3.2 evaluatePressure()

```
void MechVentilation::evaluatePressure (
            void  )
```

Alarms

3.5.3.3 getAlarms()

```
int MechVentilation::getAlarms (
            void  )
```

getters

3.5.3.4 getInsuflationTimeApp()

```
float MechVentilation::getInsuflationTimeApp (
            void  )
```

Función que permite ver el valor de la variable _timeoutIns para sensor de presion MPX.

Parameters

ninguno	

Returns

valor de la variable _timeoutIns

permite leer el valor de la variable _timeoutIns

3.5.3.5 setAlarms()

```
void MechVentilation::setAlarms (
            int alarm_value )
```

setters

3.5.3.6 setInsuflationTime()

```
void MechVentilation::setInsuflationTime (
            uint8_t value )
```

Función de seteo de la variable _tHoldIn.

Parameters

value	valor a setear

Returns

ninguno

permite cambiar el valor de la variable _tHoldIn y recalcular los demas valores de tiempos

3.5.3.7 start()

```
void MechVentilation::start (
            void  )
```

Start mechanical ventilation.

3.5.3.8 stopMech()

```
void MechVentilation::stopMech (
            void  )
```

Stop mechanical ventilation.

3.5.3.9 update()

```
void MechVentilation::update (
            void  )
```

Update mechanical ventilation.

If any control variable were to change, new value would be applied at the beginning of the next ventilation cycle.

Note

This method must be called on a timer loop.

It's called from timer1Isr Alarma?????? <----------

The documentation for this class was generated from the following files:

- MechVentilation.h
- MechVentilation.cpp

3.6 MPX_PRESSURE_SENSOR Class Reference

Clase que almacena valores del sensosr de presión.

```
#include <mpx.h>
```

Public Member Functions

- float autoajuste (float offset)

  Función de autoajuste para sensor de presion MPX.
- void setOffsetValue (float value)

  Función de seteo de la variable offset para sensor de presion MPX.
- float getOffsetValue (void)

  Función que permite ver el valor de la variable offset para sensor de presion MPX.
- MPX_PRESSURE_SENSOR (uint8_t pin)

  Constructor del objeto sensor de presion MPX.
- void MPX_SET_ADC (void)

  Configura la tensión de referencia del ADC basado en el sensor definido por ENABLED_SENSOR_MPX5050.
- float MPX_READ_PRESSURE (void)

  Realiza la lectura de presión analogica.
- float MPX_CONVERT_PRESSURE (float Vfinal)

  Realiza conversion a KPa desde la tension leida previemente.
- float MPX_CONVERT_cmH20 (float pressureKPA)

  Realiza conversion del valor de presión en KPa a cmH20.
- ~MPX_PRESSURE_SENSOR ()

  Realiza una estimación del flujo basándose en mediciones de presión.

3.6.1 Detailed Description

Clase que almacena valores del sensosr de presión.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 MPX_PRESSURE_SENSOR()

```
MPX_PRESSURE_SENSOR::MPX_PRESSURE_SENSOR (
            uint8_t pin )
```

Constructor del objeto sensor de presion MPX.

MPX_PRESSURE_SENSOR(uint8_t pin)

Parameters

ninguno	

Returns

ninguno

3.6.2.2 ~MPX_PRESSURE_SENSOR()

```
MPX_PRESSURE_SENSOR::~MPX_PRESSURE_SENSOR ( )  [inline]
```

Realiza una estimación del flujo basándose en mediciones de presión.

float MPX_CONVERT_flujo();

Parameters

ninguno	

Returns

Valor de flujo estimado

Solo funciona si está habilitada la variable ENABLED_SENSOR_VOLUME_byPRESSURE ~MPX_PRESSURE_SENSOR()

Destructor

Parameters

void	

Returns

void

3.6.3 Member Function Documentation

3.6.3.1 autoajuste()

```
float MPX_PRESSURE_SENSOR::autoajuste (
            float offset )
```

Función de autoajuste para sensor de presion MPX.

float autoajuste (float offset)

Parameters

offset	valor de offset con el cual se compara

Returns

nuevo valor de offset ajustado

toma 10 medidas, realiza un promedio y compara el valor con el offset ingresado. En base a esa ccomparación realiza un ajuste del valor del offset para el sensor de presión.

3.6.3.2 getOffsetValue()

```
float MPX_PRESSURE_SENSOR::getOffsetValue (
            void  )
```

Función que permite ver el valor de la variable offset para sensor de presion MPX.

float getOffsetValue(void)

Parameters

ninguno	

Returns

valor de la variable de offset

3.6.3.3 MPX_CONVERT_cmH20()

```
float MPX_PRESSURE_SENSOR::MPX_CONVERT_cmH20 (
            float pressureKPA )
```

Realiza conversion del valor de presión en KPa a cmH20.

MPX_CONVERT_cmH20(float pressureKPA)

Parameters

Valor	de presión en KPa

Returns

Valor en float que representa la presion en cmH2O

3.6.3.4 MPX_CONVERT_PRESSURE()

```
float MPX_PRESSURE_SENSOR::MPX_CONVERT_PRESSURE (
            float Vfinal )
```

Realiza conversion a KPa desde la tension leida previemente.

float MPX_CONVERT_PRESSURE(float Vfinal)

Parameters

Tensión	devuelta por MPX_PRESSURE_SENSOR::MPX_READ_PRESSURE

Returns

Valor que representa la presion leída en KPa

Dependiendo de la definición de ENABLED_SENSOR_MPX5050 mapea en los valores correcto. MPX 5010 Vout = VS x (0.09 x P + 0.04) = 0.45 x P + 0.2 P = (Vout - 0.2)/0.45 = Vfinal /0.45 MPX 5050 Vout = VS (P x 0.018 + 0.04) P = Vfinal / 0.09

3.6.3.5 MPX_READ_PRESSURE()

```
float MPX_PRESSURE_SENSOR::MPX_READ_PRESSURE (
            void  )
```

Realiza la lectura de presión analogica.

float MPX_READ_PRESSURE(void)

Parameters

ninguno	

Returns

Valor que representa el voltaje de la presion sin offset

Dependiendo de la definición de ENABLED_SENSOR_MPX5050 mapea la lectura del pin PIN_MPX_DATA en los valores correcto

3.6.3.6 MPX_SET_ADC()

```
void MPX_PRESSURE_SENSOR::MPX_SET_ADC (
            void  )
```

Configura la tensión de referencia del ADC basado en el sensor definido por ENABLED_SENSOR_MPX5050.

void MPX_SET_ADC(void)

Parameters

ninguno	

Returns

ninguno

Si ENABLED_SENSOR_MPX5050 = 1 la tensión de refencia es 1.1V, caso contrario la tensión será 2.56V

3.6.3.7 setOffsetValue()

```
void MPX_PRESSURE_SENSOR::setOffsetValue (
            float value )
```

Función de seteo de la variable offset para sensor de presion MPX.

void setOffsetValue(float value)

Parameters

value	valor a setear

Returns

ninguno

The documentation for this class was generated from the following files:

- mpx.h
- mpx.cpp

3.7 Sensores Class Reference

Contiene las variables y definiciones de los métodos para utilizar los sensores de presión y volumen.

`#include <Sensors.h>`

3.7.1 Detailed Description

Contiene las variables y definiciones de los métodos para utilizar los sensores de presión y volumen.

The documentation for this class was generated from the following file:

- Sensors.h

3.8 SensorLastPressure_t Struct Reference

Estructura que almacena las últimas lecturas del sensor de presión (mínima y máxima)

`#include <Sensors.h>`

Public Attributes

- uint8_t **minPressure**
- uint8_t **maxPressure**

3.8.1 Detailed Description

Estructura que almacena las últimas lecturas del sensor de presión (mínima y máxima)

Estructura que contiene información sobre el sensor de de presión.

The documentation for this struct was generated from the following file:

- Sensors.h

3.9 SensorPressureValues_t Struct Reference

Public Attributes

- SensorState **state**
- float **pressure1**
- float **pressure2**

The documentation for this struct was generated from the following file:

- Sensors.h

3.10 Sensors Class Reference

Clase "Sensores". Contiene las variables y definiciones de los métodos para utilizar los sensores de presión y volumen.

`#include <Sensors.h>`

Public Member Functions

- Sensors ()

  Funcion que comienza el proceso de inicialización de la clase sensores.
- unsigned int begin (void)

  Función Dummy.
- float readPressure (void)

  Funcion que lee el sensor de presión MPX 50XX.
- void rutinaDeAutoajuste (void)

  Función para la autocalibración del sensor de presión. Permite calibrar el valor de offset al iniciar el sistema.
- SensorLastPressure_t getLastPressure (void)

  Función que permite obtener la última presión leida.
- SensorPressureValues_t getAbsolutePressureInPascals (void)

  Función que permite obtener el valor absoluto de presión en Pascales.
- SensorPressureValues_t getAbsolutePressureInCmH2O (void)

  Función que permite obtener el valor absoluto de presión en CmH20.
- SensorPressureValues_t getRelativePressureInCmH2O (void)

  Función que permite obtener el valor relativo de presión en CmH20.

- void resetPressures (void)

  Función que permite resetear los valores de presión de la estructura SensorLastPressure_t.
- void rebootVolumeSensor (void)

  Funcion que reestablece el valor del integrador para el sensor de volumen.
- void readVolume (FlexyStepper ∗stepper)

  Función que lee volumen y lo guarda en la estructura correspondiente.
- void **resetVolumeIntegrator** (void)
- float getFlow (void)

  Función que permite obtener una medida de flujo.
- float getVolumeActual (void)

  Función que permite obtener el valor de volumen almacenado en la estructura Sensors.
- SensorVolumeValue_t getVolume (void)

  Función que accede a los datos almacenados en la estructura SensorVolumeValue_t.
- void saveVolume (void)

  Función que guarda el volumen leído como última medida tomada en la estructura correspondiente.
- bool stalledVolumeSensor ()

  Función obsoleta. No usar, no borrar.

3.10.1 Detailed Description

Clase "Sensores". Contiene las variables y definiciones de los métodos para utilizar los sensores de presión y volumen.

Función Dummy

Parameters

ninguno	

Returns

0

3.10.2 Constructor & Destructor Documentation

3.10.2.1 Sensors()

```
Sensors::Sensors ( )
```

Funcion que comienza el proceso de inicialización de la clase sensores.

Parameters

ninguno	

Returns

ninguno

3.10.3 Member Function Documentation

3.10.3.1 begin()

```
unsigned int Sensors::begin (
            void  )
```

Función Dummy.

Parameters

ninguno	

Returns

0

3.10.3.2 getAbsolutePressureInCmH2O()

```
SensorPressureValues_t Sensors::getAbsolutePressureInCmH2O (
            void  )
```

Función que permite obtener el valor absoluto de presión en CmH20.

Parameters

ninguno	

Returns

SensorValues_t - values: Valor de presión obtenido convertido a CmH20

3.10.3.3 getAbsolutePressureInPascals()

```
SensorPressureValues_t Sensors::getAbsolutePressureInPascals (
            void  )
```

Función que permite obtener el valor absoluto de presión en Pascales.

Parameters

ninguno	

Returns

SensorValues_t - values: Valor de presión obtenido convertido a Pascales

3.10.3.4 getFlow()

```
float Sensors::getFlow (
            void  )
```

Función que permite obtener una medida de flujo.

Parameters

ninguno	

Returns

float flow: valor de flujo obtenido

3.10.3.5 getLastPressure()

```
SensorLastPressure_t Sensors::getLastPressure (
            void  )
```

Función que permite obtener la última presión leida.

Parameters

ninguno	

Returns

SensorLastPressure_t - lastPres: Último valor de presión registrado

3.10.3.6 getRelativePressureInCmH2O()

```
SensorPressureValues_t Sensors::getRelativePressureInCmH2O (
            void  )
```

Función que permite obtener el valor relativo de presión en CmH20.

Parameters

ninguno	

Returns

SensorValues_t - values: Valor de presión obtenido convertido a CmH20

3.10.3.7 getVolume()

```
SensorVolumeValue_t Sensors::getVolume (
            void  )
```

Función que accede a los datos almacenados en la estructura SensorVolumeValue_t.

Parameters

ninguno	

Returns

_volumeState

_lastVolume

3.10.3.8 getVolumeActual()

```
float Sensors::getVolumeActual (
            void  )
```

Función que permite obtener el valor de volumen almacenado en la estructura Sensors.

Parameters

ninguno	

Returns

float _volume_ml

3.10.3.9 readPressure()

```
float Sensors::readPressure (
            void  )
```

Funcion que lee el sensor de presión MPX 50XX.

Parameters

ninguno	

Returns

ninguno

3.10.3.10 readVolume()

```
void Sensors::readVolume (
            FlexyStepper * stepper )
```

Función que lee volumen y lo guarda en la estructura correspondiente.

Parameters

ninguno	

Returns

ninguno

3.10.3.11 rebootVolumeSensor()

```
void Sensors::rebootVolumeSensor (
            void  )
```

Funcion que reestablece el valor del integrador para el sensor de volumen.

Parameters

ninguno	

Returns

ninguno

3.10.3.12 resetPressures()

```
void Sensors::resetPressures (
            void  )
```

Función que permite resetear los valores de presión de la estructura SensorLastPressure_t.

Parameters

ninguno	

Returns

ninguno

3.10.3.13 rutinaDeAutoajuste()

```
void Sensors::rutinaDeAutoajuste (
            void  )
```

Función para la autocalibración del sensor de presión. Permite calibrar el valor de offset al iniciar el sistema.

Parameters

ninguno	

Returns

ninguno

3.10.3.14 saveVolume()

```
void Sensors::saveVolume (
            void  )
```

Función que guarda el volumen leído como última medida tomada en la estructura correspondiente.

Parameters

ninguno	

Returns

ninguno

3.10.3.15 stalledVolumeSensor()

```
bool Sensors::stalledVolumeSensor ( )
```

Función obsoleta. No usar, no borrar.

Parameters

ninguno	

Returns

Bool

The documentation for this class was generated from the following files:

- Sensors.h
- Sensors.cpp

3.11 SensorVolumeValue_t Struct Reference

Estructura que contiene información sobre el sensor de volumen.

```
#include <Sensors.h>
```

Public Attributes

- SensorState **state**
- short **volume**

3.11.1 Detailed Description

Estructura que contiene información sobre el sensor de volumen.

The documentation for this struct was generated from the following file:

- Sensors.h

3.12 VentilationOptions_t Struct Reference

Public Attributes

- short **height**
- bool **sex**
- short **respiratoryRate**
- short **peakInspiratoryPressure**
- short **peakEspiratoryPressure**
- float **triggerThreshold**
- bool **hasTrigger**

The documentation for this struct was generated from the following file:

- defaults.h

Chapter 4

File Documentation

4.1 calc.cpp File Reference

```
#include "calc.h"
```

Macros

- #define **ELEM_SWAP**(a, b) { unsigned int t=(a);(a)=(b);(b)=t; }

Functions

- int estimateTidalVolume (int estatura, int sexo)
  estima el volumen tidal en función de estatura y sexo, en ml.
- unsigned int QuickSelectMedian (unsigned int arr[ ], uint8_t n)
  filtro de media movil.
- void refreshWatchDogTimer ()
  Refresca el WDT (Watch Dog Timer)

4.1.1 Detailed Description

Funciones de cálculo necesarias

Este archivo contiene las definiciones de las funciones de cálculo necesarias

4.1.2 Function Documentation

4.1.2.1 estimateTidalVolume()

```
int estimateTidalVolume (
            int estatura,
            int sexo )
```

estima el volumen tidal en función de estatura y sexo, en ml.

Parameters

estatura	en cm, del paciente
sexo	0: varón, 1: mujer, sexo del paciente

Returns

∗volumenTidal volumen tidal estimado, en mililitros

4.1.2.2 QuickSelectMedian()

```
int QuickSelectMedian (
            unsigned int arr[],
            uint8_t n )
```

filtro de media movil.

Parameters

estatura	en cm, del paciente
sexo	0: varón, 1: mujer, sexo del paciente

Returns

señal filtrada.

4.1.2.3 refreshWatchDogTimer()

```
void refreshWatchDogTimer ( )
```

Refresca el WDT (Watch Dog Timer)

Parameters

ninguno	\returm ninguno

4.2 calc.h File Reference

```
#include "defaults.h"
#include "Arduino.h"
#include <avr/wdt.h>
```

Functions

- int estimateTidalVolume (int estatura, int sexo)

  estima el volumen tidal en función de estatura y sexo, en ml.
- unsigned int QuickSelectMedian (unsigned int arr[ ], uint8_t n)

  filtro de media movil.
- void refreshWatchDogTimer ()

  Refresca el WDT (Watch Dog Timer)

4.2.1 Detailed Description

Funciones de cálculo necesarias

Este archivo contiene las declaraciones de las funciones de cálculo necesarias.

4.2.2 Function Documentation

4.2.2.1 estimateTidalVolume()

```
int estimateTidalVolume (
            int estatura,
            int sexo )
```

estima el volumen tidal en función de estatura y sexo, en ml.

Parameters

estatura	en cm, del paciente
sexo	0: varón, 1: mujer, sexo del paciente

Returns

∗volumenTidal volumen tidal estimado, en mililitros

4.2.2.2 QuickSelectMedian()

```
unsigned int QuickSelectMedian (
            unsigned int arr[],
            uint8_t n )
```

filtro de media movil.

Parameters

estatura	en cm, del paciente
sexo	0: varón, 1: mujer, sexo del paciente

Returns

señal filtrada.

4.2.2.3 refreshWatchDogTimer()

```
void refreshWatchDogTimer ( )
```

Refresca el WDT (Watch Dog Timer)

Parameters

ninguno	\return ninguno

4.3 defaults.h File Reference

Classes

- struct CiclosEPROM

  esta estructura conserva los ciclos que lleva funcionando el respirador y la cantidad de escrituras en EPROM realizadas.
- struct VentilationOptions_t

Macros

- #define **DEBUG_UPDATE** 0
- #define **DEBUG_PLOT** 0
- #define **TIME_BASE** 20
- #define **TIME_SENSOR** 100
- #define **TIME_EPROM** 6000
- #define **TIME_SEND_CONFIGURATION** 2000
- #define **ENABLED_SENSOR_VOLUME** 1
- #define **ENABLED_SENSOR_VOLUME_SFM3300** 0
- #define **ENABLED_SENSOR_VOLUME_STEPPER** 1
- #define **ENABLED_SENSOR_VOLUME_byPRESSURE** 0
- #define AMBU_MAX_ML 470.0F

  Volumen máximo en ml de la bolsa ambu para dultos.
- #define STEPPER_MICROSTEPS 4

  Micropasos del motor paso a paso.
- #define STEPPER_STEPS_PER_REVOLUTION 200

Cantidad de pasos por revolución del motor paso a paso.

- #define STEPPER_MICROSTEPS_PER_REVOLUTION (STEPPER_STEPS_PER_REVOLUTION ∗ STEPPER_MICROSTEPS)

  Cantidad de micropasos por revolución del motor. Se calcula como STEPPER_MICROSTEPS_PER_REVOLUTION=STEPPER_STEPS_PER_REVOLUTION ∗ STEPPER_MICROSTEPS.
- #define STEPPER_DIR -1

  dirección de movimiento del motor paso a paso.
- #define STEPPER_HOMING_DIRECTION (1)

  dirección de movimiento del motor paso a paso durante la rutina de posicionamiento inicial.
- #define STEPPER_HOMING_SPEED (STEPPER_MICROSTEPS ∗ 300)

  velocidad de movimiento del motor paso a paso durante la rutina de posicionamiento inicial.
- #define STEPPER_LOWEST_POSITION (STEPPER_DIR ∗ 2)

  mínima posición en pasos que puede alcanzar el eje motor. Físicamente se traduce en la bolsa ambu sin presionar
- #define STEPPER_HIGHEST_POSITION (STEPPER_DIR ∗ STEPPER_MICROSTEPS ∗ 181)

  máxima posición en pasos que puede alcanzar el eje motor. Físicamente se traduce en la bolsa ambu totalmente presionada.
- #define STEPPER_SPEED_DEFAULT 2000

  velocidad por defecto del movimiento del motor paso a paso.
- #define STEPPER_SPEED_MAX 4000

  Máxima velocidad del movimiento del motor paso a paso. Limite superior.
- #define STEPPER_ACC_MAX 9300

  Máxima aceleración del movimiento del motor paso a paso. Limite superior.
- #define STEPPER_SPEED_MAX_STOP 1900

  Máxima velocidad del movimiento del motor paso a paso cuando se lo quiere parar.
- #define STEPPER_ACC_MAX_STOP 9300

  Máxima aceleración del movimiento del motor paso a paso cuando se lo quiere parar.
- #define STEPPER_ACC_EXSUFFLATION (STEPPER_MICROSTEPS ∗ 600)

  Aceleración del movimiento del motor paso a paso durante la exuflación.
- #define STEPPER_ACC_INSUFFLATION (STEPPER_MICROSTEPS ∗ 450)

  Aceleración del movimiento del motor paso a paso durante la insuflación.
- #define DEFAULT_HEIGHT 170

  Altura del paciente por defecto.
- #define DEFAULT_SEX 0

  Sexo del paciente por defecto, siendo 0=varón, 1=mujer.
- #define DEFAULT_ML_PER_KG_IDEAL_WEIGHT 7

  Cantidad de mililitros de aire por Kg, se considera el peso ideal.
- #define DEFAULT_MAX_TIDAL_VOLUME 186

  Volumen tidal máximo por defecto.
- #define DEFAULT_MIN_TIDAL_VOLUME 0

  Volumen tidal mínimo por defecto.
- #define DEFAULT_DIFF_TIDAL_VOLUME 193

  Variable que define la excursión máxima que puede tener el motor entre el máximo y el mínimo volumen tidal insuflado.
- #define DEFAULT_TRIGGER_THRESHOLD 0

  Umbral de disparo por defecto. Se utiliza para detectar la intención de respiración por parte del paciente.
- #define DEFAULT_RPM 10

  Revoluciones por minuto por defecto del motor.
- #define DEFAULT_MAX_RPM 30

  Valor máximo de RPM del motor.
- #define DEFAULT_MIN_RPM 10

  Valor mínimo de RPM del motor.

- #define DEFAULT_INSPIRATORY_FRACTION 0.5F

  Valor por defecto del I/E ratio.
- #define **DEFAULT_PEAK_INSPIRATORY_PRESSURE** 35
- #define DEFAULT_PEAK_INSPIRATORY_PRESSURE_MIN 5

  Valor mínimo de presión inspiratoria por defecto.
- #define DEFAULT_PEAK_INSPIRATORY_PRESSURE_MAX 39

  Valor máximo de presión inspiratoria por defecto.
- #define DEFAULT_PEAK_ESPIRATORY_PRESSURE 10

  Valor deseado de presión espiratoria por defecto.
- #define DEFAULT_PEAK_ESPIRATORY_PRESSURE_MIN 5

  Valor mínimo de presión espiratoria por defecto.
- #define DEFAULT_PEAK_ESPIRATORY_PRESSURE_MAX 20

  Valor máximo de presión espiratoria por defecto.
- #define STEPPER_PEEP_SOLENOID_HYSTERESIS 0.8F

  Valor de presión espiratoria por defecto en cmH2O para implementar un ciclo de histéresis con los solenoides.
- #define DEFAULT_PORC_VOLTILDAL 70

  Porcentaje de volumen tidal insuflado por defecto.
- #define DEFAULT_BYARRAY_VOLTILDAL 4

  indice del arreglo volumen tidal
- #define **DEFAULT_PAUSE** 1
- #define DEFAULT_TINS 200

  Tiempo de insuflación por defecto.
- #define DEFAULT_PA_TO_CM_H2O 0.0102F

  Constante de conversión de pascales a cmH2O.
- #define DEFAULT_RECRUITMENT_TIMEOUT 20000

  Tiempo de reclutamiento en ms por defecto.
- #define **DEFAULT_RECRUITMENT_PIP** 20
- #define **VALVE_MAX_PRESSURE** 41
- #define **PID_MIN** STEPPER_LOWEST_POSITION
- #define **PID_MAX** STEPPER_HIGHEST_POSITION
- #define **PID_KP** 40
- #define **PID_KI** 0
- #define **PID_KD** 0
- #define **PID_TS** TIME_BASE
- #define **PID_BANGBANG** 8
- #define SOLENOID_CLOSED 0
- #define **SOLENOID_OPEN** 1
- #define **PIN_STEPPER_STEP** 26
- #define **PIN_STEPPER_DIRECTION** 28
- #define **PIN_STEPPER_EN** 24
- #define **PIN_BUZZ** 11
- #define **PIN_STEPPER_ENDSTOP** 3
- #define **PIN_SOLENOID1** 9
- #define **PIN_SOLENOID2** 10
- #define **PIN_TEST220V** A3
- #define ALARM_MAX_PRESSURE 38

  Máxima presión en cmH2O permitida.
- #define **ALARM_MIN_PRESSURE** 3
- #define No_Alarm 0x0000
- #define **Alarm_desconected_EN** 0x0101
- #define Alarm_desconected_DIS 0xFEFF
- #define **Alarm_Overpressure_EN** 0x0202

- #define Alarm_Overpressure_DIS 0xFDFF
- #define **Alarm_breathe_EN** 0x0404
- #define Alarm_breathe_DIS 0xFBFF
- #define Alarm_No_Flux_EN 0x0808
- #define Alarm_No_Flux_DIS 0xF7FF
- #define **Alarm_Libre_1_EN** 0x1010
- #define **Alarm_Libre_1_DIS** 0xEFFF
- #define **Alarm_Libre_2_EN** 0x2020
- #define **Alarm_Libre_2_DIS** 0xDFFF
- #define **Alarm_Libre_3_EN** 0x4040
- #define **Alarm_Libre_3_DIS** 0xBFFF
- #define **Alarm_Libre_4_EN** 0x8080
- #define **Alarm_Libre_4_DIS** 0X7FFF
- #define PIN_MPX_DATA A5

  Pin de datos del sensor de presión MPX.
- #define PIN_MPX_FLOW A9

  Pin de datos del sensor de flujo MPX.
- #define **ENABLED_SENSOR_MPX** 1
- #define DEFAULT_OFFSET_MPX 185.0F

  valor de desviación por defecto. Utilizada para corregir la desviación inicial del sensor.
- #define **DEFAULT_OFFSET_MPX2** 210.0F
- #define **ENABLED_SENSOR_MPX5050** 1

Variables

- struct {
  float **a**
  float **b**
  float **c**
  } **COEFFS**

4.3.1 Detailed Description

Valores por defecto del sistema.

Este archivo contiene todos los valores por defecto que emplea el sistema

4.3.2 Macro Definition Documentation

4.3.2.1 Alarm_breathe_DIS

```
#define Alarm_breathe_DIS 0xFBFF
```

Alarma de falta de flujo

4.3.2.2 Alarm_desconected_DIS

```
#define Alarm_desconected_DIS 0xFEFF
```

Alarma de sobrepresión

4.3.2.3 Alarm_No_Flux_DIS

```
#define Alarm_No_Flux_DIS 0xF7FF
```

Códigos libres para futuras implementaciones de alarmas

4.3.2.4 Alarm_No_Flux_EN

```
#define Alarm_No_Flux_EN 0x0808
```

4.3.2.5 Alarm_Overpressure_DIS

```
#define Alarm_Overpressure_DIS 0xFDFF
```

Alarma de intención de respiración

4.3.2.6 DEFAULT_BYARRAY_VOLTILDAL

```
#define DEFAULT_BYARRAY_VOLTILDAL 4
```

indice del arreglo volumen tidal

4.3.2.7 DEFAULT_PA_TO_CM_H2O

```
#define DEFAULT_PA_TO_CM_H2O 0.0102F
```

Constante de conversión de pascales a cmH2O.

4.3.2.8 DEFAULT_TINS

```
#define DEFAULT_TINS 200
```

Tiempo de insuflación por defecto.

4.3.2.9 No_Alarm

```
#define No_Alarm 0x0000
```

Alarma de sensor desconectado

4.3.2.10 PIN_MPX_DATA

```
#define PIN_MPX_DATA A5
```

Pin de datos del sensor de presión MPX.

Pinout para sensor MPX 5050/5010

4.3.2.11 PIN_MPX_FLOW

```
#define PIN_MPX_FLOW A9
```

Pin de datos del sensor de flujo MPX.

4.3.2.12 SOLENOID_CLOSED

```
#define SOLENOID_CLOSED 0
```

variables de control de los solenoides

4.3.2.13 STEPPER_HIGHEST_POSITION

```
#define STEPPER_HIGHEST_POSITION (STEPPER_DIR * STEPPER_MICROSTEPS * 181)
```

máxima posición en pasos que puede alcanzar el eje motor. Físicamente se traduce en la bolsa ambu totalmente presionada.

4.3.2.14 STEPPER_LOWEST_POSITION

```
#define STEPPER_LOWEST_POSITION (STEPPER_DIR * 2)
```

mínima posición en pasos que puede alcanzar el eje motor. Físicamente se traduce en la bolsa ambu sin presionar

4.3.2.15 STEPPER_MICROSTEPS

```
#define STEPPER_MICROSTEPS 4
```

Micropasos del motor paso a paso.

Stepper defines

4.3.2.16 STEPPER_PEEP_SOLENOID_HYSTERESIS

```
#define STEPPER_PEEP_SOLENOID_HYSTERESIS 0.8F
```

Valor de presión espiratoria por defecto en cmH2O para implementar un ciclo de histéresis con los solenoides.

4.4 MechVentilation.cpp File Reference

```
#include "MechVentilation.h"
```

Variables

- int **currentWaitTriggerTime** = 0
- int **currentStopInsufflationTime** = 0
- float **currentFlow** = 0

4.4.1 Detailed Description

Ventilación mecánica

Este archivo contiene las definiciones de los métodos para controlar la ventilación mecánica a través de todos sus componentes.

4.5 MechVentilation.h File Reference

```
#include "mpx.h"
#include <float.h>
#include <inttypes.h>
#include "defaults.h"
#include "calc.h"
#include "Sensors.h"
#include "src/AutoPID/AutoPID.h"
#include "src/FlexyStepper/FlexyStepper.h"
```

Classes

- struct Configuration_t

  Estructura de configuración.
- class MechVentilation

  Clase "mech Ventilation". Contiene las variables y definiciones de los métodos para controlar la ventilación mecánica.

Enumerations

- enum State {
  Init_Insufflation = 1, State_Insufflation = 2, **Init_Exsufflation** = 3, State_Exsufflation = 4,
  State_Hold_In = 5, **State_Error_Pas** = 6, **State_StandBy** = 7, **State_Homing** = 0,
  State_Error = -1 }

  Enumneración de los estados de la máquina de estados utilizada para el control de la ventilación mecánica.

Variables

- const float HOLD_IN_DURATION = 30.0

  Seteos para la temporización.
- const float MIN_PEEP_PAUSE =50.0

  Duracion en ms de la pausa luego de la exsuflación. Durante este tiempo se espera por una posible reacción del paciente.
- const float MAX_EX_DURATION = 1000.0

  duración máxima de exalación en ms
- const uint8_t VoluTidal [20]

  Arreglo donde se guarda el volumen tidal.

4.5.1 Detailed Description

Ventilación mecánica

Este archivo contiene las declaraciones de los métodos para controlar la ventilación mecánica a través de todos sus componentes.

4.5.2 Enumeration Type Documentation

4.5.2.1 State

```
enum State
```

Enumneración de los estados de la máquina de estados utilizada para el control de la ventilación mecánica.

Init_Insufflation Comienza la insuflación State_Insufflation Se insufla durante todo el tiempo de inspiración Init_↩ Exsufflation Comienza la exuflación
State_Exsufflation Comienza la exsuflación State_Hold_In Tiempo de esperaantes de la exsuflación
State_Error_Pas Estado de error
State_StandBy Estado de espera. Respirador apagado State_Homing Estado donde se realiza lacalibración del cero del motor paso a paso State_Error Estado de error

Seteos para la temporización

Enumerator

State_Insufflation	Insufflating (PID control).
State_Exsufflation	Return to position 0 and wait for the patient to exsufflate.

4.5.3 Variable Documentation

4.5.3.1 HOLD_IN_DURATION

```
HOLD_IN_DURATION = 30.0
```

Seteos para la temporización.

Duracionde la pausa en ms luego de la insuflación.

4.5.3.2 VoluTidal

```
VoluTidal
```

Initial value:

```
= {91, 96, 101, 105,
                                110, 115, 119, 124,
                                129, 133, 138, 143,
                                148, 153, 158, 163,
                                168, 173, 178, 186}
```

Arreglo donde se guarda el volumen tidal.

4.6 mpx.cpp File Reference

```
#include "mpx.h"
#include "calc.h"
```

4.6.1 Detailed Description

Este archivo contiene las definiciones de las funciones utilizadas para leer los sensores MPX 50XX
así como tambien las funciones necesarias para convertir los valores leidos a las diversas unidades requeridas.

4.7 mpx.h File Reference

```
#include "defaults.h"
#include "Arduino.h"
```

Classes

- class MPX_PRESSURE_SENSOR

  Clase que almacena valores del sensosr de presión.

4.7.1 Detailed Description

Este archivo contiene las declaraciones de las funciones utilizadas para leer los sensores MPX 50XX
así como tambien las estructureas de datos de dichos sensores.

4.8 respi_Esp_Mit.ino File Reference

```
#include "defaults.h"
#include "calc.h"
#include "Sensors.h"
#include <EEPROM.h>
#include "MechVentilation.h"
#include "mpx.h"
#include "src/AutoPID/AutoPID.h"
#include "src/FlexyStepper/FlexyStepper.h"
#include "src/TimerOne/TimerOne.h"
#include "src/TimerThree/TimerThree.h"
#include <avr/wdt.h>
```

Functions

- void readIncomingMsg (void)

  Esta función es la encargada de implementar el protocolo de comunicación con la interfaz gráfica.
- void setup ()

  Función de inicio principal del microcontrolador.
- void loop ()

  Función de bocle principal del microcontrolador.
- void **timer1Isr** (void)
- void **timer3Isr** (void)

Variables

- FlexyStepper ∗ stepper = new FlexyStepper()

  Puntero a clase que maneja motor paso a paso.
- Sensors ∗ sensors

  Puntero a clase que maneja los sensores de presion, flujo, y otros eventualmente.
- AutoPID ∗ **pid**
- MechVentilation ∗ ventilation

  Puntero a clase que maneja la maquina de estado principal del proceso respiratorio y del respirador en si mismo.
- VentilationOptions_t options

  Clase que configura las opciones por default de microcontrolador.
- volatile uint16_t dirEPROM

  Direcció EEPROM donde se guarda la dirección de la estructura CiclosEPROM.
- CiclosEPROM st_Ciclos

  Estructura que contiene la cantidad de ciclos y la cantidad de grabaciones en EEPROM.
- uint8_t pim

  Variable de configuración de Presión inspiratoria máxima.
- uint8_t peep

  Variable de configuración de Presión positiva al final de la expiración (en desuso).
- uint8_t frecResp

  Variable de configuración de Frecuencia respitatoria.
- uint8_t ier

  Variable de configuración de Relación inspiración expiración.
- uint8_t vt

  Variable de configuración de Volumen Tidal.
- uint8_t thr

  Variable de configuración de Umbral de detección de inspiración.
- uint8_t pause

  Variable de configuración de Pausa respiratoria.
- uint8_t encend

  Variable de configuración de Encendido-Apagado de respirador.
- uint8_t tins

  Variable de configuración de Tiempo de inspiración.

4.8.1 Detailed Description

Author

HOMTEC Team

Version

1.0

Date

2020-08-26

Copyright

GPLv3

4.8.2 Function Documentation

4.8.2.1 loop()

```
loop ( )
```

Función de bocle principal del microcontrolador.

A través de valores temporizado de menor prioridad que los timers, realiza actualización grafica y de valores medidos a la interfaz cada 100ms. Además cada 10 minutos actualiza los ciclos de la maquina.

Parameters

ninguno	

Returns

ninguno

4.8.2.2 readIncomingMsg()

```
void readIncomingMsg (
            void  )
```

Esta función es la encargada de implementar el protocolo de comunicación con la interfaz gráfica.

Permite configurar valores tipicos en los procesos respiratorio y realiza una retransmisión de hacia la interfaz como chequeo de parámetros. Formato: "CONFIG %d %d %d %d %d %d %d %d %d#"

Parameters

ninguno	

Returns

ninguno

4.8.2.3 setup()

```
setup ( )
```

Función de inicio principal del microcontrolador.

Configura: los baudios de los dos puertos series de comnunicación, pines digitales de entreda-salida, inicializa sensores, PID (obsoleto), la máquina de estado MechVentilation y sus opciones. Maneja valores grabados en E↩ EPROM y determina el inicio normal con calibración y valores por defectos, o el inicio rápido de emergencia con valores previos seteados. Actualiza valores para la interfaz. Configura temporizadores a 40us y 20ms para el motor paso a paso y para la maquina de estado, respectivamente.

Parameters

ninguno	

Returns

ninguno

4.8.3 Variable Documentation

4.8.3.1 pim

```
uint8_t pim
```

Variable de configuración de Presión inspiratoria máxima.

Read commands

4.8.3.2 stepper

```
FlexyStepper * stepper = new FlexyStepper()
```

Puntero a clase que maneja motor paso a paso.

Dependencies

4.9 Sensors.cpp File Reference

```
#include "Sensors.h"
#include "mpx.h"
```

Variables

- MPX_PRESSURE_SENSOR * **mpx** = new MPX_PRESSURE_SENSOR(PIN_MPX_DATA)
- MPX_PRESSURE_SENSOR * **mpxVol** = new MPX_PRESSURE_SENSOR(PIN_MPX_FLOW)
- float **pressurecmH2O**
- SensorLastPressure_t **lastPres**

4.9.1 Detailed Description

Definiciones de los métodos para utilizar los sensores de presión y volumen

Este archivo contiene las defniciones de las funciones declaradas en Sensors.h

4.10 Sensors.h File Reference

```
#include <stdint.h>
#include "defaults.h"
#include "src/SFM3200/sfm3000wedo.h"
#include "src/FlexyStepper/FlexyStepper.h"
```

Classes

- struct SensorLastPressure_t

  Estructura que almacena las últimas lecturas del sensor de presión (mínima y máxima)
- struct SensorPressureValues_t
- struct SensorVolumeValue_t

  Estructura que contiene información sobre el sensor de volumen.
- class Sensors

  Clase "Sensores". Contiene las variables y definiciones de los métodos para utilizar los sensores de presión y volumen.

Macros

- #define **SENSORS_MAX_ERRORS** 5
- #define **SENSOR_VOLUME_STEPPER_KVolumen** 1.111F

Enumerations

- enum SensorState { **SensorStateOK** = 0, **SensorStateFailed** = 1 }

  Estructura de los posibles estados de un sensor.

4.10.1 Detailed Description

Estructuras con funciones necesarias para cada sensor

Este archivo contiene las declaraciones de las estructuras con funciones necesarias para cada sensor

Index