reaterra

Версия

5/19/2020 12:55:00 PM

# Оглавление

Алфавитный указатель пространств имен 2

Иерархический список классов 3

Алфавитный указатель классов 6

GpioDriver 9

TypeParser 11

Utils 12

Классы 22

BleCharacteristic 22

BleDevice 23

BleObject 25

BleService 26

BluetoothInterface 27

RehabBosBaseMenu::BosCorrectionData 30

Context 31

DbConnector 33

DbError 43

DeviceMonitor 44

DisplayLightnessPwm 45

EnableBitMaskOperators< Enum > 46

Event 47

FreeRunMenu 50

GraphicEngine 51

HrMonitor 55

KbdEvent 57

KeyboardButton 59

Locale 61

Logger 63

MainLoop 65

MainMenu 66

Menu 67

MenuEvent 74

ModalAuthAdmin 75

ModalAuthDoctor 76

ModalCommonDialog 77

ModalCommonError 80

ModalKeyboardAlpha 81

ModalKeyboardDigital 84

ModalListSelector 86

ModalPatientHistoryFilter 87

ModalWarmUpParams 88

Parameter 89

PatientDetailsMenu 91

PatientEditMenu 92

PatientHistoryMenu 93

PatientListMenu 94

PatientNewMenu 95

PatientSearchMenu 97

PatientStatsMenu 98

PersistedParameter 99

PictureManager 100

ProceduresMenu 101

RehabBosBaseMenu 102

RehabBosHrMinMaxMenu 105

RehabBosHrPowerMenu 106

RehabExternalMenu 107

RehabMenu 108

RehabProfileRunMenu 109

RehabProtocolSelectionMenu 110

RehabSelHrMenu 111

RehabSelMethodMenu 112

RehabSelProfileMenu 113

RehabTestProtocolMenu 114

RehabUserProfilSelectMenu 115

RehabUserProtocolsEditMenu 116

RehabUserProtocolsMenu 117

RehabUserProtocolsViewMenu 119

RunSession 121

SettingsAccelMenu 129

SettingsBosParamsMenu 130

SettingsCalibrateMenu 132

SettingsClockMenu 133

SettingsExportMenu 134

SettingsExtraCalibActuatorMenu 135

SettingsExtraCalibDriveMenu 136

SettingsHrMenu 137

SettingsMenu 138

SettingsPasswordMenu 139

SettingsRegistersMenu 140

SettingsUnitsMenu 141

SizeGraph 142

SoundInterface 143

SyncQueue< A > 145

ThreadClass 147

Timer 148

TouchEvent 149

TouchScreenDriver 151

TreadmillDriver 152

UsbFlashDrive 163

WaveFile 165

WBCElement 167

Widget 168

WidgetBreadCrumbs 173

WidgetImageButton 175

WidgetProfil 179

WidgetRectangle 180

WidgetText 182

WidgetTextField 185

WiFiRemote 190

WTFStyle 191

Алфавитный указатель 193

# Алфавитный указатель пространств имен

## Пространства имен

Полный список документированных пространств имен.

**GpioDriver (Управление портами ввода-вывода )** 9

**TypeParser (Создание объектов по имени класса )** 11

**Utils (Вспомогательные утилиты )** 12

# Иерархический список классов

## Иерархия классов

Иерархия классов.

BleObject 25

BleCharacteristic 22

BleDevice 23

BleService 26

BluetoothInterface 27

RehabBosBaseMenu::BosCorrectionData 30

Context 31

DbConnector 33

DeviceMonitor 44

EnableBitMaskOperators< Enum > 46

Event 47

KbdEvent 57

MenuEvent 74

TouchEvent 149

GraphicEngine 51

HrMonitor 55

KeyboardButton 59

Locale 61

Logger 63

MainLoop 65

Menu 67

FreeRunMenu 50

MainMenu 66

ModalAuthAdmin 75

ModalAuthDoctor 76

ModalCommonDialog 77

ModalCommonError 80

ModalKeyboardAlpha 81

ModalKeyboardDigital 84

ModalListSelector 86

ModalPatientHistoryFilter 87

ModalWarmUpParams 88

PatientDetailsMenu 91

PatientEditMenu 92

PatientHistoryMenu 93

PatientListMenu 94

PatientNewMenu 95

PatientSearchMenu 97

PatientStatsMenu 98

ProceduresMenu 101

RehabBosBaseMenu 102

RehabBosHrMinMaxMenu 105

RehabBosHrPowerMenu 106

RehabExternalMenu 107

RehabMenu 108

RehabProfileRunMenu 109

RehabProtocolSelectionMenu 110

RehabSelHrMenu 111

RehabSelMethodMenu 112

RehabSelProfileMenu 113

RehabTestProtocolMenu 114

RehabUserProtocolsEditMenu 116

RehabUserProtocolsMenu 117

RehabUserProfilSelectMenu 115

RehabUserProtocolsViewMenu 119

SettingsAccelMenu 129

SettingsBosParamsMenu 130

SettingsCalibrateMenu 132

SettingsClockMenu 133

SettingsExportMenu 134

SettingsExtraCalibActuatorMenu 135

SettingsExtraCalibDriveMenu 136

SettingsHrMenu 137

SettingsMenu 138

SettingsPasswordMenu 139

SettingsRegistersMenu 140

SettingsUnitsMenu 141

Parameter 89

PersistedParameter 99

PictureManager 100

RunSession 121

runtime\_error

DbError 43

SizeGraph 142

SoundInterface 143

SyncQueue< A > 145

SyncQueue< Command > 145

SyncQueue< Sound > 145

SyncQueue< std::unique\_ptr< Event > > 145

ThreadClass 147

DisplayLightnessPwm 45

TreadmillDriver 152

Timer 148

TouchScreenDriver 151

UsbFlashDrive 163

WaveFile 165

WBCElement 167

Widget 168

WidgetBreadCrumbs 173

WidgetImageButton 175

WidgetProfil 179

WidgetRectangle 180

WidgetText 182

WidgetTextField 185

WiFiRemote 190

WTFStyle 191

# Алфавитный указатель классов

## Классы

Классы с их кратким описанием.

**BleCharacteristic (Характеристика )** 22

**BleDevice (Устройство )** 23

**BleObject (Объект BLE стека. Базовый класс )** 25

**BleService (Сервис )** 26

**BluetoothInterface (Класс драйвера BLE интерфейса )** 27

**RehabBosBaseMenu::BosCorrectionData (Структура данных коррекции )** 30

**Context (Контекст среды исполнения программы )** 31

**DbConnector (Интерфейс БД )** 33

**DbError (Ошибка БД )** 43

**DeviceMonitor (Монитор USB устройств )** 44

**DisplayLightnessPwm**  45

**EnableBitMaskOperators< Enum >**  46

**Event (События )** 47

**FreeRunMenu (Меню Свободный бег )** 50

**GraphicEngine (Графический движок )** 51

**HrMonitor (Драйвер монитора пульса )** 55

**KbdEvent (Событие клавиатуры )** 57

**KeyboardButton (Клавиша клавиатуры )** 59

**Locale (Локализатор )** 61

**Logger (Работа с лог файлом )** 63

**MainLoop (Главный цикл приложения )** 65

**MainMenu (Главное меню приложения )** 66

**Menu (Базовый класс меню )** 67

**MenuEvent (Событие перехода на меню )** 74

**ModalAuthAdmin (Диалог ввода пароля доктора/администратора )** 75

**ModalAuthDoctor (Диалог ввода пароля доктора )** 76

**ModalCommonDialog (Универсальный диалог )** 77

**ModalCommonError (Окно сообщения об ошибке )** 80

**ModalKeyboardAlpha (Алфавитно-цифровая клавиатура )** 81

**ModalKeyboardDigital (Цифровая клавиатура )** 84

**ModalListSelector (Окно выбора из списка )** 86

**ModalPatientHistoryFilter (Окно установки фильтра истории )** 87

**ModalWarmUpParams (Окно установки параметров разминки )** 88

**Parameter (Работа с параметрами )** 89

**PatientDetailsMenu (Меню "детальная иформация о пациенте" )** 91

**PatientEditMenu (Меню "редактор базы пациентов" )** 92

**PatientHistoryMenu (Меню "история пациентов" )** 93

**PatientListMenu (Меню "база пациентов" )** 94

**PatientNewMenu (Меню "создания/редактирования учетной записи пациента" )** 95

**PatientSearchMenu (Меню "поиск пациентов" )** 97

**PatientStatsMenu (Меню "статистика по учетной записи пациента" )** 98

**PersistedParameter (Сохраняемые параметры )** 99

**PictureManager (Управление картинками )** 100

**ProceduresMenu**  101

**RehabBosBaseMenu (Базовый класс для меню бега с биологической обратной связью )** 102

**RehabBosHrMinMaxMenu (Меню "бег с БОС задание ЧСС мин/макс" )** 105

**RehabBosHrPowerMenu (Меню "бег с БОС задание по ЧСС макс/мощности" )** 106

**RehabExternalMenu (Меню "тестирование с внешней стресс-системой" )** 107

**RehabMenu (Меню "выбор типа процедур" )** 108

**RehabProfileRunMenu (Меню "бег по профилю" )** 109

**RehabProtocolSelectionMenu (Меню "выбор протокола тестирования" )** 110

**RehabSelHrMenu (Меню "выбор процедуры с БОС" )** 111

**RehabSelMethodMenu (Меню "выбор метода тестирования" )** 112

**RehabSelProfileMenu (Меню "выбор бега по профилю" )** 113

**RehabTestProtocolMenu (Меню "тестирование по протоколу " )** 114

**RehabUserProfilSelectMenu**  115

**RehabUserProtocolsEditMenu (Меню "редактирование профиля/протокола" )** 116

**RehabUserProtocolsMenu (Меню "пользовательские профили/протоколы" )** 117

**RehabUserProtocolsViewMenu (Меню "просмотр/создание/редактирование профиля/протокола" )** 119

**RunSession (Беговая сессия )** 121

**SettingsAccelMenu (Меню "задание ускорения и торможения" )** 129

**SettingsBosParamsMenu (Меню "Настройка параметров режимов с БОС" )** 130

**SettingsCalibrateMenu (Меню "калибровка положения беговой дорожки" )** 132

**SettingsClockMenu (Меню "настройка времени и даты" )** 133

**SettingsExportMenu (Меню "импорт и экспорт данных" )** 134

**SettingsExtraCalibActuatorMenu (Меню "калибровка подъемника" )** 135

**SettingsExtraCalibDriveMenu (Меню "калибровка привода" )** 136

**SettingsHrMenu (Меню "подключение/отключение датчика ЧСС" )** 137

**SettingsMenu (Меню "настройки" )** 138

**SettingsPasswordMenu (Меню "смена пароля врача" )** 139

**SettingsRegistersMenu (Меню "системная информация" )** 140

**SettingsUnitsMenu (Меню "единицы измерения физических величин" )** 141

**SizeGraph (Размер контура )** 142

**SoundInterface (Управление звуком )** 143

**SyncQueue< A > (Синхронизованная очередь )** 145

**ThreadClass (Базовый класс управления потоком )** 147

**Timer (Реализация периодического таймера )** 148

**TouchEvent (Событие сенсорной панели )** 149

**TouchScreenDriver (Драйвер сенсорной панели )** 151

**TreadmillDriver (Драйвер дорожки )** 152

**UsbFlashDrive (Управление USB-диском )** 163

**WaveFile (Звуковой файл )** 165

**WBCElement (Описание элемента виджета навигации BreadCrumbs )** 167

**Widget (Базовый класс виджетов )** 168

**WidgetBreadCrumbs (Виджет навигации BreadCrumbs )** 173

**WidgetImageButton (Виджет кнопки с картинкой )** 175

**WidgetProfil (Виджет отображения графиков профиля )** 179

**WidgetRectangle (Виджет отображения прямоугольника )** 180

**WidgetText (Виджет однострочного текста )** 182

**WidgetTextField (Виджет многострочного текста с клавиатурой )** 185

**WiFiRemote (Класс поддержки работы с планшетом )** 190

**WTFStyle (Структура стиля виджета )** 191

# Пространства имен

## Пространство имен GpioDriver

Управление портами ввода-вывода.

### Перечисления

enum **PinDirection** { **PinDirection::In**, **PinDirection::Out** }*Направление.*

enum **PinLevel** { **PinLevel::Low**, **PinLevel::High** }*Уровень.*

### Функции

void **setPinDirection** (int pin, **PinDirection** dir)

*Установка направления.*

void **setPinLevel** (int pin, **PinLevel** level)

*Установка уровня.*

**PinLevel** **getPinLevel** (int pin)

*Получить уровень.*

### Подробное описание

Управление портами ввода-вывода.

### Перечисления

#### enum GpioDriver::PinDirection[strong]

Направление.

##### Элементы перечислений:

|  |  |
| --- | --- |
| In | Ввод. |
| Out | Вывод. |

#### enum GpioDriver::PinLevel[strong]

Уровень.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Low | Низкий. |
| High | Высокий. |

### Функции

#### PinLevel GpioDriver::getPinLevel (int *pin*)

Получить уровень.

##### Аргументы:

|  |  |
| --- | --- |
| *pin* | номер пина. |

##### Возвращает:

уровень.

#### void GpioDriver::setPinDirection (int *pin*, PinDirection *dir*)

Установка направления.

##### Аргументы:

|  |  |
| --- | --- |
| *pin* | номер пина. |
| *dir* | направление. |

#### void GpioDriver::setPinLevel (int *pin*, PinLevel *level*)

Установка уровня.

##### Аргументы:

|  |  |
| --- | --- |
| *pin* | номер пина. |
| *level* | уровень. |

## Пространство имен TypeParser

Создание объектов по имени класса.

### Функции

std::unique\_ptr< **Menu** > **createObject** (std::string type\_name)

### Подробное описание

Создание объектов по имени класса.

## Пространство имен Utils

Вспомогательные утилиты.

### Перечисления

enum **Phys** { **Phys::Speed**, **Phys::Distance**, **Phys::Energy**, **Phys::Power**, **Phys::PowerAir**, **Phys::Angle** }*Физические величины.*

enum **Unit** { **Unit::Meter**, **Unit::KiloMeter**, **Unit::Mile**, **Unit::KmPerHour**, **Unit::MPerSec**, **Unit::MilePerHour**, **Unit::Degree**, **Unit::Radian**, **Unit::Procent**, **Unit::KiloKall**, **Unit::Joul**, **Unit::KWattPerHour**, **Unit::Watt**, **Unit::KiloWatt**, **Unit::MET**, **Unit::MlMinKg**, **Unit::Unknown** }*Единицы измерения.*

### Функции

int **makeDir** (const char \*path)

*Создать каталог.*

int **makePath** (const char \*path)

*Создать путь (с подкаталогами).*

void **unlockFile** (void)

bool **lockFile** (std::string lock\_file)

*Занять лок файл.*

int **openSerial** (std::string port\_name)

*Открыть последовательный порт.*

std::string **int2Hex** (int n, int wide=0)

*Преобразование числа в шестнадцатеричную строку.*

std::string **bytes2Hex** (const unsigned char \*bytes, int size, bool delim=true)

*Преобразование массива байт в шестнадцатеричную строку.*

unsigned short **ba2S** (unsigned char low, unsigned char high)

*Преобразование двух байт в unsigned short.*

unsigned char **s2LB** (unsigned short s)

*Выделение младшего байта из unsigned short.*

unsigned char **s2HB** (unsigned short s)

*Выделение старшего байта из unsigned short.*

std::wstring **utf8\_to\_wstring** (const std::string &str)

*Преобразование строки в кодировке UTF-8 в строку wstring.*

std::string **wstring\_to\_utf8** (const std::wstring &str)

*Преобразование строки wstring в строку в кодировке UTF-8.*

std::u16string **utf8\_to\_u16string** (const std::string &str)

*Преобразование строки в кодировке UTF-8 в строку u16string.*

std::string **u16string\_to\_utf8** (const std::u16string &str)

*Преобразование строки u16string в строку в кодировке UTF-8.*

std::string **getFileTimestamp** ()

*Получение строки с текущим временем в формате YYYY-MM-DD.HHMMSS.*

std::string **getCurrentTimeString** (std::string format)

*Получение текущего времени используя строку формата.*

struct tm \* **getCurrentTime** ()

*Получение текущего времени.*

std::string **getLogTimestamp** ()

*Получение строки с текущим временем в формате YYYY-MM-DD HH:MM:SS.mmm.*

float **deg2Radian** (float deg)

*Преобразование градусов в радианы.*

float **ms2kmh** (float ms)

*Преобразование метров в секунду в километры в час.*

float **milih2kmh** (float milih)

*Преобразование милей в час в километры в час.*

float **kmh2ms** (float kmh)

*Преобразование километров в час в метры в секунду.*

float **kmh2milih** (float kmh)

*Преобразование километров в час в мили в час.*

std::string **allTrim** (std::string str)

*Удаление пробелов с начала и конца строки.*

float **physConvert** (**Phys** phys, **Unit** from, **Unit** to, float value)

std::string **formatFloat** (std::string fmt, float value)

*Форматирование числа с плавающей точкой.*

std::string **formatTime** (int sec)

*Форматирование времени заданном в секундах.*

bool **pathExist** (std::string path)

*Проверка существования указанного пути.*

int **setDateTime** (const struct tm \*time)

*Установка даты и времени в системе и в RTC.*

bool **isLeapYear** (unsigned year)

*Проверка года на високосность.*

std::string **typeName** (const std::type\_info &type\_inf)

*Название типа.*

std::vector< std::string > **split** (std::string str, char delimiter)

std::string **unsplit** (std::vector< std::string > vec, char delimiter)

template<typename T > constexpr T **clamp** (const T &n, const T &lower, const T &upper)

*Ограничение числа в пределах.*

### Подробное описание

Вспомогательные утилиты.

### Перечисления

#### enum Utils::Phys[strong]

Физические величины.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Speed | Скорость. |
| Distance | Расстояние. |
| Energy | Энергия. |
| Power | Механическая мощность. |
| PowerAir | Аэробная мощность. |
| Angle | Угол наклона. |

#### enum Utils::Unit[strong]

Единицы измерения.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Meter | Метры. |
| KiloMeter | Километры. |
| Mile | Мили. |
| KmPerHour | Километры в час |
| MPerSec | Метры в секунду. |
| MilePerHour | Мили в час. |
| Degree | Градусы. |
| Radian | Радианы. |
| Procent | Проценты. |
| KiloKall | Килокалории. |
| Joul | Джоули. |
| KWattPerHour | Киловатты в час. |
| Watt | Ватты. |
| KiloWatt | Киловатты. |
| MET | МЕТ-ы |
| MlMinKg | мл/мин/кг. |
| Unknown | Неизвестный. |

### Функции

#### std::string Utils::allTrim (std::string *str*)

Удаление пробелов с начала и конца строки.

##### Аргументы:

|  |  |
| --- | --- |
| *str* | строка с пробелами. |

##### Возвращает:

строка без пробелов.

#### unsigned short Utils::ba2S (unsigned char *low*, unsigned char *high*)[inline]

Преобразование двух байт в unsigned short.

##### Аргументы:

|  |  |
| --- | --- |
| *low* | младший байт. |
| *high* | старший байт. |

##### Возвращает:

результат преобразования.

#### std::string Utils::bytes2Hex (const unsigned char \* *bytes*, int *size*, bool *delim* = true)

Преобразование массива байт в шестнадцатеричную строку.

##### Аргументы:

|  |  |
| --- | --- |
| *bytes* | массив байт. |
| *size* | размер массива. |
| *delim* | использовать или нет разделитель. |

##### Возвращает:

шестнадцатеричная строка.

#### template<typename T > constexpr T Utils::clamp (const T & *n*, const T & *lower*, const T & *upper*)

Ограничение числа в пределах.

##### Аргументы:

|  |  |
| --- | --- |
| *n* | число. |
| *lower* | нижний предел. |
| *upper* | верхний предел. |

##### Возвращает:

число ограниченное в пределах lower <= n <= upper.

#### float Utils::deg2Radian (float *deg*)

Преобразование градусов в радианы.

##### Аргументы:

|  |  |
| --- | --- |
| *deg* | значение в градусах. |

##### Возвращает:

значение в радианах.

#### std::string Utils::formatFloat (std::string *fmt*, float *value*)

Форматирование числа с плавающей точкой.

##### Аргументы:

|  |  |
| --- | --- |
| *fmt* | формат в стиле printf. |
| *value* | число. |

##### Возвращает:

строка с числом в заданном формате.

#### std::string Utils::formatTime (int *sec*)

Форматирование времени заданном в секундах.

##### Аргументы:

|  |  |
| --- | --- |
| *sec* | время в секундах. |

##### Возвращает:

строка с временем в формате HH:MM:SS.

#### struct tm \* Utils::getCurrentTime ()

Получение текущего времени.

##### Возвращает:

ссылка на структуру tm с текущим временем.

#### std::string Utils::getCurrentTimeString (std::string *format*)

Получение текущего времени используя строку формата.

##### Аргументы:

|  |  |
| --- | --- |
| *format* | строка формата совместимая с strftime. |

##### Возвращает:

строка с текущим временем в указаном формате.

#### std::string Utils::getFileTimestamp ()

Получение строки с текущим временем в формате YYYY-MM-DD.HHMMSS.

##### Возвращает:

строка с текущим временем.

#### std::string Utils::getLogTimestamp ()

Получение строки с текущим временем в формате YYYY-MM-DD HH:MM:SS.mmm.

##### Возвращает:

строка с текущим временем.

#### std::string Utils::int2Hex (int *n*, int *wide* = 0)

Преобразование числа в шестнадцатеричную строку.

##### Аргументы:

|  |  |
| --- | --- |
| *n* | число. |
| *wide* | ширина результата. |

##### Возвращает:

шестнадцатеричная строка.

#### bool Utils::isLeapYear (unsigned *year*)

Проверка года на високосность.

##### Аргументы:

|  |  |
| --- | --- |
| *year* | значение года. |

##### Возвращает:

true - високосный.

#### float Utils::kmh2milih (float *kmh*)

Преобразование километров в час в мили в час.

##### Аргументы:

|  |  |
| --- | --- |
| *kmh* | значение в километрах в час. |

##### Возвращает:

значение в милях в час.

#### float Utils::kmh2ms (float *kmh*)

Преобразование километров в час в метры в секунду.

##### Аргументы:

|  |  |
| --- | --- |
| *kmh* | значение в километрах в час. |

##### Возвращает:

значение в метрах в секунду.

#### bool Utils::lockFile (std::string *lock\_file*)

Занять лок файл.

##### Аргументы:

|  |  |
| --- | --- |
| *lock\_file* | имя файла. |

##### Возвращает:

true: успех.

#### int Utils::makeDir (const char \* *path*)

Создать каталог.

##### Аргументы:

|  |  |
| --- | --- |
| *path* | имя каталога. |

##### Возвращает:

0 - успешно.

#### int Utils::makePath (const char \* *path*)

Создать путь (с подкаталогами).

##### Аргументы:

|  |  |
| --- | --- |
| *path* | путь. |

##### Возвращает:

0 - успешно.

#### float Utils::milih2kmh (float *milih*)

Преобразование милей в час в километры в час.

##### Аргументы:

|  |  |
| --- | --- |
| *milih* | значение в милях в час. |

##### Возвращает:

значение в километрах в час.

#### float Utils::ms2kmh (float *ms*)

Преобразование метров в секунду в километры в час.

##### Аргументы:

|  |  |
| --- | --- |
| *ms* | значение в метрах в секунду. |

##### Возвращает:

значение в километрах в час.

#### int Utils::openSerial (std::string *port\_name*)

Открыть последовательный порт.

##### Аргументы:

|  |  |
| --- | --- |
| *port\_name* | имя порта. |

##### Возвращает:

-1: ошибка.

#### bool Utils::pathExist (std::string *path*)

Проверка существования указанного пути.

##### Аргументы:

|  |  |
| --- | --- |
| *path* | путь в файловой системе. |

##### Возвращает:

true - путь существует, false - путь отсутствует.

#### float Utils::physConvert (Phys *phys*, Unit *from*, Unit *to*, float *value*)

Преобразование физических величин.

##### Аргументы:

|  |  |
| --- | --- |
| *phys* | физическая величина. |
| *from* | в каких единицах задано значение. |
| *to* | в какие единицы перевести. |
| *value* | заданное значение. |

##### Возвращает:

преобразованное значение.

#### unsigned char Utils::s2HB (unsigned short *s*)[inline]

Выделение старшего байта из unsigned short.

##### Аргументы:

|  |  |
| --- | --- |
| *s* | значение unsigned short. |

##### Возвращает:

старший байт.

#### unsigned char Utils::s2LB (unsigned short *s*)[inline]

Выделение младшего байта из unsigned short.

##### Аргументы:

|  |  |
| --- | --- |
| *s* | значение unsigned short. |

##### Возвращает:

младший байт.

#### int Utils::setDateTime (const struct tm \* *time*)

Установка даты и времени в системе и в RTC.

##### Аргументы:

|  |  |
| --- | --- |
| *time* | значение даты и времени. |

##### Возвращает:

код возврата, 0 - успех.

#### std::string Utils::typeName (const std::type\_info & *type\_inf*)

Название типа.

##### Аргументы:

|  |  |
| --- | --- |
| *type* | структура полученная вызовом typeid. |

##### Возвращает:

строка с названием типа.

#### std::string Utils::u16string\_to\_utf8 (const std::u16string & *str*)

Преобразование строки u16string в строку в кодировке UTF-8.

##### Аргументы:

|  |  |
| --- | --- |
| *str* | строка u16string. |

##### Возвращает:

строка в кодировке UTF-8.

#### std::u16string Utils::utf8\_to\_u16string (const std::string & *str*)

Преобразование строки в кодировке UTF-8 в строку u16string.

##### Аргументы:

|  |  |
| --- | --- |
| *str* | строка в кодировке UTF-8. |

##### Возвращает:

строка u16string.

#### std::wstring Utils::utf8\_to\_wstring (const std::string & *str*)

Преобразование строки в кодировке UTF-8 в строку wstring.

##### Аргументы:

|  |  |
| --- | --- |
| *str* | строка в кодировке UTF-8. |

##### Возвращает:

строка wstring.

#### std::string Utils::wstring\_to\_utf8 (const std::wstring & *str*)

Преобразование строки wstring в строку в кодировке UTF-8.

##### Аргументы:

|  |  |
| --- | --- |
| *str* | строка wstring. |

##### Возвращает:

строка в кодировке UTF-8.

# Классы

## Структура BleCharacteristic

Характеристика.

#include <BluetoothInterface.h>

Граф наследования:BleCharacteristic:

### Открытые члены

**BleCharacteristic** (std::string \_path, std::string \_uuid, std::string \_service)

*Конструктор.*

### Открытые атрибуты

std::string **uuid**

*Идентификатор.*

std::string **service**

*Сервис к которому принадлежит характеристика.*

std::function< void(std::vector< unsigned char >)> **on\_notify**

*Функция вызываемая при изменении характеристики.*

### Подробное описание

Характеристика.

### Конструктор(ы)

#### BleCharacteristic::BleCharacteristic (std::string *\_path*, std::string *\_uuid*, std::string *\_service*)[inline]

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *\_path* | путь к объекту. |
| *\_uuid* | идентификатор объекта. |
| *\_service* | сервис к которому принадлежит характеристика. |

#### Объявления и описания членов структуры находятся в файле:

src/BluetoothInterface.h

## Структура BleDevice

Устройство.

#include <BluetoothInterface.h>

Граф наследования:BleDevice:

### Открытые члены

**BleDevice** (std::string \_path, std::string \_address={}, std::string \_name={}, bool \_connected=false)

*Конструктор.*

### Открытые атрибуты

std::string **address**

*Уникальный BLE адрес устройства.*

std::string **name**

*Имя устройства.*

bool **connected**

*Флаг состояния подключения.*

std::vector< std::shared\_ptr< **BleCharacteristic** > > **characteristics**

*Список характеристик устройства.*

std::vector< std::shared\_ptr< **BleService** > > **services**

*Список сервисов устройства.*

### Подробное описание

Устройство.

### Конструктор(ы)

#### BleDevice::BleDevice (std::string *\_path*, std::string *\_address* = {}, std::string *\_name* = {}, bool *\_connected* = false)[inline]

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *\_path* | путь к устройству. |
| *\_address* | BLE адрес устройства. |
| *\_name* | имя устройства. |
| *\_connected* | (опционально) флаг состояния подключения. |

#### Объявления и описания членов структуры находятся в файле:

src/BluetoothInterface.h

## Структура BleObject

Объект BLE стека. Базовый класс.

#include <BluetoothInterface.h>

Граф наследования:BleObject:

### Открытые члены

**BleObject** (BleObjectType t, std::string p)

*Конструктор.*

virtual **~BleObject** ()=default

*Деструктор.*

### Открытые атрибуты

BleObjectType **type**

*Тип объекта.*

std::string **path**

*Путь к объекту (его адрес в иерархии).*

### Подробное описание

Объект BLE стека. Базовый класс.

### Конструктор(ы)

#### BleObject::BleObject (BleObjectType *t*, std::string *p*)[inline]

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *t* | тип объекта. |
| *p* | путь к объекту. |

#### Объявления и описания членов структуры находятся в файле:

src/BluetoothInterface.h

## Структура BleService

Сервис.

#include <BluetoothInterface.h>

Граф наследования:BleService:

### Открытые члены

**BleService** (std::string \_path, std::string \_uuid, std::string \_device)

*Конструктор.*

### Открытые атрибуты

std::string **uuid**

*Идентификатор.*

std::string **device**

*Устройство к которому принадлежит сервис.*

std::vector< std::shared\_ptr< **BleCharacteristic** > > **characteristics**

*Список характеристик сервиса.*

### Подробное описание

Сервис.

### Конструктор(ы)

#### BleService::BleService (std::string *\_path*, std::string *\_uuid*, std::string *\_device*)[inline]

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *\_path* | путь к объекту. |
| *\_uuid* | идентификатор объекта. |
| *\_device* | устройство к которому принадлежит сервис. |

#### Объявления и описания членов структуры находятся в файле:

src/BluetoothInterface.h

## Класс BluetoothInterface

Класс драйвера BLE интерфейса.

#include <BluetoothInterface.h>

### Открытые члены

**BluetoothInterface** (std::function< void(BleDeviceSP)> on\_connect={}, std::function< void(BleDeviceSP)> on\_disconnect={})

*Конструктор.*

virtual **~BluetoothInterface** ()

*Деструктор.*

void **startScan** (std::function< void(BleDeviceSP)> on\_device\_found={})

*Запуск сканирования.*

void **stopScan** ()

*Остановка сканирования.*

int **connect** (std::string address)

*Подключение к устройству.*

void **disconnect** (BleDeviceSP d)

*Отключение устройства.*

void **startNotify** (BleDeviceSP d, std::string uuid, std::function< void(std::vector< unsigned char >)> on\_notify={})

*Включение отправки уведомлений характеристики на устройстве.*

std::vector< unsigned char > **readCharacteristic** (BleDeviceSP d, std::string ch)

*Чтение значения характеристики.*

void **addDevice** (BleDeviceSP device)

*Добавление устройства в список.*

void **removeDevices** ()

*Удаление всех устройств из списка.*

bool **needScan** ()

*Определение необходимости выполнять сканирование.*

void **onSignal** (GDBusConnection \*connection, const gchar \*sender\_name, const gchar \*object\_path, const gchar \*interface\_name, const gchar \*signal\_name, GVariant \*parameters, gpointer user\_data)

*Обработчик событий DBus.*

### Открытые атрибуты

BleDeviceList **m\_devices**

*Список устройств.*

### Подробное описание

Класс драйвера BLE интерфейса.

### Конструктор(ы)

#### BluetoothInterface::BluetoothInterface (std::function< void(BleDeviceSP)> *on\_connect* = {}, std::function< void(BleDeviceSP)> *on\_disconnect* = {})

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *on\_connect* | функция вызываемая при подключении устройства. |
| *on\_disconnect* | функция вызываемая при отключении устройства. |

### Методы

#### void BluetoothInterface::addDevice (BleDeviceSP *device*)

Добавление устройства в список.

##### Аргументы:

|  |  |
| --- | --- |
| *device* | добавляемое устройство. |

#### int BluetoothInterface::connect (std::string *address*)

Подключение к устройству.

##### Аргументы:

|  |  |
| --- | --- |
| *address* | BLE адрес устройства. |

##### Возвращает:

0: успех, -1: неудача.

#### void BluetoothInterface::disconnect (BleDeviceSP *d*)

Отключение устройства.

##### Аргументы:

|  |  |
| --- | --- |
| *d* | отключаемое устройство. |

#### bool BluetoothInterface::needScan ()

Определение необходимости выполнять сканирование.

##### Возвращает:

true: сканирование необходимо.

#### void BluetoothInterface::onSignal (GDBusConnection \* *connection*, const gchar \* *sender\_name*, const gchar \* *object\_path*, const gchar \* *interface\_name*, const gchar \* *signal\_name*, GVariant \* *parameters*, gpointer *user\_data*)

Обработчик событий DBus.

##### Аргументы:

|  |  |
| --- | --- |
| *connection* |  |
| *sender\_name* |  |
| *object\_path* |  |
| *interface\_name* |  |
| *signal\_name* |  |
| *parameters* |  |
| *user\_data* |  |

#### std::vector< unsigned char > BluetoothInterface::readCharacteristic (BleDeviceSP *d*, std::string *ch*)

Чтение значения характеристики.

##### Аргументы:

|  |  |
| --- | --- |
| *d* | устройство. |
| *ch* | идентификатор характеристики. |

##### Возвращает:

значение характеристики.

#### void BluetoothInterface::startNotify (BleDeviceSP *d*, std::string *uuid*, std::function< void(std::vector< unsigned char >)> *on\_notify* = {})

Включение отправки уведомлений характеристики на устройстве.

##### Аргументы:

|  |  |
| --- | --- |
| *d* | устройство. |
| *uuid* | идентификатор характеристики. |
| *on\_notify* | (опционально) функция вызываемая при получении уведомления. |

#### void BluetoothInterface::startScan (std::function< void(BleDeviceSP)> *on\_device\_found* = {})

Запуск сканирования.

##### Аргументы:

|  |  |
| --- | --- |
| *on\_device\_found* | (опционально) функция вызываемая при обнаружении устройства. |

#### Объявления и описания членов классов находятся в файлах:

src/BluetoothInterface.h

src/BluetoothInterface.cpp

## Структура RehabBosBaseMenu::BosCorrectionData

Структура данных коррекции.

#include <RehabBosBaseMenu.h>

### Открытые атрибуты

int **interval**

int **speed**

int **angle**

### Подробное описание

Структура данных коррекции.

Объявления и описания членов структуры находятся в файле:

src/RehabBosBaseMenu.h

## Класс Context

Контекст среды исполнения программы.

#include <Context.h>

### Открытые члены

**Context** ()

*Конструктор*

virtual **~Context** ()

*Деструктор*

### Открытые атрибуты

**PersistedParameter** **m\_settings**

*Управление настройками.*

**DbConnector** **m\_db**

*Доступ к БД.*

**SyncQueue**< std::unique\_ptr< **Event** > > **m\_queue** {"Main Loop"}

*Очередь сообщений.*

**SoundInterface** **m\_sound**

*Звуковой интерфейс.*

**DeviceMonitor** **m\_dev\_mon**

*Управление устройствами.*

**TreadmillDriver** **m\_treadmill**

*Интерфейс с дорожкой.*

**GraphicEngine** **m\_graphic**

*Графическая система.*

**PictureManager** **m\_pic\_mgr**

*Управление картинками.*

**TouchScreenDriver** **m\_touch**

*Сенсорный ввод.*

**HrMonitor** **m\_hr\_monitor**

*Датчик пульса.*

**RunSession** **m\_session**

*Беговая сессия.*

**WiFiRemote** **m\_remote**

*Интерфейс с планшетом.*

std::string **m\_patient**

*Текущий пациент.*

int **m\_patient\_id**

*Идентификатор текущего пациента.*

int **m\_profile\_id**

*Идентификатор профиля пациента.*

int **m\_accel**

*Уровень ускорения по умолчанию.*

int **m\_decel**

*Уровень торможения по умолчанию.*

std::string **m\_admin**

*Текущий пароль.*

float **m\_patient\_weight**

*Вес пациента.*

bool **m\_notuse\_doc\_password**

*Флаг отмены пароля врача.*

std::vector< std::string > **m\_menu\_history**

std::map< std::string, std::vector< std::string > > **m\_prev\_menu**

*Контекст предыдущего меню*

### Статические открытые данные

static const char **PARAMS\_SEPARATOR** = '^'

static std::string **main\_board** {"PI"}

### Подробное описание

Контекст среды исполнения программы.

Объявления и описания членов классов находятся в файлах:

src/Context.h

src/Context.cpp

## Класс DbConnector

Интерфейс БД.

#include <DbConnector.h>

### Открытые типы

enum **ProtocolParam** { **ProtocolParam::Type** = 0, **ProtocolParam::Name** = 1, **ProtocolParam::Modification** = 2, **ProtocolParam::DurationUnit** = 3, **ProtocolParam::DurationFmt** = 4, **ProtocolParam::SpeedUnit** = 5, **ProtocolParam::SpeedFmt** = 6, **ProtocolParam::AngleUnit** = 7, **ProtocolParam::AngleFmt** = 8, **ProtocolParam::Description** = 9 }*Параметры протоколов.*

enum **IntervalData** { **IntervalData::Name** = 0, **IntervalData::Duration** = 1, **IntervalData::Speed** = 2, **IntervalData::Angle** = 3, **IntervalData::Type** = 4, **IntervalData::Acceleration** = 5 }*Параметры интервала.*

enum **ProtocolList** { **ProtocolList::Name** = 0, **ProtocolList::Id** = 1, **ProtocolList::Description** = 2 }*Параметры протокола.*

enum **PatientList** { **PatientList::Id** = 0, **PatientList::Number** = 1, **PatientList::LastName** = 2, **PatientList::FirstName** = 3, **PatientList::MiddleName** = 4, **PatientList::Sex** = 5, **PatientList::Age** = 6, **PatientList::Diagnosis** = 7, **PatientList::Doctor** = 8, **PatientList::Weight** = 9, **PatientList::ProfileId** = 10, **PatientList::Profile** = 11 }*Параметры пациента.*

enum **Statistic** { **Statistic::Fio** = 0, **Statistic::Distance** = 1, **Statistic::Duration** = 2, **Statistic::Energy** = 3, **Statistic::Acceleration** = 4, **Statistic::HRateMin** = 5, **Statistic::HRateMax** = 6 }*Параметры статистики.*

enum **History** { **History::Fio** = 0, **History::Time** = 1, **History::Profile** = 2, **History::Distance** = 3, **History::Duration** = 4, **History::Speed** = 5, **History::Acceleration** = 6, **History::Angle** = 7, **History::Energy** = 8, **History::PowerMech** = 9, **History::PowerAero** = 10, **History::HRate** = 11, **History::LastName** = 12, **History::FirstName** = 13, **History::MiddleName** = 14, **History::TimeStamp** = 15 }*Параметры истории.*

### Открытые члены

**DbConnector** ()

*Конструктор.*

virtual **~DbConnector** ()

*Деструктор.*

DataSet **getProtocols** (std::string type)

*Получение списка протоколов.*

DataSet **getProtocolModifs** (int id)

*Получение модификаций протокола.*

DataSet **getProtocolParams** (int id)

*Получение параметров протокола.*

DataSet **getIntervals** (int id)

*Получение интервалов протокола.*

std::string **getParam** (std::string name, std::string def="no\_insert")

*Запрос параметра.*

void **setParam** (std::string name, std::string value)

*Установка параметра.*

int **insertProtocol** (DataSet protocol, DataSet intervals, std::string pro\_type)

*Вставка нового протокола.*

void **updateProtocol** (int id, DataSet protocol, DataSet intervals)

*Обновление данных протокола.*

void **deleteProtocol** (int id)

*Удаление протокола.*

void **deletePatient** (int id)

*Удаление пациента.*

std::string **backupDB** ()

*Создание резервной копии БД.*

void **restoreDB** (std::string gz\_dump)

*Восстановление БД.*

**Utils::Unit** **getParamEnum** (std::string name)

*Получение значения единиц измерения по названию.*

DataSet **getPatients** (std::string order, bool asc, std::string filter="")

*Получение списка пациентов.*

DataSet **getPatient** (int id)

*Получение данных пациента.*

int **insertPatient** (DataSet patient)

*Добавление пациента.*

void **updatePatient** (DataSet patient)

*Обновление данных пациента.*

void **addRunSession** (**RunSession** &session)

*Добавление записи в историю.*

DataSet **getStatistic** (int patient\_id)

*Получение статистики.*

DataSet **getHistory** (std::string from="", std::string to="", std::string fio="")

*Получение истории.*

void **exportToEmi** (std::string file)

*Экспорт данных в формате Weintek.*

void **importFromEmi** (std::string file)

*Импорт данных в формате Weintek.*

### Открытые статические члены

static std::string **dumpDataSet** (DataSet ds, std::string name="")

*Формирование текстового дампа из набора данных.*

### Подробное описание

Интерфейс БД.

### Перечисления

#### enum DbConnector::History[strong]

Параметры истории.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Fio | ФИО. |
| Time | Потраченное время. |
| Profile | Название профиля. |
| Distance | Пройденное расстояние. |
| Duration | Потраченное время. |
| Speed | Средняя скорость. |
| Acceleration | Максимальное ускорение. |
| Angle | Максимальный угол наклона. |
| Energy | Затраченная энергия. |
| PowerMech | Средняя механическая мощность. |
| PowerAero | Средняя аэробная мощность. |
| HRate | Средний пульс. |
| LastName | Фамилия. |
| FirstName | Имя. |
| MiddleName | Отчество. |
| TimeStamp | Время записи. |

#### enum DbConnector::IntervalData[strong]

Параметры интервала.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Name | Номер интервала. |
| Duration | Длительность. |
| Speed | Скорость. |
| Angle | Угол наклона. |
| Type | Тип. |
| Acceleration | Ускорение. |

#### enum DbConnector::PatientList[strong]

Параметры пациента.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Id | Идентификатор. |
| Number | Номер. |
| LastName | Фамилия. |
| FirstName | Имя. |
| MiddleName | Отчество. |
| Sex | Пол. |
| Age | Возраст. |
| Diagnosis | Диагноз. |
| Doctor | Лечащий врач. |
| Weight | Вес. |
| ProfileId | Идентификатор профиля. |
| Profile | Наименование профиля. |

#### enum DbConnector::ProtocolList[strong]

Параметры протокола.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Name | Название. |
| Id | Идентификатор. |
| Description | Описание. |

#### enum DbConnector::ProtocolParam[strong]

Параметры протоколов.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Type | Тип. |
| Name | Название. |
| Modification | Модификация. |
| DurationUnit | Единицы времени. |
| DurationFmt | Формат времени. |
| SpeedUnit | Единицы скорости. |
| SpeedFmt | Формат скорости. |
| AngleUnit | Единицы угла наклона. |
| AngleFmt | Формат угла наклона. |
| Description | Описание. |

#### enum DbConnector::Statistic[strong]

Параметры статистики.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Fio | ФИО. |
| Distance | Пройденное расстояние. |
| Duration | Потраченное время. |
| Energy | Затраченная энергия. |
| Acceleration | Максимальное ускорение. |
| HRateMin | Минимальный пульс. |
| HRateMax | Максимальный пульс. |

### Методы

#### void DbConnector::addRunSession (RunSession & *session*)

Добавление записи в историю.

##### Аргументы:

|  |  |
| --- | --- |
| *session* | ссылка на беговую сессию. |

#### void DbConnector::deletePatient (int *id*)

Удаление пациента.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | идентификатор удаляемого пациента. |

#### void DbConnector::deleteProtocol (int *id*)

Удаление протокола.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | идентификатор удаляемого протокола. |

#### std::string DbConnector::dumpDataSet (DataSet *ds*, std::string *name* = "")[static]

Формирование текстового дампа из набора данных.

##### Аргументы:

|  |  |
| --- | --- |
| *ds* | набор данных. |
| *name* | пометка. |

##### Возвращает:

текст дампа.

#### void DbConnector::exportToEmi (std::string *file*)

Экспорт данных в формате Weintek.

##### Аргументы:

|  |  |
| --- | --- |
| *file* | название файла для экспорта. |

#### DataSet DbConnector::getHistory (std::string *from* = "", std::string *to* = "", std::string *fio* = "")

Получение истории.

##### Аргументы:

|  |  |
| --- | --- |
| *from* | начало периода. |
| *to* | конец периода. |
| *fio* | фильтр по фамилии пациента. |

##### Возвращает:

данные истории.

#### DataSet DbConnector::getIntervals (int *id*)

Получение интервалов протокола.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | идентификатор протокола. |

##### Возвращает:

интервалы протокола.

#### std::string DbConnector::getParam (std::string *name*, std::string *def* = "no\_insert")

Запрос параметра.

##### Аргументы:

|  |  |
| --- | --- |
| *name* | имя параметра. |
| *def* | значение по умолчанию. |

##### Возвращает:

значение параметра.

#### Utils::Unit DbConnector::getParamEnum (std::string *name*)

Получение значения единиц измерения по названию.

##### Аргументы:

|  |  |
| --- | --- |
| *name* | название единицы измерения. |

##### Возвращает:

значение перечисления.

#### DataSet DbConnector::getPatient (int *id*)

Получение данных пациента.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | идентификатор пациента. |

##### Возвращает:

данные пациента.

#### DataSet DbConnector::getPatients (std::string *order*, bool *asc*, std::string *filter* = "")

Получение списка пациентов.

##### Аргументы:

|  |  |
| --- | --- |
| *order* | поле сортировки. |
| *asc* | направление сортировки. |
| *filter* | фильтр записей. |

##### Возвращает:

список пациентов.

#### DataSet DbConnector::getProtocolModifs (int *id*)

Получение модификаций протокола.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | идентификатор протокола. |

##### Возвращает:

список модификаций.

#### DataSet DbConnector::getProtocolParams (int *id*)

Получение параметров протокола.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | идентификатор протокола. |

##### Возвращает:

параметры протокола.

#### DataSet DbConnector::getProtocols (std::string *type*)

Получение списка протоколов.

##### Аргументы:

|  |  |
| --- | --- |
| *type* | тип протоколов ("ПРОТ10"/"USER"/"USER\_PROF"). |

##### Возвращает:

список протоколов.

#### DataSet DbConnector::getStatistic (int *patient\_id*)

Получение статистики.

##### Аргументы:

|  |  |
| --- | --- |
| *patient\_id* | идентификатор пациента. |

##### Возвращает:

данные статистики.

#### void DbConnector::importFromEmi (std::string *file*)

Импорт данных в формате Weintek.

##### Аргументы:

|  |  |
| --- | --- |
| *file* | название файла для импорта. |

#### int DbConnector::insertPatient (DataSet *patient*)

Добавление пациента.

##### Аргументы:

|  |  |
| --- | --- |
| *patient* | данные пациента. |

##### Возвращает:

идентификатор пациента или -1 в случае ошибки

#### int DbConnector::insertProtocol (DataSet *protocol*, DataSet *intervals*, std::string *pro\_type*)

Вставка нового протокола.

##### Аргументы:

|  |  |
| --- | --- |
| *protocol* | парамтры протокола. |
| *intervals* | параметры интервалов. |
| *pro\_type* | тип протокола. |

##### Возвращает:

идентификатор протокола или -1 в случае ошибки.

#### void DbConnector::restoreDB (std::string *gz\_dump*)

Восстановление БД.

##### Аргументы:

|  |  |
| --- | --- |
| *gz\_dump* | путь к файлу резервной копии. |

#### void DbConnector::setParam (std::string *name*, std::string *value*)

Установка параметра.

##### Аргументы:

|  |  |
| --- | --- |
| *name* | имя параметра. |
| *value* | значение параметра. |

#### void DbConnector::updatePatient (DataSet *patient*)

Обновление данных пациента.

##### Аргументы:

|  |  |
| --- | --- |
| *patient* | данные пациента. |

#### void DbConnector::updateProtocol (int *id*, DataSet *protocol*, DataSet *intervals*)

Обновление данных протокола.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | идентификатор протокола. |
| *protocol* | параметры протокола. |
| *intervals* | интервалы протокола. |

#### Объявления и описания членов классов находятся в файлах:

src/DbConnector.h

src/DbConnector.cpp

## Класс DbError

Ошибка БД.

#include <DbConnector.h>

Граф наследования:DbError:

### Открытые члены

**DbError** (const std::string &what\_arg, unsigned int code=0)

### Открытые атрибуты

unsigned int **error\_code**

*Код ошибки.*

### Подробное описание

Ошибка БД.

### Конструктор(ы)

#### DbError::DbError (const std::string & *what\_arg*, unsigned int *code* = 0)[inline], [explicit]

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *what\_arg* | описание ошибки. |
| *code* | (опционально) код ошибки. 0 по умолчанию. |

#### Объявления и описания членов класса находятся в файле:

src/DbConnector.h

## Класс DeviceMonitor

Монитор USB устройств.

#include <DeviceMonitor.h>

### Открытые члены

**DeviceMonitor** ()

*Конструктор.*

virtual **~DeviceMonitor** ()

*Деструктор.*

std::vector< **UsbFlashDrive** > & **getUsbDrives** ()

*Получение списка USB устройств.*

void **setUsbDriveEventListener** (std::function< void(std::string type, **UsbFlashDrive** &drive)> on\_event)

*Установка функции обработки события.*

### Подробное описание

Монитор USB устройств.

### Методы

#### std::vector< UsbFlashDrive > & DeviceMonitor::getUsbDrives ()

Получение списка USB устройств.

##### Возвращает:

список устройств.

#### void DeviceMonitor::setUsbDriveEventListener (std::function< void(std::string type, UsbFlashDrive &drive)> *on\_event*)

Установка функции обработки события.

##### Аргументы:

|  |  |
| --- | --- |
| *on\_event* | функция вызываемая при подключении/отключении устройства. |

#### Объявления и описания членов классов находятся в файлах:

src/DeviceMonitor.h

src/DeviceMonitor.cpp

## Класс DisplayLightnessPwm

Граф наследования:DisplayLightnessPwm:

### Открытые члены

bool **process** () override

bool **init** () override

### Дополнительные унаследованные члены

Объявления и описания членов классов находятся в файлах:

src/DisplayLightnessPwm.h

src/DisplayLightnessPwm.cpp

## Шаблон структуры EnableBitMaskOperators< Enum >

### Статические открытые данные

static const bool **enable** = false

Объявления и описания членов структуры находятся в файле:

src/EnumExtension.h

## Класс Event

События.

#include <Event.h>

Граф наследования:Event:

### Открытые типы

enum **EventType** { **EventType::Quit**, **EventType::Touch**, **EventType::Timer**, **EventType::TreadmillConnected**, **EventType::TreadmillReady**, **EventType::TreadmillDisconnected**, **EventType::Refresh**, **EventType::Menu**, **EventType::ModalYes**, **EventType::ModalNo**, **EventType::ModalCancel**, **EventType::Zastup**, **EventType::EmergencyButton**, **EventType::KbdEvent**, **EventType::HrMonitorEvent**, **EventType::Shutdown**, **EventType::Restart** }*Типы событий.*

### Открытые члены

**Event** (**EventType** type, std::string extra={})

*Конструктор.*

virtual **~Event** ()

*Деструктор.*

**EventType** **getType** () const

*Получение типа события.*

std::string **getName** ()

*Получение название события.*

std::string **getExtra** ()

*Получение дополнительных данных.*

### Подробное описание

События.

### Перечисления

#### enum Event::EventType[strong]

Типы событий.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Quit | Выход. |
| Touch | Касание сенсора. |
| Timer | Событие таймера. |
| TreadmillConnected | Дорожка подключена. |
| TreadmillReady | Дорожка готова. |
| TreadmillDisconnected | Дорожка отключена. |
| Refresh | Обновление. |
| Menu | Отобразить меню. |
| ModalYes | Нажата кнопка ДА. |
| ModalNo | Нажата кнопка НЕТ. |
| ModalCancel | Нажата кнопка ОТМЕНА. |
| Zastup | Произошел заступ. |
| EmergencyButton | Нажата аварийная кнопка. |
| KbdEvent | Событие клавиатуры. |
| HrMonitorEvent | Событие монитора пульса. |
| Shutdown | Выключение. |
| Restart | Перезагрузка. |

### Конструктор(ы)

#### Event::Event (EventType *type*, std::string *extra* = {})

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *type* | тип события. |
| *extra* | дополнительные данные. |

### Методы

#### std::string Event::getExtra ()

Получение дополнительных данных.

##### Возвращает:

дополнительные данные.

#### std::string Event::getName ()

Получение название события.

##### Возвращает:

название события.

#### Event::EventType Event::getType () const

Получение типа события.

##### Возвращает:

тип события.

#### Объявления и описания членов классов находятся в файлах:

src/Event.h

src/Event.cpp

## Класс FreeRunMenu

Меню Свободный бег

#include <FreeRunMenu.h>

Граф наследования:FreeRunMenu:

### Открытые члены

**FreeRunMenu** ()

*Конструктор*

virtual **~FreeRunMenu** ()

*Деструктор*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Обработчик событий*

### Дополнительные унаследованные члены

### Подробное описание

Меню Свободный бег

Объявления и описания членов классов находятся в файлах:

src/FreeRunMenu.h

src/FreeRunMenu.cpp

## Класс GraphicEngine

Графический движок.

#include <GraphicEngine.h>

### Открытые члены

**GraphicEngine** ()

*Конструктор.*

virtual **~GraphicEngine** ()

*Деструктор.*

void **cairo\_linuxfb\_surface\_destroy** ()

*Функция обратного вызова для библиотеки Cairo.*

void **cairo\_png\_writer** (const unsigned char \*data, unsigned int length)

*Функция обратного вызова для библиотеки Cairo.*

void **drawSurface** (cairo\_surface\_t \*surface, int x=0, int y=0)

*Отобразить картинку.*

void **drawText** (std::string text, int x, int y, int size, float r, float g, float b, Align align=Align::Center, FontStyle style=FontStyle::Normal, float angle=0)

*Отображение одной строки текста.*

void **show** ()

*Отображение подготовленного изображения.*

void **drawRectangle** (int x, int y, int width, int height, uint32\_t fill\_color=DEC2RGB(100, 100, 100), int line\_width=0, uint32\_t line\_color=DEC2RGB(200, 200, 200), int xt=0, int yt=0, uint32\_t high\_color=DEC2RGB(200, 200, 200))

*Отображение прямоугольника.*

void **drawKbdButton** (const **KeyboardButton** &button)

*Отображение кнопки клавиатуры.*

void **drawMultiText** (std::string text, int x, int y, int width, int lines, int dy, int font\_size, Align h\_align=Align::Center, Align v\_align=Align::Center, FontStyle style=FontStyle::Normal, uint32\_t color=DEC2RGB(200, 200, 200), float angle=0)

*Отображение нескольких строк текста.*

**SizeGraph** **getTextSize** (std::string text, int font\_size, FontStyle style)

*Оценка размера выводимого текста.*

void **drawLine** (int x1, int y1, int x2, int y2, int line\_width=1, uint32\_t line\_color=DEC2RGB(200, 200, 200))

*Отображение линии.*

void **drawLineGradient** (int x1, int y1, int x2, int y2, int line\_width, std::vector< std::tuple< float, uint32\_t >> color)

*Отображение линии с градиентом.*

uint32\_t **getGradientColor** (uint32\_t beg, uint32\_t end, float index)

*Получение цвета градиента.*

### Подробное описание

Графический движок.

### Методы

#### void GraphicEngine::drawKbdButton (const KeyboardButton & *button*)

Отображение кнопки клавиатуры.

##### Аргументы:

|  |  |
| --- | --- |
| *button* | ссылка на кнопку. |

#### void GraphicEngine::drawLine (int *x1*, int *y1*, int *x2*, int *y2*, int *line\_width* = 1, uint32\_t *line\_color* = DEC2RGB(200,  200,  200))

Отображение линии.

##### Аргументы:

|  |  |
| --- | --- |
| *x1* | координата x начала. |
| *y1* | координата y начала. |
| *x2* | координата x конца. |
| *y2* | координата y конца. |
| *line\_width* | толщина линии. |
| *line\_color* | цвет линии. |

#### void GraphicEngine::drawLineGradient (int *x1*, int *y1*, int *x2*, int *y2*, int *line\_width*, std::vector< std::tuple< float, uint32\_t >> *color*)

Отображение линии с градиентом.

##### Аргументы:

|  |  |
| --- | --- |
| *x1* | координата x начала. |
| *y1* | координата y начала. |
| *x2* | координата x конца. |
| *y2* | координата y конца. |
| *line\_width* | толщина линии. |
| *color* | список цветов градиента. |

#### void GraphicEngine::drawMultiText (std::string *text*, int *x*, int *y*, int *width*, int *lines*, int *dy*, int *font\_size*, Align *h\_align* = Align::Center, Align *v\_align* = Align::Center, FontStyle *style* = FontStyle::Normal, uint32\_t *color* = DEC2RGB(200,  200,  200), float *angle* = 0)

Отображение нескольких строк текста.

##### Аргументы:

|  |  |
| --- | --- |
| *text* | текст для отображения. |
| *x* | координата x. |
| *y* | координата y. |
| *width* | ширина текста. |
| *lines* | количество отображаемых строк. |
| *dy* | промежуток между строками. |
| *font\_size* | размер шрифта. |
| *h\_align* | выравнивание по горизонтали. |
| *v\_align* | выравнивание по вертикали. |
| *style* | стиль. |
| *color* | цвет текста. |
| *angle* | угол поворота. |

#### void GraphicEngine::drawRectangle (int *x*, int *y*, int *width*, int *height*, uint32\_t *fill\_color* = DEC2RGB(100,  100,  100), int *line\_width* = 0, uint32\_t *line\_color* = DEC2RGB(200,  200,  200), int *xt* = 0, int *yt* = 0, uint32\_t *high\_color* = DEC2RGB(200,  200,  200))

Отображение прямоугольника.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |
| *width* | ширина. |
| *height* | высота. |
| *fill\_color* | цвет заполнения. |
| *line\_width* | толщина линии. |
| *line\_color* | цвет линии. |
| *xt* | координата x касания. |
| *yt* | координата y касания. |
| *high\_color* | цвет заполнения подсветки. |

#### void GraphicEngine::drawSurface (cairo\_surface\_t \* *surface*, int *x* = 0, int *y* = 0)

Отобразить картинку.

##### Аргументы:

|  |  |
| --- | --- |
| *surface* | ссылка на картинку. |
| *x* | координата х верхнего левого угла. |
| *y* | координата y верхнего левого угла. |

#### void GraphicEngine::drawText (std::string *text*, int *x*, int *y*, int *size*, float *r*, float *g*, float *b*, Align *align* = Align::Center, FontStyle *style* = FontStyle::Normal, float *angle* = 0)

Отображение одной строки текста.

##### Аргументы:

|  |  |
| --- | --- |
| *text* | текст для отображения. |
| *x* | координата x. |
| *y* | координата y. |
| *size* | размер шрифта. |
| *r* | красная компонента цвета. |
| *g* | зеленая компонента цвета. |
| *b* | синяя компонента цвета. |
| *align* | выравнивание. |
| *style* | стиль. |
| *angle* | угол поворота. |

#### uint32\_t GraphicEngine::getGradientColor (uint32\_t *beg*, uint32\_t *end*, float *index*)

Получение цвета градиента.

##### Аргументы:

|  |  |
| --- | --- |
| *beg* | первый цвет. |
| *end* | второй цвет. |
| *index* | точка градиента (0..1). |

##### Возвращает:

#### SizeGraph GraphicEngine::getTextSize (std::string *text*, int *font\_size*, FontStyle *style*)

Оценка размера выводимого текста.

##### Аргументы:

|  |  |
| --- | --- |
| *text* | текст для отображения. |
| *font\_size* | размер шрифта. |
| *style* | стиль. |

##### Возвращает:

размер текста.

#### Объявления и описания членов классов находятся в файлах:

src/GraphicEngine.h

src/GraphicEngine.cpp

## Класс HrMonitor

Драйвер монитора пульса.

#include <HrMonitor.h>

### Открытые члены

**HrMonitor** ()

*Конструктор.*

virtual **~HrMonitor** ()

*Деструктор.*

void **connect** (std::string path, bool rescan=true)

void **disconnect** ()

*Отключение подключенного устройства.*

float **getBattery** ()

*Получение уровня заряда батареи.*

float **getHRate** ()

*Получение значения пульса.*

bool **isConnected** ()

void **startDiscovery** ()

*Начать поиск устройств.*

void **stopDiscovery** ()

*Остановить поиск устройств.*

std::shared\_ptr< **BleDevice** > **getDeviceAt** (unsigned i)

*Получение устройства из списка обнаруженных.*

unsigned **getDeviceListSize** ()

*Получение размера списка устройств.*

void **clearStoredDevice** ()

*Сброс сохраненного в БД устройства.*

### Открытые атрибуты

std::string **m\_monitor\_path**

*Адрес текущего установленного устройства.*

### Подробное описание

Драйвер монитора пульса.

### Методы

#### void HrMonitor::connect (std::string *path*, bool *rescan* = true)

brief Подключение устройства.

##### Аргументы:

|  |  |
| --- | --- |
| *path* | адрес устройства. |
| *rescan* | выполнение предварительного сканирования (true: да, false: нет). |

#### std::shared\_ptr< BleDevice > HrMonitor::getDeviceAt (unsigned *i*)

Получение устройства из списка обнаруженных.

##### Аргументы:

|  |  |
| --- | --- |
| *i* | индекс. |

##### Возвращает:

ссылка на устройство.

#### unsigned HrMonitor::getDeviceListSize ()

Получение размера списка устройств.

##### Возвращает:

размер списка.

#### bool HrMonitor::isConnected ()

brief Получение состояния подключения устройства.

##### Возвращает:

true: подключено, false: отключено.

#### Объявления и описания членов классов находятся в файлах:

src/HrMonitor.h

src/HrMonitor.cpp

## Класс KbdEvent

Событие клавиатуры.

#include <KbdEvent.h>

Граф наследования:KbdEvent:

### Открытые члены

**KbdEvent** (std::string key="")

*Конструктор.*

virtual **~KbdEvent** ()

*Деструктор.*

const std::string & **getKey** () const

*Получение названия клавиши.*

### Дополнительные унаследованные члены

### Подробное описание

Событие клавиатуры.

### Конструктор(ы)

#### KbdEvent::KbdEvent (std::string *key* = "")

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *key* | название клавиши. |

### Методы

#### const std::string & KbdEvent::getKey () const

Получение названия клавиши.

##### Возвращает:

название клавиши.

#### Объявления и описания членов классов находятся в файлах:

src/KbdEvent.h

src/KbdEvent.cpp

## Класс KeyboardButton

Клавиша клавиатуры.

#include <KeyboardButton.h>

### Открытые члены

bool **into** (int xt, int yt)

*Определение нажатия.*

### Открытые атрибуты

int **x**

*Координата x.*

int **y**

*Координата y.*

int **width**

*Ширина.*

int **height**

*Высота.*

uint32\_t **fill\_color**

*Цвет заполнения.*

int **line\_width**

*Толщина линии.*

uint32\_t **line\_color**

*Цвет линии.*

uint32\_t **high\_fill\_color**

*Активный цвет заполнения.*

uint32\_t **high\_text\_color**

*Активный цвет текста.*

uint32\_t **text\_color**

*Цвет текста.*

int **text\_size**

*Размер текста.*

std::string **name**

*Название клавиши.*

bool **active**

*Флаг активности.*

### Подробное описание

Клавиша клавиатуры.

### Методы

#### bool KeyboardButton::into (int *xt*, int *yt*)

Определение нажатия.

##### Аргументы:

|  |  |
| --- | --- |
| *xt* | координата x касания. |
| *yt* | координата y касания. |

##### Возвращает:

true: есть нажатие, false: нет нажатия.

#### Объявления и описания членов классов находятся в файлах:

src/KeyboardButton.h

src/KeyboardButton.cpp

## Класс Locale

Локализатор.

#include <Locale.h>

### Открытые члены

std::string **tr** (std::string str, std::string extra="")

*Выполнение локализации.*

void **init** (std::string locale)

*Инициализация.*

std::string **getLocale** ()

*Получение языка локализации.*

### Открытые статические члены

static **Locale** & **instance** ()

*Получение объекта.*

### Подробное описание

Локализатор.

### Методы

#### std::string Locale::getLocale ()

Получение языка локализации.

##### Возвращает:

язык локализации.

#### void Locale::init (std::string *locale*)

Инициализация.

##### Аргументы:

|  |  |
| --- | --- |
| *locale* | язык локализации. |

#### std::string Locale::tr (std::string *str*, std::string *extra* = "")

Выполнение локализации.

##### Аргументы:

|  |  |
| --- | --- |
| *str* | ключевой текст. |
| *extra* | extra вторичный ключ. |

##### Возвращает:

локализованная строка.

#### Объявления и описания членов классов находятся в файлах:

src/Locale.h

src/Locale.cpp

## Класс Logger

Работа с лог файлом.

#include <Logger.h>

### Открытые типы

enum **LogLevel** { **LogLevel::Error**, **LogLevel::Info**, **LogLevel::Trace**, **LogLevel::Debug** }*Уровни сообщений*

### Открытые члены

void **log** (std::string msg, **LogLevel** level=**LogLevel::Info**)

*Основная функция для вывода в лог.*

void **setLogLevel** (std::string level)

*Установка уровня сообщений.*

void **flush** ()

*Сброс буферов на диск.*

### Открытые статические члены

static **Logger** & **instance** (std::string app\_name="Application", LogLevel level=**LogLevel::Info**, bool console\_flag=true)

*Получение объекта.*

### Подробное описание

Работа с лог файлом.

### Перечисления

#### enum Logger::LogLevel[strong]

Уровни сообщений

##### Элементы перечислений:

|  |  |
| --- | --- |
| Error | Информация об ошибках. |
| Info | Общая информация. |
| Trace | Расширенная информация. |
| Debug | Отладочная информация. |

### Методы

#### Logger & Logger::instance (std::string *app\_name* = "Application", LogLevel *level* = LogLevel::Info, bool *console\_flag* = true)[static]

Получение объекта.

##### Аргументы:

|  |  |
| --- | --- |
| *app\_name* | название приложения. |
| *level* | уровень сообщений. |
| *console\_flag* | флаг вывода на консоль. |

##### Возвращает:

объект класса.

#### void Logger::log (std::string *msg*, LogLevel *level* = LogLevel::Info)

Основная функция для вывода в лог.

##### Аргументы:

|  |  |
| --- | --- |
| *msg* | текст сообщения. |
| *level* | уровень информации в сообщении. |

#### void Logger::setLogLevel (std::string *level*)

Установка уровня сообщений.

##### Аргументы:

|  |  |
| --- | --- |
| *level* | уровень сообщений. |

#### Объявления и описания членов классов находятся в файлах:

src/Logger.h

src/Logger.cpp

## Класс MainLoop

Главный цикл приложения.

#include <MainLoop.h>

### Открытые члены

**MainLoop** ()

*Конструктор.*

virtual **~MainLoop** ()

*Деструктор.*

int **loop** ()

*Функция цикла.*

### Подробное описание

Главный цикл приложения.

### Методы

#### int MainLoop::loop ()

Функция цикла.

##### Возвращает:

код возврата.

#### Объявления и описания членов классов находятся в файлах:

src/MainLoop.h

src/MainLoop.cpp

## Класс MainMenu

Главное меню приложения.

#include <MainMenu.h>

Граф наследования:MainMenu:

### Открытые члены

**MainMenu** ()

*Конструктор.*

virtual **~MainMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Главное меню приложения.

### Методы

#### Menu \* MainMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/MainMenu.h

src/MainMenu.cpp

## Класс Menu

Базовый класс меню.

#include <Menu.h>

Граф наследования:Menu:

### Открытые члены

**Menu** (int x=0, int y=0, std::string background="", WidgetSPVec &&widgets={})

*Конструктор.*

virtual **Menu** \* **processEvent** (std::unique\_ptr< **Event** > event)

*Функция обработки событий.*

virtual void **show** (bool final=true)

*Отрисовка изображения.*

WidgetSP **getSelected** (int x, int y)

*Поиск виджета по координатам.*

int **getX** () const

*Получение координаты x меню.*

void **setX** (int x)

*Установка координаты x меню.*

int **getY** () const

*Получение координаты y меню.*

void **setY** (int y)

*Установка координаты y меню.*

const std::string & **getType** () const

*Получение типа меню.*

void **setType** (const std::string &type)

*Установка типа меню.*

int **getWidth** () const

*Получение ширины меню.*

int **getHeight** () const

*Получение высоты меню.*

void **setPatient** (std::string name, int id, float weight, int profile\_id)

*Установка текущего пациента.*

template<class T > std::shared\_ptr< T > **addWidget** (std::shared\_ptr< T > widget)

*Добавление виджета.*

### Открытые атрибуты

std::unique\_ptr< **Menu** > **m\_modal**

*Указатель на модальное меню*

### Защищенные типы

enum **EventType** { **Stop**, **Pause**, **None** }

### Защищенные члены

EventType **checkEvents** ()

void **addBreadCrumps** (std::vector< **WBCElement** > &&bread\_crumps)

void **addOnBackAction** (std::function< std::unique\_ptr<::**Event** >(**WidgetImageButton::Event**)> on\_back)

### Защищенные данные

int **m\_background\_id**

WidgetSPVec **m\_widgets**

WidgetSP **m\_active\_widget**

int **m\_x**

int **m\_y**

std::string **m\_type**

bool **m\_check\_events**

int **m\_width**

int **m\_height**

EventType **m\_event**

WidgetImageButtonSP **m\_patients\_btn**

WidgetImageButtonSP **m\_hr\_btn**

WidgetImageButtonSP **m\_back\_btn**

std::function< std::unique\_ptr<::**Event** >**WidgetImageButton::Event**)> **m\_on\_back**

### Подробное описание

Базовый класс меню.

### Конструктор(ы)

#### Menu::Menu (int *x* = 0, int *y* = 0, std::string *background* = "", WidgetSPVec && *widgets* = {})

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |
| *background* | фоновая картинка. |
| *widgets* | набор виджетов. |

### Методы

#### template<class T > std::shared\_ptr<T> Menu::addWidget (std::shared\_ptr< T > *widget*)[inline]

Добавление виджета.

##### Аргументы:

|  |  |
| --- | --- |
| *widget* | ссылка на виджет. |

##### Возвращает:

ссылка на добавленный виджет.

#### int Menu::getHeight () const

Получение высоты меню.

##### Возвращает:

высота меню.

#### WidgetSP Menu::getSelected (int *x*, int *y*)

Поиск виджета по координатам.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |

##### Возвращает:

ссылка на виджет.

#### const std::string & Menu::getType () const

Получение типа меню.

##### Возвращает:

тип меню.

#### int Menu::getWidth () const

Получение ширины меню.

##### Возвращает:

ширина меню.

#### int Menu::getX () const

Получение координаты x меню.

##### Возвращает:

координата x.

#### int Menu::getY () const

Получение координаты y меню.

##### Возвращает:

координата y.

#### Menu \* Menu::processEvent (std::unique\_ptr< Event > *event*)[virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяется в **ModalKeyboardAlpha** (*стр.82*), **ModalKeyboardDigital** (*стр.85*), **RehabUserProtocolsViewMenu** (*стр.120*), **PatientNewMenu** (*стр.96*), **RehabUserProtocolsMenu** (*стр.118*), **RehabBosBaseMenu** (*стр.104*), **RehabUserProtocolsEditMenu** (*стр.116*), **PatientEditMenu** (*стр.92*), **PatientListMenu** (*стр.94*), **RehabTestProtocolMenu** (*стр.114*), **PatientHistoryMenu** (*стр.93*), **RehabProfileRunMenu** (*стр.109*), **SettingsHrMenu** (*стр.137*), **ModalAuthAdmin** (*стр.75*), **SettingsCalibrateMenu** (*стр.132*), **SettingsExtraCalibActuatorMenu** (*стр.135*), **SettingsPasswordMenu** (*стр.139*), **SettingsRegistersMenu** (*стр.140*), **SettingsClockMenu** (*стр.133*), **SettingsExportMenu** (*стр.134*), **MainMenu** (*стр.66*) и **FreeRunMenu** (*стр.50*).

#### void Menu::setPatient (std::string *name*, int *id*, float *weight*, int *profile\_id*)

Установка текущего пациента.

##### Аргументы:

|  |  |
| --- | --- |
| *name* | имя пациента. |
| *id* | идентификатор. |
| *weight* | вес. |

#### void Menu::setType (const std::string & *type*)

Установка типа меню.

##### Аргументы:

|  |  |
| --- | --- |
| *type* | тип меню. |

#### void Menu::setX (int *x*)

Установка координаты x меню.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |

#### void Menu::setY (int *y*)

Установка координаты y меню.

##### Аргументы:

|  |  |
| --- | --- |
| *y* | координата y. |

#### void Menu::show (bool *final* = true)[virtual]

Отрисовка изображения.

##### Аргументы:

|  |  |
| --- | --- |
| *final* | true: финальная отрисовка на экране, false: промежуточная отрисовка в буфере. |

Переопределяется в **ModalKeyboardAlpha** (*стр.83*) и **ModalKeyboardDigital** (*стр.85*).

#### Объявления и описания членов классов находятся в файлах:

src/Menu.h

src/Menu.cpp

## Класс MenuEvent

Событие перехода на меню.

#include <MenuEvent.h>

Граф наследования:MenuEvent:

### Открытые члены

**MenuEvent** (std::unique\_ptr< **Menu** > menu)

*Конструктор.*

virtual **~MenuEvent** ()

*Деструктор.*

const std::unique\_ptr< **Menu** > & **getMenu** () const

### Дополнительные унаследованные члены

### Подробное описание

Событие перехода на меню.

### Конструктор(ы)

#### MenuEvent::MenuEvent (std::unique\_ptr< Menu > *menu*)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *menu* | ссылка на меню. |

### Методы

#### const std::unique\_ptr< Menu > & MenuEvent::getMenu () const

Получение указателя на меню.

##### Возвращает:

указатель на меню.

#### Объявления и описания членов классов находятся в файлах:

src/MenuEvent.h

src/MenuEvent.cpp

## Класс ModalAuthAdmin

Диалог ввода пароля доктора/администратора.

#include <ModalAuthAdmin.h>

Граф наследования:ModalAuthAdmin:

### Открытые члены

**ModalAuthAdmin** ()

*Конструктор.*

virtual **~ModalAuthAdmin** ()

*Деструктор.*

**Menu** \* **processEvent** (EventUP event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Диалог ввода пароля доктора/администратора.

### Методы

#### Menu \* ModalAuthAdmin::processEvent (EventUP *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/ModalAuthAdmin.h

src/ModalAuthAdmin.cpp

## Класс ModalAuthDoctor

Диалог ввода пароля доктора.

#include <ModalAuthDoctor.h>

Граф наследования:ModalAuthDoctor:

### Открытые члены

**ModalAuthDoctor** ()

*Конструктор.*

virtual **~ModalAuthDoctor** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Диалог ввода пароля доктора.

Объявления и описания членов классов находятся в файлах:

src/ModalAuthDoctor.h

src/ModalAuthDoctor.cpp

## Класс ModalCommonDialog

Универсальный диалог.

#include <ModalCommonDialog.h>

Граф наследования:ModalCommonDialog:

### Открытые типы

enum **Flags** : unsigned { **Flags::GreenTitle** = 1, **Flags::YellowTitle** = 2, **Flags::NoSound** = 8, **Flags::OkButton** = 0x10, **Flags::CloseButton** = 0x20, **Flags::YesCancel** = 0x40, **Flags::YesNoCancel** = 0x80, **Flags::Yes** = 0x100, **Flags::StopIcon** = 0x1000, **Flags::Buttons** = 0xFF0, **Flags::Icons** = 0xF000 }*Флаги.*

### Открытые члены

**ModalCommonDialog** (std::string title, std::string message, std::string type, **Flags** flags, std::string icon\_file)

*Конструктор.*

virtual **~ModalCommonDialog** ()

*Деструктор.*

void **setTimer** (int period, TimerFunc on\_timer, int id=0)

*Установка таймера.*

void **setText** (std::string text)

*Установка текста сообщения.*

### Дополнительные унаследованные члены

### Подробное описание

Универсальный диалог.

### Перечисления

#### enum ModalCommonDialog::Flags : unsigned[strong]

Флаги.

##### Элементы перечислений:

|  |  |
| --- | --- |
| GreenTitle | Зеленый заголовок. |
| YellowTitle | Желтый заголовок. |
| NoSound | Не использовать звук. |
| OkButton | Кнопка ОК. |
| CloseButton | Кнопка ЗАКРЫТЬ. |
| YesCancel | Кнопки ДА, ОТМЕНА. |
| YesNoCancel | Кнопки ДА, НЕТ, ОТМЕНА. |
| Yes | Кнопка ДА. |
| StopIcon | Иконка СТОП. |
| Buttons | Маска КНОПКИ. |
| Icons | Маска ИКОНКИ. |

### Конструктор(ы)

#### ModalCommonDialog::ModalCommonDialog (std::string *title*, std::string *message*, std::string *type*, Flags *flags*, std::string *icon\_file*)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *title* | заголовок диалога. |
| *message* | текст сообщения. |
| *type* | тип. |
| *flags* | флаги. |
| *icon\_file* | иконка. |

### Методы

#### void ModalCommonDialog::setText (std::string *text*)

Установка текста сообщения.

##### Аргументы:

|  |  |
| --- | --- |
| *text* | текст сообщения. |

#### void ModalCommonDialog::setTimer (int *period*, TimerFunc *on\_timer*, int *id* = 0)

Установка таймера.

##### Аргументы:

|  |  |
| --- | --- |
| *period* | длительность периода, мс. |
| *on\_timer* | функция обработчик события таймера. |
| *id* | идентификатор таймера. |

#### Объявления и описания членов классов находятся в файлах:

src/ModalCommonDialog.h

src/ModalCommonDialog.cpp

## Класс ModalCommonError

Окно сообщения об ошибке.

#include <ModalCommonError.h>

Граф наследования:ModalCommonError:

### Открытые члены

**ModalCommonError** (std::string title, std::string message, std::string type)

*Конструктор.*

virtual **~ModalCommonError** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Окно сообщения об ошибке.

### Конструктор(ы)

#### ModalCommonError::ModalCommonError (std::string *title*, std::string *message*, std::string *type*)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *title* | заголовок окна. |
| *message* | текст сообщения. |
| *type* | тип. |

#### Объявления и описания членов классов находятся в файлах:

src/ModalCommonError.h

src/ModalCommonError.cpp

## Класс ModalKeyboardAlpha

Алфавитно-цифровая клавиатура.

#include <ModalKeyboardAlpha.h>

Граф наследования:ModalKeyboardAlpha:

### Открытые типы

enum **Layout** { **Layout::RusSmall**, **Layout::RusCapital**, **Layout::EngSmall**, **Layout::EngCapital**, **Layout::Digits**, **Layout::Symbols** }*Раскладка.*

enum **Position** { **Position::Top**, **Position::Bottom** }*Расположение.*

### Открытые члены

**ModalKeyboardAlpha** (**Position** pos=**Position::Bottom**)

*Конструктор.*

virtual **~ModalKeyboardAlpha** ()

*Деструктор.*

void **show** (bool final=true) override

*Отрисовка изображения.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

void **setLayout** (**Layout** layout)

*Установка раскладки.*

### Дополнительные унаследованные члены

### Подробное описание

Алфавитно-цифровая клавиатура.

### Перечисления

#### enum ModalKeyboardAlpha::Layout[strong]

Раскладка.

##### Элементы перечислений:

|  |  |
| --- | --- |
| RusSmall | Кириллица строчные. |
| RusCapital | Кириллица прописные. |
| EngSmall | Английская строчные. |
| EngCapital | Английская прописные. |
| Digits | Цифры. |
| Symbols | Символы. |

#### enum ModalKeyboardAlpha::Position[strong]

Расположение.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Top | Вверху. |
| Bottom | Внизу. |

### Конструктор(ы)

#### ModalKeyboardAlpha::ModalKeyboardAlpha (Position *pos* = Position::Bottom)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *pos* | расположение. |

### Методы

#### Menu \* ModalKeyboardAlpha::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### void ModalKeyboardAlpha::setLayout (Layout *layout*)

Установка раскладки.

##### Аргументы:

|  |  |
| --- | --- |
| *layout* | раскладка. |

#### void ModalKeyboardAlpha::show (bool *final* = true)[override], [virtual]

Отрисовка изображения.

##### Аргументы:

|  |  |
| --- | --- |
| *final* | true: финальная отрисовка на экране, false: промежуточная отрисовка в буфере. |

Переопределяет метод предка **Menu** (*стр.72*).

#### Объявления и описания членов классов находятся в файлах:

src/ModalKeyboardAlpha.h

src/ModalKeyboardAlpha.cpp

## Класс ModalKeyboardDigital

Цифровая клавиатура.

#include <ModalKeyboardDigital.h>

Граф наследования:ModalKeyboardDigital:

### Открытые типы

enum **Layout** { **Layout::Integer**, **Layout::Fixed** }*Раскладка.*

### Открытые члены

**ModalKeyboardDigital** (int x=0, int y=0, std::string prompt="Введите число", Layout layout=**Layout::Integer**, bool minus\_sign=false)

*Конструктор.*

virtual **~ModalKeyboardDigital** ()

*Деструктор.*

void **show** (bool final=true) override

*Отрисовка изображения.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

void **updateSize** ()

*Пересчет положения.*

### Дополнительные унаследованные члены

### Подробное описание

Цифровая клавиатура.

### Перечисления

#### enum ModalKeyboardDigital::Layout[strong]

Раскладка.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Integer | Целые. |
| Fixed | С плавающей точкой. |

### Конструктор(ы)

#### ModalKeyboardDigital::ModalKeyboardDigital (int *x* = 0, int *y* = 0, std::string *prompt* = "Введите  число", Layout *layout* = Layout::Integer, bool *minus\_sign* = false)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |
| *prompt* | подсказка. |
| *layout* | раскладка. |
| *minus\_sign* | наличие знака минус (true: присутствует, false: отсутствует). |

### Методы

#### Menu \* ModalKeyboardDigital::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### void ModalKeyboardDigital::show (bool *final* = true)[override], [virtual]

Отрисовка изображения.

##### Аргументы:

|  |  |
| --- | --- |
| *final* | true: финальная отрисовка на экране, false: промежуточная отрисовка в буфере. |

Переопределяет метод предка **Menu** (*стр.72*).

#### Объявления и описания членов классов находятся в файлах:

src/ModalKeyboardDigital.h

src/ModalKeyboardDigital.cpp

## Класс ModalListSelector

Окно выбора из списка.

#include <ModalListSelector.h>

Граф наследования:ModalListSelector:

### Открытые члены

**ModalListSelector** (int x, int y, DataSet options, std::string active\_pic, std::string passive\_pic, uint32\_t color\_text\_active=0x3C3C3C, uint32\_t color\_text=0x3C3C3C, int font\_size=24)

*Конструктор.*

virtual **~ModalListSelector** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Окно выбора из списка.

### Конструктор(ы)

#### ModalListSelector::ModalListSelector (int *x*, int *y*, DataSet *options*, std::string *active\_pic*, std::string *passive\_pic*, uint32\_t *color\_text\_active* = 0x3C3C3C, uint32\_t *color\_text* = 0x3C3C3C, int *font\_size* = 24)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |
| *options* | элементы списка. |
| *active\_pic* | фон активного элемента. |
| *passive\_pic* | фон пассивного элемента. |
| *color\_text\_active* | цвет текста активного элемента. |
| *color\_text* | цвет текста пассивного элемента. |
| *font\_size* | размер шрифта. |

#### Объявления и описания членов классов находятся в файлах:

src/ModalListSelector.h

src/ModalListSelector.cpp

## Класс ModalPatientHistoryFilter

Окно установки фильтра истории.

#include <ModalPatientHistoryFilter.h>

Граф наследования:ModalPatientHistoryFilter:

### Открытые члены

**ModalPatientHistoryFilter** ()

*Конструктор.*

virtual **~ModalPatientHistoryFilter** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Окно установки фильтра истории.

Объявления и описания членов классов находятся в файлах:

src/ModalPatientHistoryFilter.h

src/ModalPatientHistoryFilter.cpp

## Класс ModalWarmUpParams

Окно установки параметров разминки.

#include <ModalWarmUpParams.h>

Граф наследования:ModalWarmUpParams:

### Открытые члены

**ModalWarmUpParams** ()

*Конструктор.*

virtual **~ModalWarmUpParams** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Окно установки параметров разминки.

Объявления и описания членов классов находятся в файлах:

src/ModalWarmUpParams.h

src/ModalWarmUpParams.cpp

## Класс Parameter

Работа с параметрами.

#include <Parameter.h>

Граф наследования:Parameter:

### Открытые члены

**Parameter** (std::map< std::string, std::string > default\_values={})

*Конструктор.*

void **setParamString** (std::string name, std::string value)

*Установка параметра.*

std::string **getParamString** (std::string name)

float **getParamFloat** (std::string name)

int **getParamInt** (std::string name)

### Защищенные данные

std::map< std::string, std::string > **m\_parameters**

bool **m\_changed** {false}

### Подробное описание

Работа с параметрами.

### Конструктор(ы)

#### Parameter::Parameter (std::map< std::string, std::string > *default\_values* = {})[explicit]

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *default\_values* | значения параметров по умолчанию. |

### Методы

#### float Parameter::getParamFloat (std::string *name*)

Получение параметра в виде числа с плавающей точкой.

##### Аргументы:

|  |  |
| --- | --- |
| *name* | имя параметра. |

##### Возвращает:

значение параметра.

#### int Parameter::getParamInt (std::string *name*)

Получение параметра в виде целого числа.

##### Аргументы:

|  |  |
| --- | --- |
| *name* | имя параметра. |

##### Возвращает:

значение параметра.

#### std::string Parameter::getParamString (std::string *name*)

Получение параметра в виде строки.

##### Аргументы:

|  |  |
| --- | --- |
| *name* | имя параметра. |

##### Возвращает:

значение параметра.

#### void Parameter::setParamString (std::string *name*, std::string *value*)

Установка параметра.

##### Аргументы:

|  |  |
| --- | --- |
| *name* | имя параметра. |
| *value* | значение параметра. |

#### Объявления и описания членов классов находятся в файлах:

src/Parameter.h

src/Parameter.cpp

## Класс PatientDetailsMenu

Меню "детальная иформация о пациенте".

#include <PatientDetailsMenu.h>

Граф наследования:PatientDetailsMenu:

### Открытые члены

**PatientDetailsMenu** ()

*Конструктор.*

virtual **~PatientDetailsMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "детальная иформация о пациенте".

### Конструктор(ы)

#### PatientDetailsMenu::PatientDetailsMenu ()

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | идентификатор пациента. |

#### Объявления и описания членов классов находятся в файлах:

src/PatientDetailsMenu.h

src/PatientDetailsMenu.cpp

## Класс PatientEditMenu

Меню "редактор базы пациентов".

#include <PatientEditMenu.h>

Граф наследования:PatientEditMenu:

### Открытые члены

**PatientEditMenu** ()

*Конструктор.*

virtual **~PatientEditMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "редактор базы пациентов".

### Методы

#### Menu \* PatientEditMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/PatientEditMenu.h

src/PatientEditMenu.cpp

## Класс PatientHistoryMenu

Меню "история пациентов".

#include <PatientHistoryMenu.h>

Граф наследования:PatientHistoryMenu:

### Открытые члены

**PatientHistoryMenu** ()

*Конструктор.*

virtual **~PatientHistoryMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "история пациентов".

### Методы

#### Menu \* PatientHistoryMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/PatientHistoryMenu.h

src/PatientHistoryMenu.cpp

## Класс PatientListMenu

Меню "база пациентов".

#include <PatientListMenu.h>

Граф наследования:PatientListMenu:

### Открытые члены

**PatientListMenu** ()

*Конструктор.*

virtual **~PatientListMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "база пациентов".

### Методы

#### Menu \* PatientListMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/PatientListMenu.h

src/PatientListMenu.cpp

## Класс PatientNewMenu

Меню "создания/редактирования учетной записи пациента".

#include <PatientNewMenu.h>

Граф наследования:PatientNewMenu:

### Открытые типы

enum **Mode** { **Mode::New**, **Mode::Edit** }*Режим.*

### Открытые члены

**PatientNewMenu** ()

*Конструктор.*

virtual **~PatientNewMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Открытые статические члены

static void **setProfile** (std::string profile\_id, std::string profile\_name)

*Установка профиля пациента.*

static std::string **getProfile** ()

*Получение профиля пациента.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "создания/редактирования учетной записи пациента".

### Перечисления

#### enum PatientNewMenu::Mode[strong]

Режим.

##### Элементы перечислений:

|  |  |
| --- | --- |
| New | Создание. |
| Edit | Редактирование. |

### Конструктор(ы)

#### PatientNewMenu::PatientNewMenu ()

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | идентификатор пациента. |
| *mode* | режим. |

### Методы

#### std::string PatientNewMenu::getProfile ()[static]

Получение профиля пациента.

##### Возвращает:

имя профиля.

#### Menu \* PatientNewMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### void PatientNewMenu::setProfile (std::string *profile\_id*, std::string *profile\_name*)[static]

Установка профиля пациента.

##### Аргументы:

|  |  |
| --- | --- |
| *profile\_id* | идентификатор профиля. |
| *profile\_name* | имя профиля. |

#### Объявления и описания членов классов находятся в файлах:

src/PatientNewMenu.h

src/PatientNewMenu.cpp

## Класс PatientSearchMenu

Меню "поиск пациентов".

#include <PatientSearchMenu.h>

Граф наследования:PatientSearchMenu:

### Открытые члены

**PatientSearchMenu** ()

*Конструктор.*

virtual **~PatientSearchMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "поиск пациентов".

Объявления и описания членов классов находятся в файлах:

src/PatientSearchMenu.h

src/PatientSearchMenu.cpp

## Класс PatientStatsMenu

Меню "статистика по учетной записи пациента".

#include <PatientStatsMenu.h>

Граф наследования:PatientStatsMenu:

### Открытые члены

**PatientStatsMenu** ()

*Конструктор.*

virtual **~PatientStatsMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "статистика по учетной записи пациента".

Объявления и описания членов классов находятся в файлах:

src/PatientStatsMenu.h

src/PatientStatsMenu.cpp

## Класс PersistedParameter

Сохраняемые параметры.

#include <PersistedParameter.h>

Граф наследования:PersistedParameter:

### Открытые члены

**PersistedParameter** (std::string file\_name, std::map< std::string, std::string > default\_values={})

*Конструктор.*

virtual **~PersistedParameter** ()

*Деструктор.*

void **save** ()

*Сохранение параметров.*

### Дополнительные унаследованные члены

### Подробное описание

Сохраняемые параметры.

### Конструктор(ы)

#### PersistedParameter::PersistedParameter (std::string *file\_name*, std::map< std::string, std::string > *default\_values* = {})[explicit]

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *file\_name* | имя файла для хранения параметров. |
| *default\_values* | значения по умолчанию. |

#### Объявления и описания членов классов находятся в файлах:

src/PersistedParameter.h

src/PersistedParameter.cpp

## Класс PictureManager

Управление картинками.

#include <PictureManager.h>

### Открытые члены

**PictureManager** ()

*Конструктор.*

virtual **~PictureManager** ()

*Деструктор.*

int **loadPicture** (std::string file)

*Загрузка картинки.*

cairo\_surface\_t \* **getPicture** (int id)

### Подробное описание

Управление картинками.

### Методы

#### cairo\_surface\_t \* PictureManager::getPicture (int *id*)

Получение картинки.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | идентификатор. |

##### Возвращает:

ссылка на картинку.

#### int PictureManager::loadPicture (std::string *file*)

Загрузка картинки.

##### Аргументы:

|  |  |
| --- | --- |
| *file* | имя файла. |

##### Возвращает:

идентификатор картинки, 0 в в случае ошибки.

#### Объявления и описания членов классов находятся в файлах:

src/PictureManager.h

src/PictureManager.cpp

## Класс ProceduresMenu

Граф наследования:ProceduresMenu:

### Дополнительные унаследованные члены

Объявления и описания членов классов находятся в файлах:

src/ProceduresMenu.h

src/ProceduresMenu.cpp

## Класс RehabBosBaseMenu

Базовый класс для меню бега с биологической обратной связью.

#include <RehabBosBaseMenu.h>

Граф наследования:RehabBosBaseMenu:

### Классы

struct **BosCorrectionData**

### *Структура данных коррекции.* Открытые типы

enum **Mode** { **MinMax**, **Power** }*Список режимов работы.*

### Открытые члены

**RehabBosBaseMenu** (**Mode** mode)

*Конструктор.*

virtual **~RehabBosBaseMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Защищенные члены

virtual void **update** ()

*Обновление времени.*

### Защищенные данные

WidgetTextFieldSP **m\_max\_possible\_hr\_text**

*Максимальное значение ЧСС.*

int **m\_total\_time**

*Общее время процедуры.*

int **m\_warm\_time**

*Время разминки.*

**WTFStyle** **m\_field\_style**

*Стиль оформления полей ввода.*

**WTFStyle** **m\_label\_style**

*Стиль оформления названий.*

**WTFStyle** **m\_unit\_style**

*Стиль оформления единиц измерения.*

WidgetTextFieldSP **m\_load\_control**

*Элемент управления "Управление нагрузкой".*

**Mode** **m\_mode**

*Режим работы.*

WidgetTextFieldSP **m\_proc\_time\_text**

*Виджет поля ввода времени процедуры.*

WidgetTextFieldSP **m\_proc\_time\_label**

*Виджет заголовка времени процедуры.*

WidgetTextFieldSP **m\_proc\_time\_unit\_label**

*Виджет единиц измерения времени процедуры.*

WidgetTextFieldSP **m\_angle\_label**

*Виджет заголовка кнопки режима управления по углу.*

WidgetTextFieldSP **m\_speed\_label**

*Виджет заголовка кнопки режима управления по скорости.*

WidgetTextFieldSP **m\_speed\_angle\_label**

*Виджет заголовка кнопки режима управления по скорости и углу.*

WidgetImageButtonSP **m\_speed\_angle\_btn**

*Виджет кнопки режима управления по скорости и углу.*

WidgetImageButtonSP **m\_angle\_btn**

*Виджет кнопки режима управления по углу.*

WidgetImageButtonSP **m\_speed\_btn**

*Виджет кнопки режима управления по скорости.*

WidgetImageButtonSP **m\_extra\_line**

*Виджет дополнительного ряда данных.*

WidgetTextFieldSP **m\_max\_speed\_text**

*Виджет поля ввода максимальной скорости.*

WidgetTextFieldSP **m\_max\_angle\_text**

*Виджет поля ввода максимального угла.*

int **m\_proc\_time**

*Длительность процедуры.*

### Дополнительные унаследованные члены

### Подробное описание

Базовый класс для меню бега с биологической обратной связью.

### Конструктор(ы)

#### RehabBosBaseMenu::RehabBosBaseMenu (Mode *mode*)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *mode* | режим работы. |

### Методы

#### Menu \* RehabBosBaseMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/RehabBosBaseMenu.h

src/RehabBosBaseMenu.cpp

## Класс RehabBosHrMinMaxMenu

Меню "бег с БОС задание ЧСС мин/макс".

#include <RehabBosHrMinMaxMenu.h>

Граф наследования:RehabBosHrMinMaxMenu:

### Открытые члены

**RehabBosHrMinMaxMenu** ()

*Конструктор.*

virtual **~RehabBosHrMinMaxMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "бег с БОС задание ЧСС мин/макс".

Объявления и описания членов классов находятся в файлах:

src/RehabBosHrMinMaxMenu.h

src/RehabBosHrMinMaxMenu.cpp

## Класс RehabBosHrPowerMenu

Меню "бег с БОС задание по ЧСС макс/мощности".

#include <RehabBosHrPowerMenu.h>

Граф наследования:RehabBosHrPowerMenu:

### Открытые члены

**RehabBosHrPowerMenu** ()

*Конструктор.*

virtual **~RehabBosHrPowerMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "бег с БОС задание по ЧСС макс/мощности".

Объявления и описания членов классов находятся в файлах:

src/RehabBosHrPowerMenu.h

src/RehabBosHrPowerMenu.cpp

## Класс RehabExternalMenu

Меню "тестирование с внешней стресс-системой".

#include <RehabExternalMenu.h>

Граф наследования:RehabExternalMenu:

### Открытые члены

**RehabExternalMenu** ()

*Конструктор.*

virtual **~RehabExternalMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "тестирование с внешней стресс-системой".

Объявления и описания членов классов находятся в файлах:

src/RehabExternalMenu.h

src/RehabExternalMenu.cpp

## Класс RehabMenu

Меню "выбор типа процедур".

#include <RehabMenu.h>

Граф наследования:RehabMenu:

### Открытые члены

**RehabMenu** ()

*Конструктор.*

virtual **~RehabMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "выбор типа процедур".

Объявления и описания членов классов находятся в файлах:

src/RehabMenu.h

src/RehabMenu.cpp

## Класс RehabProfileRunMenu

Меню "бег по профилю".

#include <RehabProfileRunMenu.h>

Граф наследования:RehabProfileRunMenu:

### Открытые члены

**RehabProfileRunMenu** ()

*Конструктор.*

virtual **~RehabProfileRunMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "бег по профилю".

### Методы

#### Menu \* RehabProfileRunMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/RehabProfileRunMenu.h

src/RehabProfileRunMenu.cpp

## Класс RehabProtocolSelectionMenu

Меню "выбор протокола тестирования".

#include <RehabProtocolSelectionMenu.h>

Граф наследования:RehabProtocolSelectionMenu:

### Открытые члены

**RehabProtocolSelectionMenu** ()

*Конструктор.*

virtual **~RehabProtocolSelectionMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "выбор протокола тестирования".

Объявления и описания членов классов находятся в файлах:

src/RehabProtocolSelectionMenu.h

src/RehabProtocolSelectionMenu.cpp

## Класс RehabSelHrMenu

Меню "выбор процедуры с БОС".

#include <RehabSelHrMenu.h>

Граф наследования:RehabSelHrMenu:

### Открытые члены

**RehabSelHrMenu** ()

*Конструктор.*

virtual **~RehabSelHrMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "выбор процедуры с БОС".

Объявления и описания членов классов находятся в файлах:

src/RehabSelHrMenu.h

src/RehabSelHrMenu.cpp

## Класс RehabSelMethodMenu

Меню "выбор метода тестирования".

#include <RehabSelMethodMenu.h>

Граф наследования:RehabSelMethodMenu:

### Открытые члены

**RehabSelMethodMenu** ()

*Конструктор.*

virtual **~RehabSelMethodMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "выбор метода тестирования".

Объявления и описания членов классов находятся в файлах:

src/RehabSelMethodMenu.h

src/RehabSelMethodMenu.cpp

## Класс RehabSelProfileMenu

Меню "выбор бега по профилю".

#include <RehabSelProfileMenu.h>

Граф наследования:RehabSelProfileMenu:

### Открытые члены

**RehabSelProfileMenu** ()

*Конструктор.*

virtual **~RehabSelProfileMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "выбор бега по профилю".

Объявления и описания членов классов находятся в файлах:

src/RehabSelProfileMenu.h

src/RehabSelProfileMenu.cpp

## Класс RehabTestProtocolMenu

Меню "тестирование по протоколу ".

#include <RehabTestProtocolMenu.h>

Граф наследования:RehabTestProtocolMenu:

### Открытые члены

**RehabTestProtocolMenu** ()

*Конструктор.*

virtual **~RehabTestProtocolMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "тестирование по протоколу ".

### Методы

#### Menu \* RehabTestProtocolMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/RehabTestProtocolMenu.h

src/RehabTestProtocolMenu.cpp

## Класс RehabUserProfilSelectMenu

Граф наследования:RehabUserProfilSelectMenu:

### Дополнительные унаследованные члены

Объявления и описания членов класса находятся в файле:

src/RehabUserProfilSelectMenu.h

## Класс RehabUserProtocolsEditMenu

Меню "редактирование профиля/протокола".

#include <RehabUserProtocolsEditMenu.h>

Граф наследования:RehabUserProtocolsEditMenu:

### Открытые члены

**RehabUserProtocolsEditMenu** ()

*Конструктор.*

virtual **~RehabUserProtocolsEditMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "редактирование профиля/протокола".

### Методы

#### Menu \* RehabUserProtocolsEditMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/RehabUserProtocolsEditMenu.h

src/RehabUserProtocolsEditMenu.cpp

## Класс RehabUserProtocolsMenu

Меню "пользовательские профили/протоколы".

#include <RehabUserProtocolsMenu.h>

Граф наследования:RehabUserProtocolsMenu:

### Открытые типы

enum **ProMode** { **ProMode::Protocol**, **ProMode::Profil**, **ProMode::ProfilSelect** }*Режим.*

### Открытые члены

**RehabUserProtocolsMenu** ()

*Конструктор.*

virtual **~RehabUserProtocolsMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "пользовательские профили/протоколы".

### Перечисления

#### enum RehabUserProtocolsMenu::ProMode[strong]

Режим.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Protocol | Работа с протоколами. |
| Profil | Работа с профилями. |
| ProfilSelect | Выбор профиля для пациента. |

### Методы

#### Menu \* RehabUserProtocolsMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/RehabUserProtocolsMenu.h

src/RehabUserProtocolsMenu.cpp

## Класс RehabUserProtocolsViewMenu

Меню "просмотр/создание/редактирование профиля/протокола".

#include <RehabUserProtocolsViewMenu.h>

Граф наследования:RehabUserProtocolsViewMenu:

### Открытые типы

enum **Mode** { **Mode::View**, **Mode::New**, **Mode::Edit** }*Режим.*

### Открытые члены

**RehabUserProtocolsViewMenu** ()

*Конструктор.*

virtual **~RehabUserProtocolsViewMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "просмотр/создание/редактирование профиля/протокола".

### Перечисления

#### enum RehabUserProtocolsViewMenu::Mode[strong]

Режим.

##### Элементы перечислений:

|  |  |
| --- | --- |
| View | Просмотр. |
| New | Создание. |
| Edit | Редактирование. |

### Конструктор(ы)

#### RehabUserProtocolsViewMenu::RehabUserProtocolsViewMenu ()

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | идентификатор протокола/профиля. |
| *mode* | режим просмотр/создание/редактирование. |
| *pro\_mode* | выбор профиль/протокол. |

### Методы

#### Menu \* RehabUserProtocolsViewMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/RehabUserProtocolsViewMenu.h

src/RehabUserProtocolsViewMenu.cpp

## Класс RunSession

Беговая сессия.

#include <RunSession.h>

### Открытые члены

**RunSession** ()

*Конструктор.*

virtual **~RunSession** ()

*Деструктор.*

void **start** (const int patient\_id, const std::string profile)

*Начало сессии.*

void **finish** ()

*Завершение сессии.*

void **incSec** ()

*Увеличение времени сессии на 1 секунду.*

int **getDuration** () const

*Получение длительности сессии.*

float **getDistance** () const

*Получение пройденного расстояния.*

float **getEnergy** () const

*Получение затраченной энергии.*

float **getPowerAero** () const

*Получение аэробной мощности.*

float **getPowerMech** () const

*Получение механической мощности.*

float **getAngle** () const

*Получение текущего угла наклона.*

float **getSpeed** () const

*Получение скорости дорожки.*

std::string **getDistanceString** (float dist=-1) const

*Получение расстояния в виде строки.*

std::string **getAngleString** (float angle=-1, std::string format="") const

*Получение угла наклона в виде строки.*

std::string **getSpeedString** (float speed=-1, std::string format="") const

*Получение скорости в виде строки.*

std::string **getEnergyString** (float energy=-1) const

*Получение энергии в виде строки.*

std::string **getPowerAeroString** (float const value=-1) const

*Получение аэробной мощности в виде строки.*

std::string **getPowerMechString** (float const value=-1) const

*Получение механической мощности в виде строки.*

std::string **getSpeedFormat** () const

*Получение форматной строки для скорости.*

std::string **getAngleFormat** () const

*Получение форматной строки для угла.*

std::string **getPowerMechFormat** () const

*Получение форматной строки для механической мощности.*

std::string **getAccelString** (float accel=-1) const

*Получение ускорения в виде строки.*

float **getSpeedKmH** (const float value)

*Получение скорости в км/ч.*

float **getAngleDegree** (const float value)

*Получение угла в градусах.*

float **getPowerWatt** (const float value)

*Получение мощности в ваттах.*

int **getHrate** () const

*Получение текущего значения пульса.*

int **getPatientId** () const

*Получение идентификатора текущего пациента.*

std::string **getProfile** () const

*Получение названия вида бега текущей сессии.*

float **getSpeedAvg** () const

*Получение средней скорости.*

float **getAccelMax** () const

*Получение максимального ускорения.*

float **getAngleMax** () const

*Получение максимального угла наклона.*

int **getHrateMin** () const

*Получение минимального значения пульса.*

int **getHrateMax** () const

*Получение максимального значения пульса.*

float **getPowerAeroAvg** () const

*Получение средней аэробной мощности.*

float **getPowerMechAvg** () const

*Получение средней механической мощности.*

float **getHrateAvg** () const

*Получение среднего значения пульса.*

float **getAcceleration** () const

*Получение текущего ускорения.*

float **getPowerMechByParams** (float speed, float angle, float weight, **TreadmillDriver::Direction** direction=**TreadmillDriver::Direction::Forward**)

*Получение мощности по параметрам.*

### Подробное описание

Беговая сессия.

### Методы

#### float RunSession::getAcceleration () const

Получение текущего ускорения.

##### Возвращает:

текущее ускорение.

#### float RunSession::getAccelMax () const

Получение максимального ускорения.

##### Возвращает:

максимальное ускорение.

#### std::string RunSession::getAccelString (float *accel* = -1) const

Получение ускорения в виде строки.

##### Аргументы:

|  |  |
| --- | --- |
| *accel* | ускорение. |

##### Возвращает:

отформатированная строка.

#### float RunSession::getAngle () const

Получение текущего угла наклона.

##### Возвращает:

угол наклона в градусах.

#### float RunSession::getAngleDegree (const float *value*)

Получение угла в градусах.

##### Аргументы:

|  |  |
| --- | --- |
| *value* | угол. |

##### Возвращает:

угол в градусах.

#### std::string RunSession::getAngleFormat () const

Получение форматной строки для угла.

##### Возвращает:

форматная строка.

#### float RunSession::getAngleMax () const

Получение максимального угла наклона.

##### Возвращает:

максимальный угол наклонаю

#### std::string RunSession::getAngleString (float *angle* = -1, std::string *format* = "") const

Получение угла наклона в виде строки.

##### Аргументы:

|  |  |
| --- | --- |
| *angle* | угол наклона. |
| *format* | формат отображения. |

##### Возвращает:

отформатированная строка.

#### float RunSession::getDistance () const

Получение пройденного расстояния.

##### Возвращает:

пройденное расстояние в км.

#### std::string RunSession::getDistanceString (float *dist* = -1) const

Получение расстояния в виде строки.

##### Аргументы:

|  |  |
| --- | --- |
| *dist* | расстояние. |

##### Возвращает:

отформатированная строка.

#### int RunSession::getDuration () const

Получение длительности сессии.

##### Возвращает:

длительность сессии в секундах.

#### float RunSession::getEnergy () const

Получение затраченной энергии.

##### Возвращает:

затраченная энергия в джоулях.

#### std::string RunSession::getEnergyString (float *energy* = -1) const

Получение энергии в виде строки.

##### Аргументы:

|  |  |
| --- | --- |
| *energy* | энергия. |

##### Возвращает:

отформатированная строка.

#### int RunSession::getHrate () const

Получение текущего значения пульса.

##### Возвращает:

пульс.

#### float RunSession::getHrateAvg () const

Получение среднего значения пульса.

##### Возвращает:

средний пульс.

#### int RunSession::getHrateMax () const

Получение максимального значения пульса.

##### Возвращает:

максимальный пульс.

#### int RunSession::getHrateMin () const

Получение минимального значения пульса.

##### Возвращает:

минимальный пульс.

#### int RunSession::getPatientId () const

Получение идентификатора текущего пациента.

##### Возвращает:

идентификатор пациента.

#### float RunSession::getPowerAero () const

Получение аэробной мощности.

##### Возвращает:

аэробная мощность в мл/мин/кг.

#### float RunSession::getPowerAeroAvg () const

Получение средней аэробной мощности.

##### Возвращает:

средняя аэробная мощность.

#### std::string RunSession::getPowerAeroString (float const *value* = -1) const

Получение аэробной мощности в виде строки.

##### Аргументы:

|  |  |
| --- | --- |
| *value* | аэробная мощность. |

##### Возвращает:

отформатированная строка.

#### float RunSession::getPowerMech () const

Получение механической мощности.

##### Возвращает:

механическая мощность в ваттах.

#### float RunSession::getPowerMechAvg () const

Получение средней механической мощности.

##### Возвращает:

средняя механическая мощность.

#### float RunSession::getPowerMechByParams (float *speed*, float *angle*, float *weight*, TreadmillDriver::Direction *direction* = TreadmillDriver::Direction::Forward)

Получение мощности по параметрам.

##### Аргументы:

|  |  |
| --- | --- |
| *speed* | скорость. |
| *angle* | угол. |
| *weight* | вес пациента. |
| *direction* | направление движения. |

##### Возвращает:

мощность в ваттах.

#### std::string RunSession::getPowerMechFormat () const

Получение форматной строки для механической мощности.

##### Возвращает:

форматная строка.

#### std::string RunSession::getPowerMechString (float const *value* = -1) const

Получение механической мощности в виде строки.

##### Аргументы:

|  |  |
| --- | --- |
| *value* | механическая мощность. |

##### Возвращает:

отформатированная строка.

#### float RunSession::getPowerWatt (const float *value*)

Получение мощности в ваттах.

##### Аргументы:

|  |  |
| --- | --- |
| *value* | мощность. |

##### Возвращает:

мощность в ваттах.

#### std::string RunSession::getProfile () const

Получение названия вида бега текущей сессии.

##### Возвращает:

название вида бега.

#### float RunSession::getSpeed () const

Получение скорости дорожки.

##### Возвращает:

скорость дорожки в км/ч.

#### float RunSession::getSpeedAvg () const

Получение средней скорости.

##### Возвращает:

средняя скорость.

#### std::string RunSession::getSpeedFormat () const

Получение форматной строки для скорости.

##### Возвращает:

форматная строка.

#### float RunSession::getSpeedKmH (const float *value*)

Получение скорости в км/ч.

##### Аргументы:

|  |  |
| --- | --- |
| *value* | скорость. |

##### Возвращает:

скорость в км/ч.

#### std::string RunSession::getSpeedString (float *speed* = -1, std::string *format* = "") const

Получение скорости в виде строки.

##### Аргументы:

|  |  |
| --- | --- |
| *speed* | скорость. |
| *format* | формат отображения. |

##### Возвращает:

отформатированная строка.

#### void RunSession::start (const int *patient\_id*, const std::string *profile*)

Начало сессии.

##### Аргументы:

|  |  |
| --- | --- |
| *patient\_id* | идентификатор пациента. |
| *profile* | название вида бега. |

#### Объявления и описания членов классов находятся в файлах:

src/RunSession.h

src/RunSession.cpp

## Класс SettingsAccelMenu

Меню "задание ускорения и торможения".

#include <SettingsAccelMenu.h>

Граф наследования:SettingsAccelMenu:

### Открытые члены

**SettingsAccelMenu** ()

*Конструктор.*

virtual **~SettingsAccelMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "задание ускорения и торможения".

Объявления и описания членов классов находятся в файлах:

src/SettingsAccelMenu.h

src/SettingsAccelMenu.cpp

## Класс SettingsBosParamsMenu

Меню "Настройка параметров режимов с БОС".

#include <SettingsBosParamsMenu.h>

Граф наследования:SettingsBosParamsMenu:

### Открытые члены

**SettingsBosParamsMenu** ()

*Конструктор.*

virtual **~SettingsBosParamsMenu** ()

*Деструктор.*

### Открытые статические члены

static std::vector< std::string > **getParamsLow** ()

*Вектор значений по умолчанию нижней границы.*

static std::vector< std::string > **getParamsHigh** ()

*Вектор значений по умолчанию верхней границы.*

static float **getSpeed** (std::vector< std::string > data, int row)

*Получение дельты скорости из таблицы параметров.*

static float **getAngle** (std::vector< std::string > data, int row)

*Получение дельты угла из таблицы параметров.*

static float **getTime** (std::vector< std::string > data, int row)

*Получение длительности интервала из таблицы параметров.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "Настройка параметров режимов с БОС".

### Методы

#### float SettingsBosParamsMenu::getAngle (std::vector< std::string > *data*, int *row*)[static]

Получение дельты угла из таблицы параметров.

##### Аргументы:

|  |  |
| --- | --- |
| *data* | вектор значений. |
| *row* | индекс ряда. |

##### Возвращает:

значение дельты угла.

#### float SettingsBosParamsMenu::getSpeed (std::vector< std::string > *data*, int *row*)[static]

Получение дельты скорости из таблицы параметров.

##### Аргументы:

|  |  |
| --- | --- |
| *data* | вектор значений. |
| *row* | индекс ряда. |

##### Возвращает:

значение дельты скорости.

#### float SettingsBosParamsMenu::getTime (std::vector< std::string > *data*, int *row*)[static]

Получение длительности интервала из таблицы параметров.

##### Аргументы:

|  |  |
| --- | --- |
| *data* | вектор значений. |
| *row* | индекс ряда. |

##### Возвращает:

значение длительности интервала.

#### Объявления и описания членов классов находятся в файлах:

src/SettingsBosParamsMenu.h

src/SettingsBosParamsMenu.cpp

## Класс SettingsCalibrateMenu

Меню "калибровка положения беговой дорожки".

#include <SettingsCalibrateMenu.h>

Граф наследования:SettingsCalibrateMenu:

### Открытые члены

**SettingsCalibrateMenu** ()

*Конструктор.*

virtual **~SettingsCalibrateMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "калибровка положения беговой дорожки".

### Методы

#### Menu \* SettingsCalibrateMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/SettingsCalibrateMenu.h

src/SettingsCalibrateMenu.cpp

## Класс SettingsClockMenu

Меню "настройка времени и даты".

#include <SettingsClockMenu.h>

Граф наследования:SettingsClockMenu:

### Открытые члены

**SettingsClockMenu** ()

*Конструктор.*

virtual **~SettingsClockMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "настройка времени и даты".

### Методы

#### Menu \* SettingsClockMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/SettingsClockMenu.h

src/SettingsClockMenu.cpp

## Класс SettingsExportMenu

Меню "импорт и экспорт данных".

#include <SettingsExportMenu.h>

Граф наследования:SettingsExportMenu:

### Открытые члены

**SettingsExportMenu** ()

*Конструктор.*

virtual **~SettingsExportMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "импорт и экспорт данных".

### Методы

#### Menu \* SettingsExportMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/SettingsExportMenu.h

src/SettingsExportMenu.cpp

## Класс SettingsExtraCalibActuatorMenu

Меню "калибровка подъемника".

#include <SettingsExtraCalibActuatorMenu.h>

Граф наследования:SettingsExtraCalibActuatorMenu:

### Открытые члены

**SettingsExtraCalibActuatorMenu** ()

*Конструктор.*

virtual **~SettingsExtraCalibActuatorMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "калибровка подъемника".

### Методы

#### Menu \* SettingsExtraCalibActuatorMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/SettingsExtraCalibActuatorMenu.h

src/SettingsExtraCalibActuatorMenu.cpp

## Класс SettingsExtraCalibDriveMenu

Меню "калибровка привода".

#include <SettingsExtraCalibDriveMenu.h>

Граф наследования:SettingsExtraCalibDriveMenu:

### Открытые члены

**SettingsExtraCalibDriveMenu** ()

*Конструктор.*

virtual **~SettingsExtraCalibDriveMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "калибровка привода".

Объявления и описания членов классов находятся в файлах:

src/SettingsExtraCalibDriveMenu.h

src/SettingsExtraCalibDriveMenu.cpp

## Класс SettingsHrMenu

Меню "подключение/отключение датчика ЧСС".

#include <SettingsHrMenu.h>

Граф наследования:SettingsHrMenu:

### Открытые члены

**SettingsHrMenu** ()

*Конструктор.*

virtual **~SettingsHrMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "подключение/отключение датчика ЧСС".

### Методы

#### Menu \* SettingsHrMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/SettingsHrMenu.h

src/SettingsHrMenu.cpp

## Класс SettingsMenu

Меню "настройки".

#include <SettingsMenu.h>

Граф наследования:SettingsMenu:

### Открытые члены

**SettingsMenu** (std::string user={})

*Конструктор.*

virtual **~SettingsMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "настройки".

Объявления и описания членов классов находятся в файлах:

src/SettingsMenu.h

src/SettingsMenu.cpp

## Класс SettingsPasswordMenu

Меню "смена пароля врача".

#include <SettingsPasswordMenu.h>

Граф наследования:SettingsPasswordMenu:

### Открытые члены

**SettingsPasswordMenu** ()

*Конструктор.*

virtual **~SettingsPasswordMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "смена пароля врача".

### Методы

#### Menu \* SettingsPasswordMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/SettingsPasswordMenu.h

src/SettingsPasswordMenu.cpp

## Класс SettingsRegistersMenu

Меню "системная информация".

#include <SettingsRegistersMenu.h>

Граф наследования:SettingsRegistersMenu:

### Открытые члены

**SettingsRegistersMenu** ()

*Конструктор.*

virtual **~SettingsRegistersMenu** ()

*Деструктор.*

**Menu** \* **processEvent** (std::unique\_ptr< **Event** > event) override

*Функция обработки событий.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "системная информация".

### Методы

#### Menu \* SettingsRegistersMenu::processEvent (std::unique\_ptr< Event > *event*)[override], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

Переопределяет метод предка **Menu** (*стр.71*).

#### Объявления и описания членов классов находятся в файлах:

src/SettingsRegistersMenu.h

src/SettingsRegistersMenu.cpp

## Класс SettingsUnitsMenu

Меню "единицы измерения физических величин".

#include <SettingsUnitsMenu.h>

Граф наследования:SettingsUnitsMenu:

### Открытые члены

**SettingsUnitsMenu** ()

*Конструктор.*

virtual **~SettingsUnitsMenu** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Меню "единицы измерения физических величин".

Объявления и описания членов классов находятся в файлах:

src/SettingsUnitsMenu.h

src/SettingsUnitsMenu.cpp

## Структура SizeGraph

Размер контура.

#include <GraphicEngine.h>

### Открытые атрибуты

int **width**

*Ширина.*

int **height**

*Высота.*

### Подробное описание

Размер контура.

Объявления и описания членов структуры находятся в файле:

src/GraphicEngine.h

## Класс SoundInterface

Управление звуком.

#include <SoundInterface.h>

### Открытые типы

enum **Sound** { **Sound::Silent** = -1, **Sound::Click** = 0, **Sound::Welcome**, **Sound::Alarm**, **Sound::Bell**, **Sound::Info** }*Вид звука.*

### Открытые члены

**SoundInterface** ()

*Конструктор.*

virtual **~SoundInterface** ()

*Деструктор.*

void **play** (**Sound** sound)

### Подробное описание

Управление звуком.

### Перечисления

#### enum SoundInterface::Sound[strong]

Вид звука.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Silent | Тишина. |
| Click | Щелчок. |
| Welcome | Приветствие. |
| Alarm | Предупреждение. |
| Bell | Предупреждение ЧСС. |
| Info | Информация. |

### Методы

#### void SoundInterface::play (Sound *sound*)

Воспроизвести звук.

##### Аргументы:

|  |  |
| --- | --- |
| *sound* | тип звука. |

#### Объявления и описания членов классов находятся в файлах:

src/SoundInterface.h

src/SoundInterface.cpp

## Шаблон класса SyncQueue< A >

Синхронизованная очередь.

#include <SyncQueue.h>

### Открытые члены

**SyncQueue** (std::string m\_name="")

*Конструктор.*

virtual **~SyncQueue** ()

*Деструктор.*

void **putEvent** (A event)

*Поместить событие в конец очереди.*

void **putAsNext** (A event)

*Поместить событие в начало очереди.*

A **getEvent** ()

*Получить событие из очереди.*

int **getSize** ()

*Получить размер очереди.*

void **clear** ()

*очистить очередь.*

### Подробное описание

#### template<class A>

#### class SyncQueue< A >

Синхронизованная очередь.

### Методы

#### template<class A > A SyncQueue< A >::getEvent ()

Получить событие из очереди.

##### Возвращает:

событие.

#### template<class A > int SyncQueue< A >::getSize ()

Получить размер очереди.

##### Возвращает:

размер очереди.

#### template<class A> void SyncQueue< A >::putAsNext (A *event*)

Поместить событие в начало очереди.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

#### template<class A> void SyncQueue< A >::putEvent (A *event*)

Поместить событие в конец очереди.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

#### Объявления и описания членов класса находятся в файле:

src/SyncQueue.h

## Класс ThreadClass

Базовый класс управления потоком.

#include <ThreadClass.h>

Граф наследования:ThreadClass:

### Открытые члены

**ThreadClass** ()

*Конструктор.*

virtual **~ThreadClass** ()

*Деструктор.*

### Защищенные члены

virtual bool **process** ()=0

virtual bool **init** ()

void **start** ()

void **stop** ()

### Подробное описание

Базовый класс управления потоком.

Объявления и описания членов классов находятся в файлах:

src/ThreadClass.h

src/ThreadClass.cpp

## Класс Timer

Реализация периодического таймера.

#include <Timer.h>

### Открытые члены

**Timer** (TimerFunc on\_timer, int id=0)

*Конструктор.*

virtual **~Timer** ()

*Деструктор.*

void **startTimerMillis** (int delay)

*Запуск таймера.*

void **stopTimer** ()

*Остановка таймера.*

### Подробное описание

Реализация периодического таймера.

### Конструктор(ы)

#### Timer::Timer (TimerFunc *on\_timer*, int *id* = 0)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *on\_timer* | функция обработчик события таймера. |
| *id* | идентификатор таймера. |

### Методы

#### void Timer::startTimerMillis (int *delay*)

Запуск таймера.

##### Аргументы:

|  |  |
| --- | --- |
| *delay* | период таймара, мс. |

#### Объявления и описания членов классов находятся в файлах:

src/Timer.h

src/Timer.cpp

## Класс TouchEvent

Событие сенсорной панели.

#include <TouchEvent.h>

Граф наследования:TouchEvent:

### Открытые типы

enum **TouchType** { **TouchType::Touch**, **TouchType::Untouch**, **TouchType::Move**, **TouchType::Clear** }*Тип события.*

### Открытые члены

**TouchEvent** (**TouchType** type, int x, int y)

*Конструктор.*

virtual **~TouchEvent** ()

*Деструктор.*

**TouchType** **getType** () const

*Получение типа события.*

int **getX** () const

*Получение координаты x.*

int **getY** () const

*Получение координаты y.*

### Подробное описание

Событие сенсорной панели.

### Перечисления

#### enum TouchEvent::TouchType[strong]

Тип события.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Touch | Прикосновение. |
| Untouch | Отрыв. |
| Move | Перемещение. |
| Clear | Очистка. |

### Конструктор(ы)

#### TouchEvent::TouchEvent (TouchType *type*, int *x*, int *y*)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *type* | тип события. |
| *x* | координата x. |
| *y* | координата y. |

### Методы

#### TouchEvent::TouchType TouchEvent::getType () const

Получение типа события.

##### Возвращает:

тип события.

#### int TouchEvent::getX () const

Получение координаты x.

##### Возвращает:

координата x.

#### int TouchEvent::getY () const

Получение координаты y.

##### Возвращает:

координата y.

#### Объявления и описания членов классов находятся в файлах:

src/TouchEvent.h

src/TouchEvent.cpp

## Класс TouchScreenDriver

Драйвер сенсорной панели.

#include <TouchScreenDriver.h>

### Открытые члены

**TouchScreenDriver** ()

*Конструктор.*

virtual **~TouchScreenDriver** ()

*Деструктор.*

### Подробное описание

Драйвер сенсорной панели.

Объявления и описания членов классов находятся в файлах:

src/TouchScreenDriver.h

src/TouchScreenDriver.cpp

## Класс TreadmillDriver

Драйвер дорожки.

#include <TreadmillDriver.h>

Граф наследования:TreadmillDriver:

### Открытые типы

enum **RunState** { **RunState::Stop**, **RunState::Run**, **RunState::Pause** }*Состояние дорожки.*

enum **Direction** { **Direction::Forward**, **Direction::Backward** }*Направление движения.*

enum **Lift** { **Lift::Stop**, **Lift::Up**, **Lift::Down** }

enum **ExtMode** { **ExtMode::On** = 1, **ExtMode::Off** = 2 }*Управление внешней стресс-системой.*

enum **Reg** { **Reg::PLC\_ID** = 0, **Reg::PLC\_STAT** = 1, **Reg::PLC\_CS** = 2, **Reg::PLC\_CD** = 3, **Reg::PLC\_CTA** = 4, **Reg::PLC\_CTD** = 5, **Reg::PLC\_SD** = 6, **Reg::PLC\_ED** = 7, **Reg::PLC\_CL** = 8, **Reg::PLC\_AX** = 9, **Reg::PLC\_ACX** = 10, **Reg::PLC\_ACY** = 11, **Reg::PLC\_CAL** = 12, **Reg::PLC\_EX** = 78 }*Регистры.*

### Открытые члены

**TreadmillDriver** ()

*Конструктор.*

virtual **~TreadmillDriver** ()

*Деструктор.*

void **log** (char level, std::string msg)

int **getAcceleration** () const

*Получение текущего ускорения.*

void **setAcceleration** (int acceleration)

*Установка ускорения.*

int **getAngle** () const

*Получение угла наклона.*

void **setAngle** (int angle, bool not\_set=false)

*Установка угла наклона.*

int **getDeceleration** () const

*Получение текущего значения торможения.*

void **setDeceleration** (int deceleration)

*Установка текущего значения торможения.*

**Direction** **getDirection** () const

*Получение текущего направления движения.*

void **setDirection** (**Direction** direction)

*Установка текущего направления движения.*

int **getSpeed** () const

*Получение текущей скорости движения.*

void **setSpeed** (int speed)

*Установка текущей скорости движения.*

**RunState** **getState** () const

*Получение состояния дорожки.*

void **setState** (**RunState** state)

*Установка состояния дорожки.*

void **toggleMode** ()

*Переключение в режим калибровки и обратно.*

bool **isReady** () const

*Определение готовности дорожки.*

void **setLift** (**Lift** lift)

*Установка подъемника в режиме пульта.*

void **setRegister** (**Reg** reg, int value)

*Запись в регистр дорожки.*

int **getRegister** (**Reg** reg)

*Чтение регистра дорожки.*

void **setRemoteControlLifting** (bool enable)

*Включение/выключение режима пульта.*

bool **isCrossError** () const

*Проверка наличия ошибки заступа.*

bool **isEmegencyStop** () const

*Проверка состояния аварийной кнопки.*

bool **isPlcConnected** ()

*Проверка наличия связи с дорожкой.*

bool **isPFreqAlarm** ()

*Проверка работы преобразователя частоты.*

bool **isLifterAlarm** ()

*Проверка наличия связи с подъемником.*

bool **isCalibrating** ()

*Проверка состояния калибровки.*

void **resetError** ()

*Сброс ошибок.*

int **getMaxForwardSpeed** ()

*Получение максимальной скорости дорожки вперед.*

int **getMaxBackwardSpeed** ()

*Получение максимальной скорости дорожки назад.*

int **getMinSpeed** ()

*Получение минимальной скорости дорожки.*

int **getMaxAngle** ()

*Получение максимального угла подъема.*

int **getSpeedStep** ()

*Получение шага изменения скорости.*

int **getCalibrationAngleX** ()

*Получение угла калибровки по оси X.*

int **getCalibrationAngleY** ()

*Получение угла калибровки по оси Y.*

void **setExternal** (**ExtMode** mode)

*Установка режима работы с внешней стресс-системой.*

uint16\_t **getRegisterState** (**Reg** name)

*Получение значения регистра.*

int **getActualSpeed** () const

*Получение реальной скорости дорожки.*

bool **isZeroPoint** () const

*Проверка, что дорожка находится в исходном состоянии.*

void **doZeroPoint** ()

*Перевод дорожки в исходное состояние.*

void **setDNF** (bool value)

*Управление частотой калибровки скорости.*

void **setSpeedCalib** (unsigned short value)

*Установка калибровочного коэффициента.*

void **actuatorCalibReset** ()

*Сброс калибровки подъемника.*

void **actuatorCalibSet** ()

*Установка калибровки подъемника.*

bool **isSettingPoint** ()

*Проверка готовности подъемника.*

### Открытые статические члены

static int **getAccelValue** (const int id)

*Получение значения ускорения по уровню.*

static int **getAccelTimeValue** (const int id)

*Получение времени разгона/торможения по уровню.*

### Дополнительные унаследованные члены

### Подробное описание

Драйвер дорожки.

### Перечисления

#### enum TreadmillDriver::Direction[strong]

Направление движения.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Forward | Вперед. |
| Backward | Назад. |

#### enum TreadmillDriver::ExtMode[strong]

Управление внешней стресс-системой.

##### Элементы перечислений:

|  |  |
| --- | --- |
| On | Включена. |
| Off | Отключена. |

#### enum TreadmillDriver::Lift[strong]

##### Элементы перечислений:

|  |  |
| --- | --- |
| Stop | Остановлен. |
| Up | Подъем. |
| Down | Опускание. |

#### enum TreadmillDriver::Reg[strong]

Регистры.

##### Элементы перечислений:

|  |  |
| --- | --- |
| PLC\_ID | Идентификатор. |
| PLC\_STAT | Статус системы. |
| PLC\_CS | Регистр управления системой. |
| PLC\_CD | Регистр управления приводом. |
| PLC\_CTA | Регистр задания времени ускорения. |
| PLC\_CTD | Регистр задания времени торможения. |
| PLC\_SD | Регистр состояния привода. |
| PLC\_ED | Регистр ошибки привода. |
| PLC\_CL | Регистр управления подъемником. |
| PLC\_AX | Регистр данных угла. |
| PLC\_ACX | Регистр калибровочных данных угла Х. |
| PLC\_ACY | Регистр калибровочных данных угла Y. |
| PLC\_CAL | Регистр калибровки скорости. |
| PLC\_EX | Регистр управления внешней стресс системой. |

#### enum TreadmillDriver::RunState[strong]

Состояние дорожки.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Stop | Остановлена. |
| Run | Бег. |
| Pause | Пауза. |

### Методы

#### int TreadmillDriver::getAcceleration () const

Получение текущего ускорения.

##### Возвращает:

ускорение.

#### int TreadmillDriver::getAccelTimeValue (const int *id*)[static]

Получение времени разгона/торможения по уровню.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | уровень. |

##### Возвращает:

время разгона/торможения в секундах.

#### int TreadmillDriver::getAccelValue (const int *id*)[static]

Получение значения ускорения по уровню.

##### Аргументы:

|  |  |
| --- | --- |
| *id* | уровень ускорения (1-7). |

##### Возвращает:

значение ускорения.

#### int TreadmillDriver::getActualSpeed () const

Получение реальной скорости дорожки.

##### Возвращает:

реальная скорость дорожки.

#### int TreadmillDriver::getAngle () const

Получение угла наклона.

##### Возвращает:

угол наклона.

#### int TreadmillDriver::getCalibrationAngleX ()

Получение угла калибровки по оси X.

##### Возвращает:

угол калибровки.

#### int TreadmillDriver::getCalibrationAngleY ()

Получение угла калибровки по оси Y.

##### Возвращает:

угол калибровки.

#### int TreadmillDriver::getDeceleration () const

Получение текущего значения торможения.

##### Возвращает:

значение торможения.

#### TreadmillDriver::Direction TreadmillDriver::getDirection () const

Получение текущего направления движения.

##### Возвращает:

текущее направление движения.

#### int TreadmillDriver::getMaxAngle ()

Получение максимального угла подъема.

##### Возвращает:

максимальный угол подъема.

#### int TreadmillDriver::getMaxBackwardSpeed ()

Получение максимальной скорости дорожки назад.

##### Возвращает:

максимальная скорость дорожки назад.

#### int TreadmillDriver::getMaxForwardSpeed ()

Получение максимальной скорости дорожки вперед.

##### Возвращает:

максимальная скорость дорожки вперед.

#### int TreadmillDriver::getMinSpeed ()

Получение минимальной скорости дорожки.

##### Возвращает:

минимальная скорость дорожки.

#### int TreadmillDriver::getRegister (Reg *reg*)

Чтение регистра дорожки.

##### Аргументы:

|  |  |
| --- | --- |
| *reg* | номер регистра. |

##### Возвращает:

значение.

#### uint16\_t TreadmillDriver::getRegisterState (Reg *name*)

Получение значения регистра.

##### Аргументы:

|  |  |
| --- | --- |
| *name* | идентификатор регистра. |

##### Возвращает:

значение регистра.

#### int TreadmillDriver::getSpeed () const

Получение текущей скорости движения.

##### Возвращает:

текущая скорость движения.

#### int TreadmillDriver::getSpeedStep ()

Получение шага изменения скорости.

##### Возвращает:

шаг изменения скорости.

#### TreadmillDriver::RunState TreadmillDriver::getState () const

Получение состояния дорожки.

##### Возвращает:

состояние дорожки.

#### bool TreadmillDriver::isCalibrating ()

Проверка состояния калибровки.

##### Возвращает:

true: режим калибровки включен, false: калибровка выключена.

#### bool TreadmillDriver::isCrossError () const

Проверка наличия ошибки заступа.

##### Возвращает:

true: есть ошибка, false: ошибки нет.

#### bool TreadmillDriver::isEmegencyStop () const

Проверка состояния аварийной кнопки.

##### Возвращает:

true: кнопка нажата, false: кнопка не нажата.

#### bool TreadmillDriver::isLifterAlarm ()

Проверка наличия связи с подъемником.

##### Возвращает:

true: связь есть, false: связи нет.

#### bool TreadmillDriver::isPFreqAlarm ()

Проверка работы преобразователя частоты.

##### Возвращает:

true: преобразователь работает, false: преобразователь не работает

#### bool TreadmillDriver::isPlcConnected ()

Проверка наличия связи с дорожкой.

##### Возвращает:

true: связь есть, false: связи нет.

#### bool TreadmillDriver::isReady () const

Определение готовности дорожки.

##### Возвращает:

true: дорожка готова.

#### bool TreadmillDriver::isSettingPoint ()

Проверка готовности подъемника.

##### Возвращает:

true: подъемник готов, false: подъемник в процессе работы.

#### bool TreadmillDriver::isZeroPoint () const

Проверка, что дорожка находится в исходном состоянии.

##### Возвращает:

true: исходное состояние, false: не в исходном состоянии.

#### void TreadmillDriver::log (char *level*, std::string *msg*)

Функция записи в лог файл для библиотеки Modbus.

##### Аргументы:

|  |  |
| --- | --- |
| *level* | уровень сообщения. |
| *msg* | текст сообщения. |

#### void TreadmillDriver::setAcceleration (int *acceleration*)

Установка ускорения.

##### Аргументы:

|  |  |
| --- | --- |
| *acceleration* | ускорение. |

#### void TreadmillDriver::setAngle (int *angle*, bool *not\_set* = false)

Установка угла наклона.

##### Аргументы:

|  |  |
| --- | --- |
| *angle* | угол наклона. |

#### void TreadmillDriver::setDeceleration (int *deceleration*)

Установка текущего значения торможения.

##### Аргументы:

|  |  |
| --- | --- |
| *deceleration* | значение торможения. |

#### void TreadmillDriver::setDirection (Direction *direction*)

Установка текущего направления движения.

##### Аргументы:

|  |  |
| --- | --- |
| *direction* | направление движения |

#### void TreadmillDriver::setDNF (bool *value*)

Управление частотой калибровки скорости.

##### Аргументы:

|  |  |
| --- | --- |
| *value* | true: включить частоту калибровки, false: отключить частоту калибровки. |

#### void TreadmillDriver::setExternal (ExtMode *mode*)

Установка режима работы с внешней стресс-системой.

##### Аргументы:

|  |  |
| --- | --- |
| *mode* | режима работы с внешней стресс-системой. |

#### void TreadmillDriver::setLift (Lift *lift*)

Установка подъемника в режиме пульта.

##### Аргументы:

|  |  |
| --- | --- |
| *lift* | режим подъемника. |

#### void TreadmillDriver::setRegister (Reg *reg*, int *value*)

Запись в регистр дорожки.

##### Аргументы:

|  |  |
| --- | --- |
| *reg* | номер регистра. |
| *value* | значение. |

#### void TreadmillDriver::setRemoteControlLifting (bool *enable*)

Включение/выключение режима пульта.

##### Аргументы:

|  |  |
| --- | --- |
| *enable* | true: включить, false: выключить. |

#### void TreadmillDriver::setSpeed (int *speed*)

Установка текущей скорости движения.

##### Аргументы:

|  |  |
| --- | --- |
| *speed* | скорость движения. |

#### void TreadmillDriver::setSpeedCalib (unsigned short *value*)

Установка калибровочного коэффициента.

##### Аргументы:

|  |  |
| --- | --- |
| *value* | калибровочный коэффициент. |

#### void TreadmillDriver::setState (RunState *state*)

Установка состояния дорожки.

##### Аргументы:

|  |  |
| --- | --- |
| *state* | состояние дорожки. |

#### Объявления и описания членов классов находятся в файлах:

src/TreadmillDriver.h

src/TreadmillDriver.cpp

## Класс UsbFlashDrive

Управление USB-диском.

#include <UsbFlashDrive.h>

### Открытые члены

virtual **~UsbFlashDrive** ()

*Деструктор.*

bool **isValid** () const

*Проверка работоспособности.*

void **fileCopy** (std::string src, std::string dst)

*Копирование файла.*

unsigned long int **getFreeSpace** ()

*Запрос свободного места.*

bool **isFileExisted** (std::string name)

*Проверка существования файла.*

void **driveMount** ()

*Монтирование устройства.*

void **driveUmount** ()

*Размонтирование устройства.*

### Друзья

class **DeviceMonitor**

### Подробное описание

Управление USB-диском.

### Методы

#### void UsbFlashDrive::fileCopy (std::string *src*, std::string *dst*)

Копирование файла.

##### Аргументы:

|  |  |
| --- | --- |
| *src* | откуда. |
| *dst* | куда. |

#### unsigned long int UsbFlashDrive::getFreeSpace ()

Запрос свободного места.

##### Возвращает:

размер свободного места.

#### bool UsbFlashDrive::isFileExisted (std::string *name*)

Проверка существования файла.

##### Аргументы:

|  |  |
| --- | --- |
| *name* | имя файла. |

##### Возвращает:

true: файл существует.

#### bool UsbFlashDrive::isValid () const

Проверка работоспособности.

##### Возвращает:

true: устройство работоспособно.

#### Объявления и описания членов классов находятся в файлах:

src/UsbFlashDrive.h

src/UsbFlashDrive.cpp

## Класс WaveFile

Звуковой файл.

#include <WaveFile.h>

### Открытые члены

**WaveFile** (std::string file\_name)

*Конструктор.*

virtual **~WaveFile** ()

*Деструктор.*

char \* **getData** (size\_t size)

*Получение звуковых данных.*

void **rewind** ()

*Установка указателя на начало данных.*

unsigned short **getChannels** () const

*Получение количества каналов.*

unsigned int **getSamplesPerSec** () const

*Получение количества отсчетов в секунду.*

snd\_pcm\_format\_t **getFormat** () const

*Получение формата кодирования.*

unsigned short **getBitsPerSample** () const

*Получение количества битов на отсчет.*

### Подробное описание

Звуковой файл.

### Конструктор(ы)

#### WaveFile::WaveFile (std::string *file\_name*)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *file\_name* | имя файла. |

### Методы

#### unsigned short WaveFile::getBitsPerSample () const

Получение количества битов на отсчет.

##### Возвращает:

количество битов на отсчет.

#### unsigned short WaveFile::getChannels () const

Получение количества каналов.

##### Возвращает:

количество каналов.

#### char \* WaveFile::getData (size\_t *size*)

Получение звуковых данных.

##### Аргументы:

|  |  |
| --- | --- |
| *size* | размер данных. |

##### Возвращает:

данные.

#### snd\_pcm\_format\_t WaveFile::getFormat () const

Получение формата кодирования.

##### Возвращает:

формат кодирования.

#### unsigned int WaveFile::getSamplesPerSec () const

Получение количества отсчетов в секунду.

##### Возвращает:

количество отсчетов в секунду.

#### Объявления и описания членов классов находятся в файлах:

src/WaveFile.h

src/WaveFile.cpp

## Структура WBCElement

Описание элемента виджета навигации BreadCrumbs.

#include <WidgetBreadCrumbs.h>

### Открытые атрибуты

std::string **text**

*Текст элемента.*

std::function< std::unique\_ptr<::**Event** >**WidgetImageButton::Event**)> **on\_click**

*Функция обратного вызова на нажатие элемента.*

**WTFStyle** **style**

*Стиль элемента.*

### Подробное описание

Описание элемента виджета навигации BreadCrumbs.

Объявления и описания членов структуры находятся в файле:

src/WidgetBreadCrumbs.h

## Класс Widget

Базовый класс виджетов.

#include <Widget.h>

Граф наследования:Widget:

### Открытые типы

enum **State** { **State::Active**, **State::Passive**, **State::Disabled** }*Состояние виджета.*

### Открытые члены

**Widget** (int x=0, int y=0, **State** active=**State::Passive**)

*Конструктор.*

virtual **~Widget** ()

*Деструктор.*

int **getX** () const

*Получение координаты x.*

**Widget** \* **setX** (int x)

*Установка координаты x.*

int **getY** () const

*Получение координаты y.*

**Widget** \* **setY** (int y)

*Установка координаты y.*

**State** **getState** () const

*Получение состояния.*

virtual **Widget** \* **setState** (**State** state)

*Установка состояния.*

virtual **Menu** \* **processEvent** (**Event** \*event)

*Функция обработки событий.*

virtual void **show** ()

*Отрисовка изображения.*

int **getHeight** () const

*Получение высоты виджета.*

void **setHeight** (int height)

*Установка высоты виджета.*

int **getWidth** () const

*Получение ширины виджета.*

void **setWidth** (int wide)

bool **isSelected** (int x, int y)

*Определение нажатия.*

**Menu** \* **getParent** () const

*Получение родительского виджета.*

void **setParent** (**Menu** \*parent)

*Установка родительского виджета.*

bool **isVisible** () const

*Получение видимости.*

**Widget** \* **setVisible** (bool visible)

*Установка видимости.*

### Открытые статические члены

static std::string **getStateName** (const **State** state)

*Получение названия состояния.*

### Открытые атрибуты

bool **m\_kbd\_focus**

*true: виджет имеет фокус клавиатуры, false: виджет не имеет фокуса клавиатуры.*

### Подробное описание

Базовый класс виджетов.

### Перечисления

#### enum Widget::State[strong]

Состояние виджета.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Active | Активный. |
| Passive | Пассивный. |
| Disabled | Заблокированный. |

### Конструктор(ы)

#### Widget::Widget (int *x* = 0, int *y* = 0, State *active* = State::Passive)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |
| *active* | состояние. |

### Методы

#### int Widget::getHeight () const

Получение высоты виджета.

##### Возвращает:

высота виджета.

#### Menu \* Widget::getParent () const

Получение родительского виджета.

##### Возвращает:

ссылка на родительский виджет.

#### Widget::State Widget::getState () const

Получение состояния.

##### Возвращает:

состояние.

#### std::string Widget::getStateName (const State *state*)[static]

Получение названия состояния.

##### Аргументы:

|  |  |
| --- | --- |
| *state* | состояние. |

##### Возвращает:

название состояния.

#### int Widget::getWidth () const

Получение ширины виджета.

##### Возвращает:

ширина виджета.

#### int Widget::getX () const

Получение координаты x.

##### Возвращает:

координата x.

#### int Widget::getY () const

Получение координаты y.

##### Возвращает:

координата y.

#### bool Widget::isSelected (int *x*, int *y*)

Определение нажатия.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x касания. |
| *y* | координата y касания. |

##### Возвращает:

true: есть нажатие, false: нет нажатия.

#### bool Widget::isVisible () const

Получение видимости.

##### Возвращает:

true: виджет отображается, false: виджет не отображается.

#### virtual Menu\* Widget::processEvent (Event \* *event*)[inline], [virtual]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

#### void Widget::setHeight (int *height*)

Установка высоты виджета.

##### Аргументы:

|  |  |
| --- | --- |
| *height* | высота виджета. |

#### void Widget::setParent (Menu \* *parent*)

Установка родительского виджета.

##### Аргументы:

|  |  |
| --- | --- |
| *parent* | ссылка на родительский виджет. |

#### Widget \* Widget::setState (State *state*)[virtual]

Установка состояния.

##### Аргументы:

|  |  |
| --- | --- |
| *state* | состояние. |

##### Возвращает:

ссылка на виджет.

Переопределяется в **WidgetImageButton** (*стр.177*).

#### Widget \* Widget::setVisible (bool *visible*)

Установка видимости.

##### Аргументы:

|  |  |
| --- | --- |
| *visible* | true: виджет отображается, false: виджет не отображается. |

##### Возвращает:

ссылка на виджет.

#### void Widget::setWidth (int *wide*)

Установка ширины виджета.

##### Аргументы:

|  |  |
| --- | --- |
| *wide* | ширина виджета. |

#### Widget \* Widget::setX (int *x*)

Установка координаты x.

##### Аргументы:

|  |  |
| --- | --- |
| *x* |  |

##### Возвращает:

ссылка на виджет.

#### Widget \* Widget::setY (int *y*)

Установка координаты y.

##### Аргументы:

|  |  |
| --- | --- |
| *y* | координата y. |

##### Возвращает:

ссылка на виджет.

#### Объявления и описания членов классов находятся в файлах:

src/Widget.h

src/Widget.cpp

## Класс WidgetBreadCrumbs

Виджет навигации BreadCrumbs.

#include <WidgetBreadCrumbs.h>

Граф наследования:WidgetBreadCrumbs:

### Открытые члены

**WidgetBreadCrumbs** (int x, int y, int width, int height, std::vector< **WBCElement** > &&elements)

*Конструктор.*

virtual **~WidgetBreadCrumbs** ()

*Деструктор.*

**Menu** \* **processEvent** (::**Event** \*event) override

*Функция обработки событий.*

void **show** () override

*Отрисовка изображения.*

### Дополнительные унаследованные члены

### Подробное описание

Виджет навигации BreadCrumbs.

### Конструктор(ы)

#### WidgetBreadCrumbs::WidgetBreadCrumbs (int *x*, int *y*, int *width*, int *height*, std::vector< WBCElement > && *elements*)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |
| *width* | ширина. |
| *height* | высота. |
| *elements* | элементы навигации. |

### Методы

#### Menu \* WidgetBreadCrumbs::processEvent (::Event \* *event*)[override]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

#### Объявления и описания членов классов находятся в файлах:

src/WidgetBreadCrumbs.h

src/WidgetBreadCrumbs.cpp

## Класс WidgetImageButton

Виджет кнопки с картинкой.

#include <WidgetImageButton.h>

Граф наследования:WidgetImageButton:

### Открытые типы

enum **Event** { **Event::Click**, **Event::Press**, **Event::Release** }*Событие.*

enum **Type** { **Type::Clicker**, **Type::PressRelease**, **Type::Generator**, **Type::Trigger** }*Тип.*

### Открытые члены

**WidgetImageButton** (int x, int y, std::string active\_file, std::string disabled\_file="", std::string passive\_file="", std::function< std::unique\_ptr<::**Event** >(**Event**)> on\_event={}, **Type** type=**Type::Clicker**, WidgetTextFieldSP text=WidgetTextFieldSP())

*Конструктор.*

virtual **~WidgetImageButton** ()

*Деструктор.*

**Menu** \* **processEvent** (::**Event** \*event) override

*Функция обработки событий.*

void **show** () override

*Отрисовка изображения.*

void **setPassive** (std::string file)

void **setActive** (std::string file)

void **setDisabled** (std::string file)

void **clearTrigger** ()

*Очистка кнопки-триггера.*

void **setTrigger** ()

*Установка кнопки-триггера.*

**WidgetImageButton** \* **setState** (**State** state) override

bool **isSet** ()

void **setRepeat** (int delay, int step)

void **setSound** (**SoundInterface::Sound** sound)

**WidgetImageButton** (const **WidgetImageButton** &)=delete

### Дополнительные унаследованные члены

### Подробное описание

Виджет кнопки с картинкой.

### Перечисления

#### enum WidgetImageButton::Event[strong]

Событие.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Click | Щелчок. |
| Press | Нажатие. |
| Release | Отпускание. |

#### enum WidgetImageButton::Type[strong]

Тип.

##### Элементы перечислений:

|  |  |
| --- | --- |
| Clicker | Кнопка-щелчок. |
| PressRelease | Нажатие-отпускание. |
| Generator | Кнопка с повтором. |
| Trigger | Кнопка-триггер. |

### Конструктор(ы)

#### WidgetImageButton::WidgetImageButton (int *x*, int *y*, std::string *active\_file*, std::string *disabled\_file* = "", std::string *passive\_file* = "", std::function< std::unique\_ptr<::Event >(Event)> *on\_event* = {}, Type *type* = Type::Clicker, WidgetTextFieldSP *text* = WidgetTextFieldSP())

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |
| *active\_file* | картинка активного состояния. |
| *disabled\_file* | картинка заблокированного состояния. |
| *passive\_file* | картинка пассивного состояния. |
| *on\_event* | функция обработки событий. |
| *type* | тип кнопки. |
| *text* | текстовый виджет, связанный с кнопкой. |

### Методы

#### bool WidgetImageButton::isSet ()

Проверка состояния кнопки-триггера.

##### Возвращает:

true: установлена, false: сброшена.

#### Menu \* WidgetImageButton::processEvent (::Event \* *event*)[override]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

#### void WidgetImageButton::setActive (std::string *file*)

Установка картинки активного состояния.

##### Аргументы:

|  |  |
| --- | --- |
| *file* | картинка активного состояния. |

#### void WidgetImageButton::setDisabled (std::string *file*)

Установка картинки заблокированного состояния.

##### Аргументы:

|  |  |
| --- | --- |
| *file* | картинка заблокированного состояния. |

#### void WidgetImageButton::setPassive (std::string *file*)

Установка картинки пассивного состояния.

##### Аргументы:

|  |  |
| --- | --- |
| *file* | картинка пассивного состояния. |

#### void WidgetImageButton::setRepeat (int *delay*, int *step*)

Установка параметров кнопки с повтором.

##### Аргументы:

|  |  |
| --- | --- |
| *delay* | задержка перед повтором. |
| *step* | период повтора. |

#### void WidgetImageButton::setSound (SoundInterface::Sound *sound*)

Установка звука на кнопку.

##### Аргументы:

|  |  |
| --- | --- |
| *sound* | идентификатор звука. |

#### WidgetImageButton \* WidgetImageButton::setState (State *state*)[override], [virtual]

Установка состояния.

##### Аргументы:

|  |  |
| --- | --- |
| *state* | состояние. |

##### Возвращает:

ссылка на виджет.

Переопределяет метод предка **Widget** (*стр.171*).

#### Объявления и описания членов классов находятся в файлах:

src/WidgetImageButton.h

src/WidgetImageButton.cpp

## Класс WidgetProfil

Виджет отображения графиков профиля.

#include <WidgetProfil.h>

Граф наследования:WidgetProfil:

### Открытые члены

**WidgetProfil** (int x, int y, std::vector< std::tuple< unsigned, float, float >> points)

*Конструктор.*

virtual **~WidgetProfil** ()

*Деструктор.*

### Дополнительные унаследованные члены

### Подробное описание

Виджет отображения графиков профиля.

### Конструктор(ы)

#### WidgetProfil::WidgetProfil (int *x*, int *y*, std::vector< std::tuple< unsigned, float, float >> *points*)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |
| *points* | точки графиков. |

#### Объявления и описания членов классов находятся в файлах:

src/WidgetProfil.h

src/WidgetProfil.cpp

## Класс WidgetRectangle

Виджет отображения прямоугольника.

#include <WidgetRectangle.h>

Граф наследования:WidgetRectangle:

### Открытые члены

**WidgetRectangle** (int x, int y, int width, int height, uint32\_t color=0x808080, bool visible=false)

*Конструктор.*

virtual **~WidgetRectangle** ()

*Деструктор.*

void **show** () override

*Отрисовка изображения.*

void **setColor** (uint32\_t color)

*Установка цвета.*

### Дополнительные унаследованные члены

### Подробное описание

Виджет отображения прямоугольника.

### Конструктор(ы)

#### WidgetRectangle::WidgetRectangle (int *x*, int *y*, int *width*, int *height*, uint32\_t *color* = 0x808080, bool *visible* = false)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |
| *width* | ширина. |
| *height* | высота. |
| *color* | цвет. |
| *visible* | видимость. |

### Методы

#### void WidgetRectangle::setColor (uint32\_t *color*)

Установка цвета.

##### Аргументы:

|  |  |
| --- | --- |
| *color* | цвет. |

#### Объявления и описания членов классов находятся в файлах:

src/WidgetRectangle.h

src/WidgetRectangle.cpp

## Класс WidgetText

Виджет однострочного текста.

#include <WidgetText.h>

Граф наследования:WidgetText:

### Открытые члены

**WidgetText** (int x, int y, std::string text, Align align=Align::Left, int size=30, uint32\_t color=0, uint32\_t color\_disabled=0x808080, std::function< std::string(void)> on\_update={}, FontStyle font\_style=FontStyle::Normal, int width=0, int lines=1, Align v\_align=Align::Center, int dy=0)

*Конструктор.*

virtual **~WidgetText** ()

*Деструктор.*

void **show** () override

*Отрисовка изображения.*

Align **getAlign** () const

*Получение вида горизонтального выравнивания.*

void **setAlign** (Align align)

*Установка вида горизонтального выравнивания.*

uint32\_t **getColor** () const

*Получение цвета текста.*

uint32\_t **getColorDisabled** () const

*Получение цвета заблокированного текста.*

const std::string & **getText** () const

*Получение текста виджета.*

void **setText** (const std::string &text)

### Дополнительные унаследованные члены

### Подробное описание

Виджет однострочного текста.

### Конструктор(ы)

#### WidgetText::WidgetText (int *x*, int *y*, std::string *text*, Align *align* = Align::Left, int *size* = 30, uint32\_t *color* = 0, uint32\_t *color\_disabled* = 0x808080, std::function< std::string(void)> *on\_update* = {}, FontStyle *font\_style* = FontStyle::Normal, int *width* = 0, int *lines* = 1, Align *v\_align* = Align::Center, int *dy* = 0)

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |
| *text* | отображаемый текст. |
| *align* | выравнивание. |
| *size* | размер шрифта. |
| *color* | цвет текста. |
| *color\_disabled* | цвет заблокированного текста. |
| *on\_update* | функция, вызываемая при отображении виджета. |
| *font\_style* | стиль шрифта. |
| *width* | ширина текста. |
| *lines* | количество строк. |
| *v\_align* | вертикальное выравнивание. |
| *dy* | расстояние между строками. |

### Методы

#### Align WidgetText::getAlign () const

Получение вида горизонтального выравнивания.

##### Возвращает:

вид горизонтального выравнивания.

#### uint32\_t WidgetText::getColor () const

Получение цвета текста.

##### Возвращает:

цвет текста.

#### uint32\_t WidgetText::getColorDisabled () const

Получение цвета заблокированного текста.

##### Возвращает:

цвет заблокированного текста.

#### const std::string & WidgetText::getText () const

Получение текста виджета.

##### Возвращает:

текст виджета.

#### void WidgetText::setAlign (Align *align*)

Установка вида горизонтального выравнивания.

##### Аргументы:

|  |  |
| --- | --- |
| *align* | вид горизонтального выравнивания. |

#### void WidgetText::setText (const std::string & *text*)

Установка текста виджета.

##### Аргументы:

|  |  |
| --- | --- |
| *text* | текст виджета. |

#### Объявления и описания членов классов находятся в файлах:

src/WidgetText.h

src/WidgetText.cpp

## Класс WidgetTextField

Виджет многострочного текста с клавиатурой.

#include <WidgetTextField.h>

Граф наследования:WidgetTextField:

### Открытые члены

**WidgetTextField** (int x, int y, int width, int height, **WTFStyle** style=default\_style, std::string text="", std::string prompt="", CheckFunc checkfunc=CheckFunc {})

*Конструктор.*

virtual **~WidgetTextField** ()

*Деструктор.*

void **show** () override

*Отрисовка изображения.*

**WidgetTextField** & **setText** (const std::string &text, const std::string &id="")

*Установка текста виджета.*

std::string **getText** () const

*Получение текста виджета.*

**WidgetTextField** & **setUpdateFunc** (UpdateFunc update\_func)

*Установка функции обновления.*

**WidgetTextField** & **setCheckFunc** (CheckFunc check\_func)

*Установка функции проверки.*

**WidgetTextField** & **setPrompt** (std::string prompt)

*Установка подсказки ввода.*

**WidgetTextField** & **setLimits** (float min, float max, std::string format="")

*Установка пределов ввода числа.*

**WidgetTextField** & **setMaxLimit** (float max)

*Установка верхнего предела ввода числа.*

**Menu** \* **processEvent** (::**Event** \*event) override

*Функция обработки событий.*

int **getIdCol** () const

*Получение идентификатора столбца.*

**WidgetTextField** & **setIdCol** (int idCol)

*Установка идентификатора столбца.*

int **getIdRow** () const

*Получение идентификатора строки.*

**WidgetTextField** & **setIdRow** (int idRow)

*Установка идентификатора строки.*

std::string **getTextId** () const

*Получение идентификатора текста.*

void **setFontSize** (unsigned size)

*Установка размера текста.*

### Дополнительные унаследованные члены

### Подробное описание

Виджет многострочного текста с клавиатурой.

### Конструктор(ы)

#### WidgetTextField::WidgetTextField (int *x*, int *y*, int *width*, int *height*, WTFStyle *style* = default\_style, std::string *text* = "", std::string *prompt* = "", CheckFunc *checkfunc* = CheckFunc  {})

Конструктор.

##### Аргументы:

|  |  |
| --- | --- |
| *x* | координата x. |
| *y* | координата y. |
| *width* | ширина текста. |
| *height* | высота текста. |
| *style* | стиль виджета. |
| *text* | отображаемый текст. |
| *prompt* | подсказка ввода. |
| *checkfunc* | функция проверки ввода. |

### Методы

#### int WidgetTextField::getIdCol () const

Получение идентификатора столбца.

##### Возвращает:

идентификатор столбца.

#### int WidgetTextField::getIdRow () const

Получение идентификатора строки.

##### Возвращает:

идентификатор строки.

#### std::string WidgetTextField::getText () const

Получение текста виджета.

##### Возвращает:

текст виджета.

#### std::string WidgetTextField::getTextId () const

Получение идентификатора текста.

##### Возвращает:

идентификатор текста.

#### Menu \* WidgetTextField::processEvent (::Event \* *event*)[override]

Функция обработки событий.

##### Аргументы:

|  |  |
| --- | --- |
| *event* | событие. |

##### Возвращает:

ссылка на новое меню.

#### WidgetTextField & WidgetTextField::setCheckFunc (CheckFunc *check\_func*)

Установка функции проверки.

##### Аргументы:

|  |  |
| --- | --- |
| *check\_func* | функция проверки. |

##### Возвращает:

ссылка на виджет.

#### void WidgetTextField::setFontSize (unsigned *size*)

Установка размера текста.

##### Аргументы:

|  |  |
| --- | --- |
| *size* | размер текста. |

#### WidgetTextField & WidgetTextField::setIdCol (int *idCol*)

Установка идентификатора столбца.

##### Аргументы:

|  |  |
| --- | --- |
| *idCol* | идентификатор столбца. |

##### Возвращает:

ссылка на виджет.

#### WidgetTextField & WidgetTextField::setIdRow (int *idRow*)

Установка идентификатора строки.

##### Аргументы:

|  |  |
| --- | --- |
| *idRow* | идентификатор строки. |

##### Возвращает:

ссылка на виджет.

#### WidgetTextField & WidgetTextField::setLimits (float *min*, float *max*, std::string *format* = "")

Установка пределов ввода числа.

##### Аргументы:

|  |  |
| --- | --- |
| *min* | нижний предел. |
| *max* | верхний предел. |
| *format* | формат вывода пределов. |

##### Возвращает:

ссылка на виджет.

#### WidgetTextField & WidgetTextField::setMaxLimit (float *max*)

Установка верхнего предела ввода числа.

##### Аргументы:

|  |  |
| --- | --- |
| *max* | верхний предел ввода числа. |

##### Возвращает:

ссылка на виджет.

#### WidgetTextField & WidgetTextField::setPrompt (std::string *prompt*)

Установка подсказки ввода.

##### Аргументы:

|  |  |
| --- | --- |
| *prompt* | подсказка ввода. |

##### Возвращает:

ссылка на виджет.

#### WidgetTextField & WidgetTextField::setText (const std::string & *text*, const std::string & *id* = "")

Установка текста виджета.

##### Аргументы:

|  |  |
| --- | --- |
| *text* | текст виджета. |
| *id* | идентификатор. |

##### Возвращает:

ссылка на виджет.

#### WidgetTextField & WidgetTextField::setUpdateFunc (UpdateFunc *update\_func*)

Установка функции обновления.

##### Аргументы:

|  |  |
| --- | --- |
| *update\_func* | функция обновления. |

##### Возвращает:

ссылка на виджет.

#### Объявления и описания членов классов находятся в файлах:

src/WidgetTextField.h

src/WidgetTextField.cpp

## Класс WiFiRemote

Класс поддержки работы с планшетом.

#include <WiFiRemote.h>

### Открытые члены

**WiFiRemote** ()

*Конструктор.*

virtual **~WiFiRemote** ()

*Деструктор.*

void **send** (std::vector< unsigned char > &data, int id)

*Отправка данных на планшет.*

bool **isConnected** ()

*Проверка подключения.*

### Подробное описание

Класс поддержки работы с планшетом.

### Методы

#### bool WiFiRemote::isConnected ()

Проверка подключения.

##### Возвращает:

true: подключен, false: не подключен.

#### void WiFiRemote::send (std::vector< unsigned char > & *data*, int *id*)

Отправка данных на планшет.

##### Аргументы:

|  |  |
| --- | --- |
| *data* | данные. |
| *id* | идентификатор пакета. |

#### Объявления и описания членов классов находятся в файлах:

src/WiFiRemote.h

src/WiFiRemote.cpp

## Структура WTFStyle

Структура стиля виджета.

#include <WidgetTextField.h>

### Открытые атрибуты

std::string **text**

*Текст виджета.*

std::string **prompt**

*Подсказка ввода.*

Align **halign**

*Горизонтальное выравнивание.*

Align **valign**

*Вертикальное выравнивание.*

unsigned **lines**

*Максимальное количество строк текста.*

unsigned **dy**

*Расстояние между строками.*

FontStyle **font\_style**

*Стиль шрифта.*

int **font\_size**

*Размер шрифта.*

float **text\_rotate**

*Угол поворота текста в радианах.*

uint32\_t **text\_color\_active**

*Цвет активного текста.*

uint32\_t **text\_color\_passive**

*Цвет текста.*

uint32\_t **text\_color\_disabled**

*Цвет заблокированного текста.*

uint32\_t **rect\_color\_active**

*Цвет активного поля текста.*

uint32\_t **rect\_color\_passive**

*Цвет поля текста.*

uint32\_t **rect\_color\_disabled**

*Цвет заблокированного поля текста.*

std::string **img\_file\_active**

*Картинка активного фона.*

std::string **img\_file\_passive**

*Картинка фона.*

std::string **img\_file\_disabled**

*Картинка заблокированного фона.*

CheckFunc **check\_func**

*Функция, вызываемая при вводе текста.*

UpdateFunc **update\_func**

*Функция, вызываемая при отображении виджета.*

KbdType **kbd\_type**

*Тип клавиатуры.*

int **text\_max\_size**

*Максимальная длина текста.*

float **max\_limit**

*Верхний предел при вводе числа.*

float **min\_limit**

*Нижний предел при вводе числа.*

std::string **limit\_format**

*Формат вывода пределов.*

std::string **password\_char**

*Символ отображения пароля.*

std::string **num\_format**

*Формат отображения чисел.*

int **hpad**

*Отступ от края справа и слева.*

### Подробное описание

Структура стиля виджета.

Объявления и описания членов структуры находятся в файле:

src/WidgetTextField.h

# Алфавитный указатель

Acceleration

DbConnector, 35, 37

Active

Widget, 169

addDevice

BluetoothInterface, 28

addRunSession

DbConnector, 37

addWidget

Menu, 70

Age

DbConnector, 36

Alarm

SoundInterface, 143

allTrim

Utils, 15

Angle

DbConnector, 35

Utils, 14

AngleFmt

DbConnector, 37

AngleUnit

DbConnector, 37

ba2S

Utils, 15

Backward

TreadmillDriver, 154

Bell

SoundInterface, 143

BleCharacteristic, 22

BleCharacteristic, 22

BleDevice, 23

BleDevice, 23

BleObject, 25

BleObject, 25

BleService, 26

BleService, 26

BluetoothInterface, 27

addDevice, 28

BluetoothInterface, 27

connect, 28

disconnect, 28

needScan, 28

onSignal, 28

readCharacteristic, 29

startNotify, 29

startScan, 29

Bottom

ModalKeyboardAlpha, 82

Buttons

ModalCommonDialog, 78

bytes2Hex

Utils, 15

clamp

Utils, 15

Clear

TouchEvent, 149

Click

SoundInterface, 143

WidgetImageButton, 176

Clicker

WidgetImageButton, 176

CloseButton

ModalCommonDialog, 78

connect

BluetoothInterface, 28

HrMonitor, 55

Context, 31

DbConnector, 33

Acceleration, 35, 37

addRunSession, 37

Age, 36

Angle, 35

AngleFmt, 37

AngleUnit, 37

deletePatient, 38

deleteProtocol, 38

Description, 36, 37

Diagnosis, 36

Distance, 35, 37

Doctor, 36

dumpDataSet, 38

Duration, 35, 37

DurationFmt, 37

DurationUnit, 37

Energy, 35, 37

exportToEmi, 38

Fio, 34, 37

FirstName, 35, 36

getHistory, 38

getIntervals, 39

getParam, 39

getParamEnum, 39

getPatient, 39

getPatients, 39

getProtocolModifs, 40

getProtocolParams, 40

getProtocols, 40

getStatistic, 40

History, 34

HRate, 35

HRateMax, 37

HRateMin, 37

Id, 36

importFromEmi, 40

insertPatient, 41

insertProtocol, 41

IntervalData, 35

LastName, 35, 36

MiddleName, 35, 36

Modification, 37

Name, 35, 36, 37

Number, 36

PatientList, 36

PowerAero, 35

PowerMech, 35

Profile, 34, 36

ProfileId, 36

ProtocolList, 36

ProtocolParam, 36

restoreDB, 41

setParam, 41

Sex, 36

Speed, 35

SpeedFmt, 37

SpeedUnit, 37

Statistic, 37

Time, 34

TimeStamp, 35

Type, 35, 37

updatePatient, 41

updateProtocol, 42

Weight, 36

DbError, 43

DbError, 43

Debug

Logger, 63

deg2Radian

Utils, 15

Degree

Utils, 14

deletePatient

DbConnector, 38

deleteProtocol

DbConnector, 38

Description

DbConnector, 36, 37

DeviceMonitor, 44

getUsbDrives, 44

setUsbDriveEventListener, 44

Diagnosis

DbConnector, 36

Digits

ModalKeyboardAlpha, 82

Direction

TreadmillDriver, 154

Disabled

Widget, 169

disconnect

BluetoothInterface, 28

DisplayLightnessPwm, 45

Distance

DbConnector, 35, 37

Utils, 13

Doctor

DbConnector, 36

Down

TreadmillDriver, 155

drawKbdButton

GraphicEngine, 52

drawLine

GraphicEngine, 52

drawLineGradient

GraphicEngine, 52

drawMultiText

GraphicEngine, 52

drawRectangle

GraphicEngine, 53

drawSurface

GraphicEngine, 53

drawText

GraphicEngine, 53

dumpDataSet

DbConnector, 38

Duration

DbConnector, 35, 37

DurationFmt

DbConnector, 37

DurationUnit

DbConnector, 37

Edit

PatientNewMenu, 95

RehabUserProtocolsViewMenu, 119

EmergencyButton

Event, 48

EnableBitMaskOperators< Enum >, 46

Energy

DbConnector, 35, 37

Utils, 13

EngCapital

ModalKeyboardAlpha, 82

EngSmall

ModalKeyboardAlpha, 81

Error

Logger, 63

Event, 47

EmergencyButton, 48

Event, 48

EventType, 47

getExtra, 48

getName, 49

getType, 49

HrMonitorEvent, 48

KbdEvent, 48

Menu, 48

ModalCancel, 48

ModalNo, 48

ModalYes, 48

Quit, 47

Refresh, 48

Restart, 48

Shutdown, 48

Timer, 47

Touch, 47

TreadmillConnected, 48

TreadmillDisconnected, 48

TreadmillReady, 48

WidgetImageButton, 175

Zastup, 48

EventType

Event, 47

exportToEmi

DbConnector, 38

ExtMode

TreadmillDriver, 154

fileCopy

UsbFlashDrive, 163

Fio

DbConnector, 34, 37

FirstName

DbConnector, 35, 36

Fixed

ModalKeyboardDigital, 84

Flags

ModalCommonDialog, 77

formatFloat

Utils, 16

formatTime

Utils, 16

Forward

TreadmillDriver, 154

FreeRunMenu, 50

Generator

WidgetImageButton, 176

getAcceleration

RunSession, 122

TreadmillDriver, 156

getAccelMax

RunSession, 123

getAccelString

RunSession, 123

getAccelTimeValue

TreadmillDriver, 156

getAccelValue

TreadmillDriver, 156

getActualSpeed

TreadmillDriver, 156

getAlign

WidgetText, 183

getAngle

RunSession, 123

SettingsBosParamsMenu, 130

TreadmillDriver, 157

getAngleDegree

RunSession, 123

getAngleFormat

RunSession, 123

getAngleMax

RunSession, 123

getAngleString

RunSession, 124

getBitsPerSample

WaveFile, 165

getCalibrationAngleX

TreadmillDriver, 157

getCalibrationAngleY

TreadmillDriver, 157

getChannels

WaveFile, 166

getColor

WidgetText, 183

getColorDisabled

WidgetText, 183

getCurrentTime

Utils, 16

getCurrentTimeString

Utils, 16

getData

WaveFile, 166

getDeceleration

TreadmillDriver, 157

getDeviceAt

HrMonitor, 55

getDeviceListSize

HrMonitor, 56

getDirection

TreadmillDriver, 157

getDistance

RunSession, 124

getDistanceString

RunSession, 124

getDuration

RunSession, 124

getEnergy

RunSession, 124

getEnergyString

RunSession, 124

getEvent

SyncQueue, 145

getExtra

Event, 48

getFileTimestamp

Utils, 16

getFormat

WaveFile, 166

getFreeSpace

UsbFlashDrive, 163

getGradientColor

GraphicEngine, 54

getHeight

Menu, 70

Widget, 169

getHistory

DbConnector, 38

getHrate

RunSession, 125

getHrateAvg

RunSession, 125

getHrateMax

RunSession, 125

getHrateMin

RunSession, 125

getIdCol

WidgetTextField, 186

getIdRow

WidgetTextField, 186

getIntervals

DbConnector, 39

getKey

KbdEvent, 57

getLocale

Locale, 61

getLogTimestamp

Utils, 17

getMaxAngle

TreadmillDriver, 157

getMaxBackwardSpeed

TreadmillDriver, 157

getMaxForwardSpeed

TreadmillDriver, 158

getMenu

MenuEvent, 74

getMinSpeed

TreadmillDriver, 158

getName

Event, 49

getParam

DbConnector, 39

getParamEnum

DbConnector, 39

getParamFloat

Parameter, 89

getParamInt

Parameter, 90

getParamString

Parameter, 90

getParent

Widget, 170

getPatient

DbConnector, 39

getPatientId

RunSession, 125

getPatients

DbConnector, 39

getPicture

PictureManager, 100

getPinLevel

GpioDriver, 10

getPowerAero

RunSession, 125

getPowerAeroAvg

RunSession, 126

getPowerAeroString

RunSession, 126

getPowerMech

RunSession, 126

getPowerMechAvg

RunSession, 126

getPowerMechByParams

RunSession, 126

getPowerMechFormat

RunSession, 126

getPowerMechString

RunSession, 127

getPowerWatt

RunSession, 127

getProfile

PatientNewMenu, 96

RunSession, 127

getProtocolModifs

DbConnector, 40

getProtocolParams

DbConnector, 40

getProtocols

DbConnector, 40

getRegister

TreadmillDriver, 158

getRegisterState

TreadmillDriver, 158

getSamplesPerSec

WaveFile, 166

getSelected

Menu, 70

getSize

SyncQueue, 145

getSpeed

RunSession, 127

SettingsBosParamsMenu, 131

TreadmillDriver, 158

getSpeedAvg

RunSession, 127

getSpeedFormat

RunSession, 127

getSpeedKmH

RunSession, 128

getSpeedStep

TreadmillDriver, 158

getSpeedString

RunSession, 128

getState

TreadmillDriver, 159

Widget, 170

getStateName

Widget, 170

getStatistic

DbConnector, 40

getText

WidgetText, 183

WidgetTextField, 186

getTextId

WidgetTextField, 187

getTextSize

GraphicEngine, 54

getTime

SettingsBosParamsMenu, 131

getType

Event, 49

Menu, 71

TouchEvent, 150

getUsbDrives

DeviceMonitor, 44

getWidth

Menu, 71

Widget, 170

getX

Menu, 71

TouchEvent, 150

Widget, 170

getY

Menu, 71

TouchEvent, 150

Widget, 170

GpioDriver, 9

getPinLevel, 10

High, 9

In, 9

Low, 9

Out, 9

PinDirection, 9

PinLevel, 9

setPinDirection, 10

setPinLevel, 10

GraphicEngine, 51

drawKbdButton, 52

drawLine, 52

drawLineGradient, 52

drawMultiText, 52

drawRectangle, 53

drawSurface, 53

drawText, 53

getGradientColor, 54

getTextSize, 54

GreenTitle

ModalCommonDialog, 77

High

GpioDriver, 9

History

DbConnector, 34

HRate

DbConnector, 35

HRateMax

DbConnector, 37

HRateMin

DbConnector, 37

HrMonitor, 55

connect, 55

getDeviceAt, 55

getDeviceListSize, 56

isConnected, 56

HrMonitorEvent

Event, 48

Icons

ModalCommonDialog, 78

Id

DbConnector, 36

importFromEmi

DbConnector, 40

In

GpioDriver, 9

Info

Logger, 63

SoundInterface, 143

init

Locale, 61

insertPatient

DbConnector, 41

insertProtocol

DbConnector, 41

instance

Logger, 64

int2Hex

Utils, 17

Integer

ModalKeyboardDigital, 84

IntervalData

DbConnector, 35

into

KeyboardButton, 59

isCalibrating

TreadmillDriver, 159

isConnected

HrMonitor, 56

WiFiRemote, 190

isCrossError

TreadmillDriver, 159

isEmegencyStop

TreadmillDriver, 159

isFileExisted

UsbFlashDrive, 164

isLeapYear

Utils, 17

isLifterAlarm

TreadmillDriver, 159

isPFreqAlarm

TreadmillDriver, 159

isPlcConnected

TreadmillDriver, 160

isReady

TreadmillDriver, 160

isSelected

Widget, 171

isSet

WidgetImageButton, 176

isSettingPoint

TreadmillDriver, 160

isValid

UsbFlashDrive, 164

isVisible

Widget, 171

isZeroPoint

TreadmillDriver, 160

Joul

Utils, 14

KbdEvent, 57

Event, 48

getKey, 57

KbdEvent, 57

KeyboardButton, 59

into, 59

KiloKall

Utils, 14

KiloMeter

Utils, 14

KiloWatt

Utils, 14

kmh2milih

Utils, 17

kmh2ms

Utils, 17

KmPerHour

Utils, 14

KWattPerHour

Utils, 14

LastName

DbConnector, 35, 36

Layout

ModalKeyboardAlpha, 81

ModalKeyboardDigital, 84

Lift

TreadmillDriver, 155

loadPicture

PictureManager, 100

Locale, 61

getLocale, 61

init, 61

tr, 61

lockFile

Utils, 18

log

Logger, 64

TreadmillDriver, 160

Logger, 63

Debug, 63

Error, 63

Info, 63

instance, 64

log, 64

LogLevel, 63

setLogLevel, 64

Trace, 63

LogLevel

Logger, 63

loop

MainLoop, 65

Low

GpioDriver, 9

MainLoop, 65

loop, 65

MainMenu, 66

processEvent, 66

makeDir

Utils, 18

makePath

Utils, 18

Menu, 67

addWidget, 70

Event, 48

getHeight, 70

getSelected, 70

getType, 71

getWidth, 71

getX, 71

getY, 71

Menu, 70

processEvent, 71

setPatient, 72

setType, 72

setX, 72

setY, 72

show, 72

MenuEvent, 74

getMenu, 74

MenuEvent, 74

MET

Utils, 14

Meter

Utils, 14

MiddleName

DbConnector, 35, 36

Mile

Utils, 14

MilePerHour

Utils, 14

milih2kmh

Utils, 18

MlMinKg

Utils, 14

ModalAuthAdmin, 75

processEvent, 75

ModalAuthDoctor, 76

ModalCancel

Event, 48

ModalCommonDialog, 77

Buttons, 78

CloseButton, 78

Flags, 77

GreenTitle, 77

Icons, 78

ModalCommonDialog, 78

NoSound, 77

OkButton, 78

setText, 78

setTimer, 78

StopIcon, 78

YellowTitle, 77

Yes, 78

YesCancel, 78

YesNoCancel, 78

ModalCommonError, 80

ModalCommonError, 80

ModalKeyboardAlpha, 81

Bottom, 82

Digits, 82

EngCapital, 82

EngSmall, 81

Layout, 81

ModalKeyboardAlpha, 82

Position, 82

processEvent, 82

RusCapital, 81

RusSmall, 81

setLayout, 82

show, 83

Symbols, 82

Top, 82

ModalKeyboardDigital, 84

Fixed, 84

Integer, 84

Layout, 84

ModalKeyboardDigital, 85

processEvent, 85

show, 85

ModalListSelector, 86

ModalListSelector, 86

ModalNo

Event, 48

ModalPatientHistoryFilter, 87

ModalWarmUpParams, 88

ModalYes

Event, 48

Mode

PatientNewMenu, 95

RehabUserProtocolsViewMenu, 119

Modification

DbConnector, 37

Move

TouchEvent, 149

MPerSec

Utils, 14

ms2kmh

Utils, 18

Name

DbConnector, 35, 36, 37

needScan

BluetoothInterface, 28

New

PatientNewMenu, 95

RehabUserProtocolsViewMenu, 119

NoSound

ModalCommonDialog, 77

Number

DbConnector, 36

Off

TreadmillDriver, 154

OkButton

ModalCommonDialog, 78

On

TreadmillDriver, 154

onSignal

BluetoothInterface, 28

openSerial

Utils, 19

Out

GpioDriver, 9

Parameter, 89

getParamFloat, 89

getParamInt, 90

getParamString, 90

Parameter, 89

setParamString, 90

Passive

Widget, 169

pathExist

Utils, 19

PatientDetailsMenu, 91

PatientDetailsMenu, 91

PatientEditMenu, 92

processEvent, 92

PatientHistoryMenu, 93

processEvent, 93

PatientList

DbConnector, 36

PatientListMenu, 94

processEvent, 94

PatientNewMenu, 95

Edit, 95

getProfile, 96

Mode, 95

New, 95

PatientNewMenu, 96

processEvent, 96

setProfile, 96

PatientSearchMenu, 97

PatientStatsMenu, 98

Pause

TreadmillDriver, 156

PersistedParameter, 99

PersistedParameter, 99

Phys

Utils, 13

physConvert

Utils, 19

PictureManager, 100

getPicture, 100

loadPicture, 100

PinDirection

GpioDriver, 9

PinLevel

GpioDriver, 9

play

SoundInterface, 143

PLC\_ACX

TreadmillDriver, 155

PLC\_ACY

TreadmillDriver, 155

PLC\_AX

TreadmillDriver, 155

PLC\_CAL

TreadmillDriver, 155

PLC\_CD

TreadmillDriver, 155

PLC\_CL

TreadmillDriver, 155

PLC\_CS

TreadmillDriver, 155

PLC\_CTA

TreadmillDriver, 155

PLC\_CTD

TreadmillDriver, 155

PLC\_ED

TreadmillDriver, 155

PLC\_EX

TreadmillDriver, 155

PLC\_ID

TreadmillDriver, 155

PLC\_SD

TreadmillDriver, 155

PLC\_STAT

TreadmillDriver, 155

Position

ModalKeyboardAlpha, 82

Power

Utils, 13

PowerAero

DbConnector, 35

PowerAir

Utils, 14

PowerMech

DbConnector, 35

Press

WidgetImageButton, 176

PressRelease

WidgetImageButton, 176

ProceduresMenu, 101

Procent

Utils, 14

processEvent

MainMenu, 66

Menu, 71

ModalAuthAdmin, 75

ModalKeyboardAlpha, 82

ModalKeyboardDigital, 85

PatientEditMenu, 92

PatientHistoryMenu, 93

PatientListMenu, 94

PatientNewMenu, 96

RehabBosBaseMenu, 104

RehabProfileRunMenu, 109

RehabTestProtocolMenu, 114

RehabUserProtocolsEditMenu, 116

RehabUserProtocolsMenu, 118

RehabUserProtocolsViewMenu, 120

SettingsCalibrateMenu, 132

SettingsClockMenu, 133

SettingsExportMenu, 134

SettingsExtraCalibActuatorMenu, 135

SettingsHrMenu, 137

SettingsPasswordMenu, 139

SettingsRegistersMenu, 140

Widget, 171

WidgetBreadCrumbs, 173

WidgetImageButton, 177

WidgetTextField, 187

Profil

RehabUserProtocolsMenu, 117

Profile

DbConnector, 34, 36

ProfileId

DbConnector, 36

ProfilSelect

RehabUserProtocolsMenu, 117

ProMode

RehabUserProtocolsMenu, 117

Protocol

RehabUserProtocolsMenu, 117

ProtocolList

DbConnector, 36

ProtocolParam

DbConnector, 36

putAsNext

SyncQueue, 146

putEvent

SyncQueue, 146

Quit

Event, 47

Radian

Utils, 14

readCharacteristic

BluetoothInterface, 29

Refresh

Event, 48

Reg

TreadmillDriver, 155

RehabBosBaseMenu, 102

processEvent, 104

RehabBosBaseMenu, 103

RehabBosBaseMenu::BosCorrectionData, 30

RehabBosHrMinMaxMenu, 105

RehabBosHrPowerMenu, 106

RehabExternalMenu, 107

RehabMenu, 108

RehabProfileRunMenu, 109

processEvent, 109

RehabProtocolSelectionMenu, 110

RehabSelHrMenu, 111

RehabSelMethodMenu, 112

RehabSelProfileMenu, 113

RehabTestProtocolMenu, 114

processEvent, 114

RehabUserProfilSelectMenu, 115

RehabUserProtocolsEditMenu, 116

processEvent, 116

RehabUserProtocolsMenu, 117

processEvent, 118

Profil, 117

ProfilSelect, 117

ProMode, 117

Protocol, 117

RehabUserProtocolsViewMenu, 119

Edit, 119

Mode, 119

New, 119

processEvent, 120

RehabUserProtocolsViewMenu, 119

View, 119

Release

WidgetImageButton, 176

Restart

Event, 48

restoreDB

DbConnector, 41

Run

TreadmillDriver, 156

RunSession, 121

getAcceleration, 122

getAccelMax, 123

getAccelString, 123

getAngle, 123

getAngleDegree, 123

getAngleFormat, 123

getAngleMax, 123

getAngleString, 124

getDistance, 124

getDistanceString, 124

getDuration, 124

getEnergy, 124

getEnergyString, 124

getHrate, 125

getHrateAvg, 125

getHrateMax, 125

getHrateMin, 125

getPatientId, 125

getPowerAero, 125

getPowerAeroAvg, 126

getPowerAeroString, 126

getPowerMech, 126

getPowerMechAvg, 126

getPowerMechByParams, 126

getPowerMechFormat, 126

getPowerMechString, 127

getPowerWatt, 127

getProfile, 127

getSpeed, 127

getSpeedAvg, 127

getSpeedFormat, 127

getSpeedKmH, 128

getSpeedString, 128

start, 128

RunState

TreadmillDriver, 156

RusCapital

ModalKeyboardAlpha, 81

RusSmall

ModalKeyboardAlpha, 81

s2HB

Utils, 19

s2LB

Utils, 19

send

WiFiRemote, 190

setAcceleration

TreadmillDriver, 160

setActive

WidgetImageButton, 177

setAlign

WidgetText, 183

setAngle

TreadmillDriver, 160

setCheckFunc

WidgetTextField, 187

setColor

WidgetRectangle, 180

setDateTime

Utils, 20

setDeceleration

TreadmillDriver, 161

setDirection

TreadmillDriver, 161

setDisabled

WidgetImageButton, 177

setDNF

TreadmillDriver, 161

setExternal

TreadmillDriver, 161

setFontSize

WidgetTextField, 187

setHeight

Widget, 171

setIdCol

WidgetTextField, 187

setIdRow

WidgetTextField, 187

setLayout

ModalKeyboardAlpha, 82

setLift

TreadmillDriver, 161

setLimits

WidgetTextField, 188

setLogLevel

Logger, 64

setMaxLimit

WidgetTextField, 188

setParam

DbConnector, 41

setParamString

Parameter, 90

setParent

Widget, 171

setPassive

WidgetImageButton, 177

setPatient

Menu, 72

setPinDirection

GpioDriver, 10

setPinLevel

GpioDriver, 10

setProfile

PatientNewMenu, 96

setPrompt

WidgetTextField, 188

setRegister

TreadmillDriver, 161

setRemoteControlLifting

TreadmillDriver, 162

setRepeat

WidgetImageButton, 177

setSound

WidgetImageButton, 177

setSpeed

TreadmillDriver, 162

setSpeedCalib

TreadmillDriver, 162

setState

TreadmillDriver, 162

Widget, 171

WidgetImageButton, 177

setText

ModalCommonDialog, 78

WidgetText, 184

WidgetTextField, 188

setTimer

ModalCommonDialog, 78

SettingsAccelMenu, 129

SettingsBosParamsMenu, 130

getAngle, 130

getSpeed, 131

getTime, 131

SettingsCalibrateMenu, 132

processEvent, 132

SettingsClockMenu, 133

processEvent, 133

SettingsExportMenu, 134

processEvent, 134

SettingsExtraCalibActuatorMenu, 135

processEvent, 135

SettingsExtraCalibDriveMenu, 136

SettingsHrMenu, 137

processEvent, 137

SettingsMenu, 138

SettingsPasswordMenu, 139

processEvent, 139

SettingsRegistersMenu, 140

processEvent, 140

SettingsUnitsMenu, 141

setType

Menu, 72

setUpdateFunc

WidgetTextField, 188

setUsbDriveEventListener

DeviceMonitor, 44

setVisible

Widget, 172

setWidth

Widget, 172

setX

Menu, 72

Widget, 172

setY

Menu, 72

Widget, 172

Sex

DbConnector, 36

show

Menu, 72

ModalKeyboardAlpha, 83

ModalKeyboardDigital, 85

Shutdown

Event, 48

Silent

SoundInterface, 143

SizeGraph, 142

Sound

SoundInterface, 143

SoundInterface, 143

Alarm, 143

Bell, 143

Click, 143

Info, 143

play, 143

Silent, 143

Sound, 143

Welcome, 143

Speed

DbConnector, 35

Utils, 13

SpeedFmt

DbConnector, 37

SpeedUnit

DbConnector, 37

start

RunSession, 128

startNotify

BluetoothInterface, 29

startScan

BluetoothInterface, 29

startTimerMillis

Timer, 148

State

Widget, 169

Statistic

DbConnector, 37

Stop

TreadmillDriver, 155, 156

StopIcon

ModalCommonDialog, 78

Symbols

ModalKeyboardAlpha, 82

SyncQueue

getEvent, 145

getSize, 145

putAsNext, 146

putEvent, 146

SyncQueue< A >, 145

ThreadClass, 147

Time

DbConnector, 34

Timer, 148

Event, 47

startTimerMillis, 148

Timer, 148

TimeStamp

DbConnector, 35

Top

ModalKeyboardAlpha, 82

Touch

Event, 47

TouchEvent, 149

TouchEvent, 149

Clear, 149

getType, 150

getX, 150

getY, 150

Move, 149

Touch, 149

TouchEvent, 150

TouchType, 149

Untouch, 149

TouchScreenDriver, 151

TouchType

TouchEvent, 149

tr

Locale, 61

Trace

Logger, 63

TreadmillConnected

Event, 48

TreadmillDisconnected

Event, 48

TreadmillDriver, 152

Backward, 154

Direction, 154

Down, 155

ExtMode, 154

Forward, 154

getAcceleration, 156

getAccelTimeValue, 156

getAccelValue, 156

getActualSpeed, 156

getAngle, 157

getCalibrationAngleX, 157

getCalibrationAngleY, 157

getDeceleration, 157

getDirection, 157

getMaxAngle, 157

getMaxBackwardSpeed, 157

getMaxForwardSpeed, 158

getMinSpeed, 158

getRegister, 158

getRegisterState, 158

getSpeed, 158

getSpeedStep, 158

getState, 159

isCalibrating, 159

isCrossError, 159

isEmegencyStop, 159

isLifterAlarm, 159

isPFreqAlarm, 159

isPlcConnected, 160

isReady, 160

isSettingPoint, 160

isZeroPoint, 160

Lift, 155

log, 160

Off, 154

On, 154

Pause, 156

PLC\_ACX, 155

PLC\_ACY, 155

PLC\_AX, 155

PLC\_CAL, 155

PLC\_CD, 155

PLC\_CL, 155

PLC\_CS, 155

PLC\_CTA, 155

PLC\_CTD, 155

PLC\_ED, 155

PLC\_EX, 155

PLC\_ID, 155

PLC\_SD, 155

PLC\_STAT, 155

Reg, 155

Run, 156

RunState, 156

setAcceleration, 160

setAngle, 160

setDeceleration, 161

setDirection, 161

setDNF, 161

setExternal, 161

setLift, 161

setRegister, 161

setRemoteControlLifting, 162

setSpeed, 162

setSpeedCalib, 162

setState, 162

Stop, 155, 156

Up, 155

TreadmillReady

Event, 48

Trigger

WidgetImageButton, 176

Type

DbConnector, 35, 37

WidgetImageButton, 176

typeName

Utils, 20

TypeParser, 11

u16string\_to\_utf8

Utils, 20

Unit

Utils, 14

Unknown

Utils, 14

Untouch

TouchEvent, 149

Up

TreadmillDriver, 155

updatePatient

DbConnector, 41

updateProtocol

DbConnector, 42

UsbFlashDrive, 163

fileCopy, 163

getFreeSpace, 163

isFileExisted, 164

isValid, 164

utf8\_to\_u16string

Utils, 20

utf8\_to\_wstring

Utils, 20

Utils, 12

allTrim, 15

Angle, 14

ba2S, 15

bytes2Hex, 15

clamp, 15

deg2Radian, 15

Degree, 14

Distance, 13

Energy, 13

formatFloat, 16

formatTime, 16

getCurrentTime, 16

getCurrentTimeString, 16

getFileTimestamp, 16

getLogTimestamp, 17

int2Hex, 17

isLeapYear, 17

Joul, 14

KiloKall, 14

KiloMeter, 14

KiloWatt, 14

kmh2milih, 17

kmh2ms, 17

KmPerHour, 14

KWattPerHour, 14

lockFile, 18

makeDir, 18

makePath, 18

MET, 14

Meter, 14

Mile, 14

MilePerHour, 14

milih2kmh, 18

MlMinKg, 14

MPerSec, 14

ms2kmh, 18

openSerial, 19

pathExist, 19

Phys, 13

physConvert, 19

Power, 13

PowerAir, 14

Procent, 14

Radian, 14

s2HB, 19

s2LB, 19

setDateTime, 20

Speed, 13

typeName, 20

u16string\_to\_utf8, 20

Unit, 14

Unknown, 14

utf8\_to\_u16string, 20

utf8\_to\_wstring, 20

Watt, 14

wstring\_to\_utf8, 21

View

RehabUserProtocolsViewMenu, 119

Watt

Utils, 14

WaveFile, 165

getBitsPerSample, 165

getChannels, 166

getData, 166

getFormat, 166

getSamplesPerSec, 166

WaveFile, 165

WBCElement, 167

Weight

DbConnector, 36

Welcome

SoundInterface, 143

Widget, 168

Active, 169

Disabled, 169

getHeight, 169

getParent, 170

getState, 170

getStateName, 170

getWidth, 170

getX, 170

getY, 170

isSelected, 171

isVisible, 171

Passive, 169

processEvent, 171

setHeight, 171

setParent, 171

setState, 171

setVisible, 172

setWidth, 172

setX, 172

setY, 172

State, 169

Widget, 169

WidgetBreadCrumbs, 173

processEvent, 173

WidgetBreadCrumbs, 173

WidgetImageButton, 175

Click, 176

Clicker, 176

Event, 175

Generator, 176

isSet, 176

Press, 176

PressRelease, 176

processEvent, 177

Release, 176

setActive, 177

setDisabled, 177

setPassive, 177

setRepeat, 177

setSound, 177

setState, 177

Trigger, 176

Type, 176

WidgetImageButton, 176

WidgetProfil, 179

WidgetProfil, 179

WidgetRectangle, 180

setColor, 180

WidgetRectangle, 180

WidgetText, 182

getAlign, 183

getColor, 183

getColorDisabled, 183

getText, 183

setAlign, 183

setText, 184

WidgetText, 182

WidgetTextField, 185

getIdCol, 186

getIdRow, 186

getText, 186

getTextId, 187

processEvent, 187

setCheckFunc, 187

setFontSize, 187

setIdCol, 187

setIdRow, 187

setLimits, 188

setMaxLimit, 188

setPrompt, 188

setText, 188

setUpdateFunc, 188

WidgetTextField, 186

WiFiRemote, 190

isConnected, 190

send, 190

wstring\_to\_utf8

Utils, 21

WTFStyle, 191

YellowTitle

ModalCommonDialog, 77

Yes

ModalCommonDialog, 78

YesCancel

ModalCommonDialog, 78

YesNoCancel

ModalCommonDialog, 78

Zastup

Event, 48