

PiQuantum the raspbery pi powered quantum computer simulator  
Alpha1.0

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
2.1	Data Structures . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Data Structure Documentation</b>	<b>7</b>
4.1	Alarm Class Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Constructor & Destructor Documentation . . . . .	8
4.1.2.1	Alarm() . . . . .	8
4.2	Button Class Reference . . . . .	8
4.2.1	Constructor & Destructor Documentation . . . . .	8
4.2.1.1	Button() [1/2] . . . . .	8
4.2.1.2	~Button() . . . . .	9
4.2.1.3	Button() [2/2] . . . . .	9
4.2.2	Member Function Documentation . . . . .	9
4.2.2.1	get_position() [1/2] . . . . .	9
4.2.2.2	get_position() [2/2] . . . . .	9
4.2.2.3	get_state() [1/2] . . . . .	9
4.2.2.4	get_state() [2/2] . . . . .	9
4.2.2.5	rgb() . . . . .	9

4.2.3	Friends And Related Function Documentation . . . . .	10
4.2.3.1	InputOutput . . . . .	10
4.3	Hadamard Class Reference . . . . .	10
4.3.1	Constructor & Destructor Documentation . . . . .	11
4.3.1.1	Hadamard() . . . . .	11
4.4	InputOutput Class Reference . . . . .	12
4.4.1	Detailed Description . . . . .	13
4.4.2	Constructor & Destructor Documentation . . . . .	13
4.4.2.1	InputOutput() . . . . .	13
4.4.3	Member Function Documentation . . . . .	13
4.4.3.1	deregister_button() . . . . .	13
4.4.3.2	deregister_led() . . . . .	13
4.4.3.3	print() . . . . .	13
4.4.3.4	read_button_states() . . . . .	13
4.4.3.5	register_button() . . . . .	14
4.4.3.6	register_led() . . . . .	14
4.4.3.7	set_leds() . . . . .	14
4.5	Led Class Reference . . . . .	15
4.5.1	Detailed Description . . . . .	15
4.5.2	Constructor & Destructor Documentation . . . . .	15
4.5.2.1	Led() [1/2] . . . . .	15
4.5.2.2	Led() [2/2] . . . . .	15
4.5.3	Member Function Documentation . . . . .	16
4.5.3.1	get_positions() . . . . .	16
4.5.3.2	get_rgb() . . . . .	16
4.5.3.3	set_rgb() . . . . .	16
4.6	Operator Class Reference . . . . .	16
4.6.1	Constructor & Destructor Documentation . . . . .	17
4.6.1.1	Operator() . . . . .	17
4.6.2	Member Function Documentation . . . . .	17

4.6.2.1	<a href="#">get_num_qubits()</a>	18
4.6.2.2	<a href="#">print()</a>	18
4.6.2.3	<a href="#">selected()</a>	18
4.6.2.4	<a href="#">set_btn()</a>	18
4.6.3	<a href="#">Field Documentation</a>	18
4.6.3.1	<a href="#">btn_ptr</a>	18
4.6.3.2	<a href="#">matrix</a>	18
4.6.3.3	<a href="#">name</a>	18
4.6.3.4	<a href="#">num_qubits</a>	19
4.7	<a href="#">PIN Class Reference</a>	19
4.7.1	<a href="#">Field Documentation</a>	19
4.7.1.1	<a href="#">LE</a>	19
4.7.1.2	<a href="#">OE</a>	19
4.7.1.3	<a href="#">SHLD</a>	19
4.8	<a href="#">Position Struct Reference</a>	20
4.8.1	<a href="#">Detailed Description</a>	20
4.8.2	<a href="#">Field Documentation</a>	20
4.8.2.1	<a href="#">chip</a>	20
4.8.2.2	<a href="#">line</a>	20
4.9	<a href="#">Qubit Class Reference</a>	20
4.9.1	<a href="#">Constructor &amp; Destructor Documentation</a>	21
4.9.1.1	<a href="#">Qubit()</a>	21
4.9.2	<a href="#">Member Function Documentation</a>	21
4.9.2.1	<a href="#">check_uptodate()</a>	21
4.9.2.2	<a href="#">get_one_amp()</a>	21
4.9.2.3	<a href="#">get_phase()</a>	22
4.9.2.4	<a href="#">get_zero_amp()</a>	22
4.9.2.5	<a href="#">selected()</a>	22
4.9.2.6	<a href="#">set_amps()</a> [1/2]	22
4.9.2.7	<a href="#">set_amps()</a> [2/2]	22

4.9.2.8	<a href="#">set_led()</a>	22
4.9.2.9	<a href="#">set_one()</a>	22
4.9.2.10	<a href="#">set_phase()</a>	23
4.9.2.11	<a href="#">set_uptodate()</a>	23
4.9.2.12	<a href="#">set_zero()</a>	23
4.10	<a href="#">Qubit::Qubit_state Struct Reference</a>	23
4.10.1	<a href="#">Detailed Description</a>	23
4.10.2	<a href="#">Field Documentation</a>	23
4.10.2.1	<a href="#">one_amp</a>	24
4.10.2.2	<a href="#">phase</a>	24
4.10.2.3	<a href="#">zero_amp</a>	24
4.11	<a href="#">Rotation_X Class Reference</a>	24
4.11.1	<a href="#">Constructor &amp; Destructor Documentation</a>	25
4.11.1.1	<a href="#">Rotation_X()</a>	25
4.12	<a href="#">Rotation_Y Class Reference</a>	26
4.12.1	<a href="#">Constructor &amp; Destructor Documentation</a>	27
4.12.1.1	<a href="#">Rotation_Y()</a>	27
4.13	<a href="#">Rotation_Z Class Reference</a>	27
4.13.1	<a href="#">Constructor &amp; Destructor Documentation</a>	28
4.13.1.1	<a href="#">Rotation_Z()</a>	28
4.14	<a href="#">SpiChannel Class Reference</a>	29
4.14.1	<a href="#">Detailed Description</a>	29
4.14.2	<a href="#">Constructor &amp; Destructor Documentation</a>	29
4.14.2.1	<a href="#">SpiChannel()</a>	29
4.14.3	<a href="#">Member Function Documentation</a>	29
4.14.3.1	<a href="#">change_frequency()</a>	29
4.14.3.2	<a href="#">read()</a>	29
4.14.3.3	<a href="#">write()</a>	30
4.15	<a href="#">State_vector Class Reference</a>	30
4.15.1	<a href="#">Constructor &amp; Destructor Documentation</a>	31

4.15.1.1	State_vector() [1/2]	31
4.15.1.2	State_vector() [2/2]	31
4.15.2	Member Function Documentation	31
4.15.2.1	apply() [1/2]	31
4.15.2.2	apply() [2/2]	32
4.15.2.3	disp()	32
4.15.2.4	disp_cycle()	32
4.15.2.5	display_avg()	32
4.15.2.6	get_num_qubits()	32
4.15.2.7	get_qubit()	32
4.15.2.8	get_size()	33
4.15.2.9	print()	33
4.15.2.10	set_superpos()	33
4.15.2.11	set_vacuum()	33
4.15.3	Field Documentation	33
4.15.3.1	qubits	33
4.16	WiringPi Class Reference	33
4.16.1	Detailed Description	34
4.16.2	Constructor & Destructor Documentation	34
4.16.2.1	WiringPi()	34

<b>5</b>	<b>File Documentation</b>	<b>35</b>
5.1	<a href="#">/home/oliver/Documents/piquantum/src/final/buttons.cpp File Reference</a>	35
5.1.1	Detailed Description	35
5.2	<a href="#">/home/oliver/Documents/piquantum/src/final/buttons.hpp File Reference</a>	36
5.2.1	Detailed Description	36
5.3	<a href="#">/home/oliver/Documents/piquantum/src/final/interface.cpp File Reference</a>	37
5.3.1	Detailed Description	37
5.4	<a href="#">/home/oliver/Documents/piquantum/src/final/interface.hpp File Reference</a>	37
5.4.1	Detailed Description	38
5.5	<a href="#">/home/oliver/Documents/piquantum/src/final/io-test.cpp File Reference</a>	39
5.5.1	Function Documentation	39
5.5.1.1	main()	39
5.6	<a href="#">/home/oliver/Documents/piquantum/src/final/io.cpp File Reference</a>	40
5.6.1	Detailed Description	40
5.6.2	Function Documentation	40
5.6.2.1	getInputOutput()	41
5.6.2.2	operator<<()	41
5.7	<a href="#">/home/oliver/Documents/piquantum/src/final/io.hpp File Reference</a>	41
5.7.1	Detailed Description	42
5.7.2	Function Documentation	42
5.7.2.1	getInputOutput()	42
5.7.2.2	operator<<()	43
5.8	<a href="#">/home/oliver/Documents/piquantum/src/final/math_test.cpp File Reference</a>	43
5.8.1	Function Documentation	43
5.8.1.1	delay()	44
5.8.1.2	get_qubit_btn()	44
5.8.1.3	main()	44
5.8.1.4	make_leds_light_up()	44
5.9	<a href="#">/home/oliver/Documents/piquantum/src/final/pin_mappings.hpp File Reference</a>	44
5.10	<a href="#">/home/oliver/Documents/piquantum/src/final/qubit.cpp File Reference</a>	45



5.10.1	Function Documentation	45
5.10.1.1	main()	45
5.11	/home/oliver/Documents/piquantum/src/final/spi.cpp File Reference	46
5.11.1	Detailed Description	46
5.11.2	Function Documentation	46
5.11.2.1	getSpiChannel()	46
5.12	/home/oliver/Documents/piquantum/src/final/spi.hpp File Reference	47
5.12.1	Detailed Description	48
5.12.2	Typedef Documentation	48
5.12.2.1	byte	48
5.12.3	Function Documentation	48
5.12.3.1	getSpiChannel()	48
5.13	/home/oliver/Documents/piquantum/src/final/state.cpp File Reference	49
5.13.1	Detailed Description	49
5.13.2	Variable Documentation	49
5.13.2.1	qubit_disp_cycle	49
5.14	/home/oliver/Documents/piquantum/src/final/state.hpp File Reference	50
5.14.1	Detailed Description	51
5.14.2	Typedef Documentation	51
5.14.2.1	Qubits_type	51
5.14.3	Function Documentation	51
5.14.3.1	I_unit()	51
5.14.4	Variable Documentation	52
5.14.4.1	PI	52
5.15	/home/oliver/Documents/piquantum/src/final/wpi.cpp File Reference	52
5.15.1	Detailed Description	52
5.16	/home/oliver/Documents/piquantum/src/final/wpi.hpp File Reference	53
5.16.1	Detailed Description	53



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Alarm . . . . .	7
InputOutput . . . . .	12
Button . . . . .	8
Led . . . . .	15
Operator . . . . .	16
Hadamard . . . . .	10
Rotation_X . . . . .	24
Rotation_Y . . . . .	26
Rotation_Z . . . . .	27
PIN . . . . .	19
Position . . . . .	20
Qubit . . . . .	20
Qubit::Qubit_state . . . . .	23
SpiChannel . . . . .	29
State_vector . . . . .	30
WiringPi . . . . .	33



## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">Alarm</a>	
Interrupts in Linux . . . . .	7
<a href="#">Button</a> . . . . .	8
<a href="#">Hadamard</a> . . . . .	10
<a href="#">InputOutput</a>	
Class for reading buttons and writing to LEDs . . . . .	12
<a href="#">Led</a>	
RGB <a href="#">Led</a> class . . . . .	15
<a href="#">Operator</a> . . . . .	16
<a href="#">PIN</a> . . . . .	19
<a href="#">Position</a>	
Chip/line structure . . . . .	20
<a href="#">Qubit</a> . . . . .	20
<a href="#">Qubit::Qubit_state</a>	
Define a struct to hold the qubit properties . . . . .	23
<a href="#">Rotation_X</a> . . . . .	24
<a href="#">Rotation_Y</a> . . . . .	26
<a href="#">Rotation_Z</a> . . . . .	27
<a href="#">SpiChannel</a>	
<a href="#">SpiChannel</a> class . . . . .	29
<a href="#">State_vector</a> . . . . .	30
<a href="#">WiringPi</a>	
Class for initialising wiringPi . . . . .	33



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

/home/oliver/Documents/piquantum/src/final/buttons.cpp	35
/home/oliver/Documents/piquantum/src/final/buttons.hpp	36
/home/oliver/Documents/piquantum/src/final/interface.cpp	37
/home/oliver/Documents/piquantum/src/final/interface.hpp	37
/home/oliver/Documents/piquantum/src/final/io-test.cpp	39
/home/oliver/Documents/piquantum/src/final/io.cpp	40
/home/oliver/Documents/piquantum/src/final/io.hpp	41
/home/oliver/Documents/piquantum/src/final/math_test.cpp	43
/home/oliver/Documents/piquantum/src/final/pin_mappings.hpp	44
/home/oliver/Documents/piquantum/src/final/qubit.cpp	45
/home/oliver/Documents/piquantum/src/final/spi.cpp	46
/home/oliver/Documents/piquantum/src/final/spi.hpp	47
/home/oliver/Documents/piquantum/src/final/state.cpp	
State vector class	49
/home/oliver/Documents/piquantum/src/final/state.hpp	
Header for state vector class and operators	50
/home/oliver/Documents/piquantum/src/final/wpi.cpp	52
/home/oliver/Documents/piquantum/src/final/wpi.hpp	53





## Chapter 4

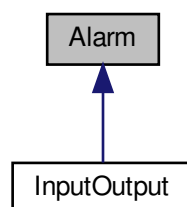
# Data Structure Documentation

### 4.1 Alarm Class Reference

Interrupts in Linux.

```
#include <io.hpp>
```

Inheritance diagram for Alarm:



#### Public Member Functions

- [Alarm](#) (int delay\_us)

#### 4.1.1 Detailed Description

Interrupts in Linux.

This class implements the SIGALARM function in Linux. SIGALARM can be setup to call a function at regular intervals. The function (interrupt) does not have an implementation in this class. It is defined in the derived [Input↔Output](#) class. The constructor sets up the SIGALARM (defining the period, etc.)

## 4.1.2 Constructor & Destructor Documentation

### 4.1.2.1 Alarm()

```
Alarm::Alarm (
    int delay_us )
```

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

- [/home/oliver/Documents/piquantum/src/final/io.hpp](#)
- [/home/oliver/Documents/piquantum/src/final/io.cpp](#)

## 4.2 Button Class Reference

```
#include <buttons.hpp>
```

### Public Member Functions

- [Button](#) ([Position](#) position)
- [~Button](#) ()
- int [get\\_state](#) ()
- [std::vector< double >](#) [rgb](#) ()
- [Position](#) [get\\_position](#) ()
- [Button](#) ([Position](#) position)
- int [get\\_state](#) ()
- [Position](#) [get\\_position](#) ()

### Friends

- class [InputOutput](#)

## 4.2.1 Constructor & Destructor Documentation

### 4.2.1.1 Button() [1/2]

```
Button::Button (
    Position position )
```

#### 4.2.1.2 ~Button()

```
Button::~~Button ( )
```

#### 4.2.1.3 Button() [2/2]

```
Button::Button (
    Position position )
```

### 4.2.2 Member Function Documentation

#### 4.2.2.1 get\_position() [1/2]

```
Position Button::get_position ( ) [inline]
```

#### 4.2.2.2 get\_position() [2/2]

```
Position Button::get_position ( ) [inline]
```

#### 4.2.2.3 get\_state() [1/2]

```
int Button::get_state ( )
```

#### 4.2.2.4 get\_state() [2/2]

```
int Button::get_state ( )
```

#### 4.2.2.5 rgb()

```
std::vector<double> Button::rgb ( ) [inline]
```

### 4.2.3 Friends And Related Function Documentation

#### 4.2.3.1 InputOutput

`InputOutput` [friend]

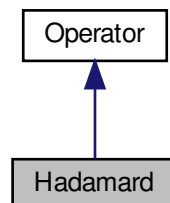
The documentation for this class was generated from the following files:

- [/home/oliver/Documents/piquantum/src/final/buttons.hpp](#)
- [/home/oliver/Documents/piquantum/src/final/interface.hpp](#)
- [/home/oliver/Documents/piquantum/src/final/buttons.cpp](#)
- [/home/oliver/Documents/piquantum/src/final/interface.cpp](#)

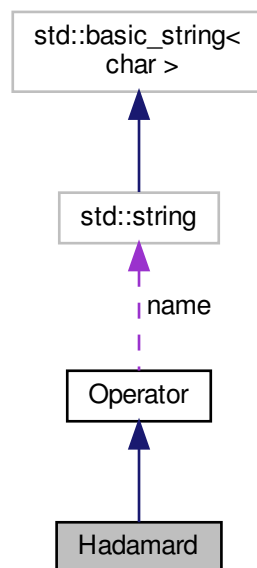
## 4.3 Hadamard Class Reference

```
#include <state.hpp>
```

Inheritance diagram for Hadamard:



Collaboration diagram for Hadamard:



## Public Member Functions

- [Hadamard](#) (std::shared\_ptr< [Button](#) > btn\_ptr\_in=nullptr, int num\_qubits\_act\_on=1)

## Additional Inherited Members

### 4.3.1 Constructor & Destructor Documentation

#### 4.3.1.1 Hadamard()

```

Hadamard::Hadamard (
    std::shared_ptr< Button > btn_ptr_in = nullptr,
    int num_qubits_act_on = 1 )

```

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

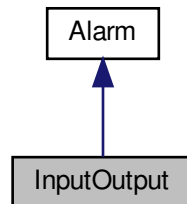
- /home/oliver/Documents/piquantum/src/final/[state.hpp](#)
- /home/oliver/Documents/piquantum/src/final/[state.cpp](#)

## 4.4 InputOutput Class Reference

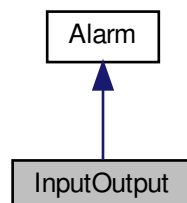
Class for reading buttons and writing to LEDs.

```
#include <io.hpp>
```

Inheritance diagram for InputOutput:



Collaboration diagram for InputOutput:



### Public Member Functions

- `InputOutput` (`std::shared_ptr< SpiChannel > spi`)
- `void print ()`
- `int set_leds (std::vector< byte > data)`  
*Send data to the LEDs.*
- `std::vector< byte > read_button_states (int num)`  
*Read all the button states.*
- `int register_led (Led *led)`  
*Register and deregister LEDs and buttons.*
- `void deregister_led (int id)`
- `int register_button (Button *btn)`
- `void deregister_button (int id)`

#### 4.4.1 Detailed Description

Class for reading buttons and writing to LEDs.

There are two main parts to the class. The first part is two arrays of pointers (one for button objects and one for LEDs) which are 'registered' with this class automatically when they are instantiated.

The second part is a function, called `interrupt()`, which is called at regular intervals by the SIGALARM signal (a linux thing). This function is responsible for updating the states of the LEDs and reading the state of all the buttons.

The LEDs are controlled using a pulse width modulation type scheme, which `interrupt()` simulates. I keeps track of a counter, which is incremented by 1 every time `interrupt()` is called. When it gets to period, it is reset to zero. As the counter ranges through 0 to period, each registered LED object is queried to get an RGB values (between 0 and 1). If this value exceeds counter/period (also between 0 and 1), then the LED is turned off. The LEDs are all switched back on when the counter is reset to zero

The user program never uses this program. It is implicitly instantiated when [Led](#) and [Button](#) objects are instantiated.

#### 4.4.2 Constructor & Destructor Documentation

##### 4.4.2.1 InputOutput()

```
InputOutput::InputOutput (
    std::shared_ptr< SpiChannel > spi )
```

#### 4.4.3 Member Function Documentation

##### 4.4.3.1 deregister\_button()

```
void InputOutput::deregister_button (
    int id )
```

##### 4.4.3.2 deregister\_led()

```
void InputOutput::deregister_led (
    int id )
```

##### 4.4.3.3 print()

```
void InputOutput::print ( )
```

##### 4.4.3.4 read\_button\_states()

```
std::vector< unsigned char > InputOutput::read_button_states (
    int num )
```

Read all the button states.

Read a number

## Parameters

<i>num</i>	of bytes from the button shift registers and return it in a standard vector. The SIGALARM function (func) calls this function to update the state of all the button objects
------------	---

## 4.4.3.5 register\_button()

```
int InputOutput::register_button (
    Button * btn )
```

## 4.4.3.6 register\_led()

```
int InputOutput::register_led (
    Led * led )
```

Register and deregister LEDs and buttons.

Register a objects with the driver. The function stores a pointer to the object. There's a problem still to solve: how to remove the entry when the [Led](#) object no longer exists. That can be fixed later.

## 4.4.3.7 set\_leds()

```
int InputOutput::set_leds (
    std::vector< byte > data )
```

Send data to the LEDs.

Turn on an LED via the external display driver TLC591x.

Send a std::vector of bytes to the LED drivers. This function is called repeatedly in the func SIGALARM function to update the state of the LEDs

On power on, the chip (TLC591x) is in normal mode which means that the clocked bytes sent to the chip set which LEDs are on and which are off (as opposed to setting the current of the LEDs).

This function assumes that a number of the TLC591 chips are connected together. Data is sent via SPI to the first chip and passed along the chain to other devices.

To write to the device:

1) Write a number of bytes to the SPI port, equal to the number of chips connected together 2) Momentarily set the LE(ED1) pin to latch the data onto the output register. 3) Bring the OE(ED2) pin low to enable the current sinking to turn on the LEDs.

See the timing diagram on page 17 of the datasheet for details.

Arguments: a std::vector of the data to write to the display chips

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

- [/home/oliver/Documents/piquantum/src/final/io.hpp](#)
- [/home/oliver/Documents/piquantum/src/final/io.cpp](#)



## 4.5 Led Class Reference

RGB [Led](#) class.

```
#include <interface.hpp>
```

### Public Member Functions

- [Led](#) ([Position](#) r, [Position](#) g, [Position](#) b)
- [Led](#) (std::vector< [Position](#) > pos)
- void [set\\_rgb](#) (double red, double green, double blue)
- std::vector< double > [get\\_rgb](#) ()
- std::vector< [Position](#) > [get\\_positions](#) ()

### 4.5.1 Detailed Description

RGB [Led](#) class.

Objects of this class represent RGB LEDs. When the class is instantiated, it creates an [InputOutput](#) class driver automatically (if one does not already exist) and then registers itself with it, so that its RGB values are automatically sent to the device.

The constructor takes three arguments which are the positions on the driver chips of the R, G and B lines. They are specified in the form {chip, line}, where chip is the chip number and line is the line number on that chip.

RGB values are updated using the [set\\_rgb](#) function. RGB values are between 0 and 1, with 0 representing off and 1 representing maximum brightness. The new state will be written immediately to the LEDs. To turn the LED off write RGB values of 0.

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 [Led\(\)](#) [1/2]

```
Led::Led (
    Position r,
    Position g,
    Position b )
```

#### 4.5.2.2 [Led\(\)](#) [2/2]

```
Led::Led (
    std::vector< Position > pos )
```

### 4.5.3 Member Function Documentation

#### 4.5.3.1 `get_positions()`

```
std::vector< Position > Led::get_positions ( )
```

#### 4.5.3.2 `get_rgb()`

```
std::vector< double > Led::get_rgb ( )
```

#### 4.5.3.3 `set_rgb()`

```
void Led::set_rgb (
    double red,
    double green,
    double blue )
```

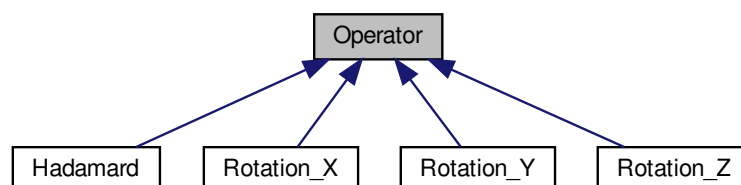
The documentation for this class was generated from the following files:

- [/home/oliver/Documents/piquantum/src/final/interface.hpp](#)
- [/home/oliver/Documents/piquantum/src/final/interface.cpp](#)

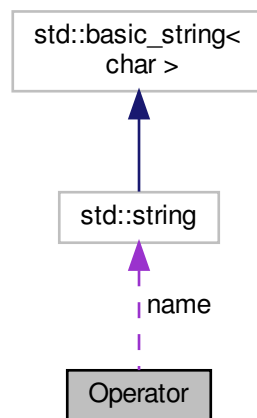
## 4.6 Operator Class Reference

```
#include <state.hpp>
```

Inheritance diagram for Operator:



Collaboration diagram for Operator:



### Public Member Functions

- int `get_num_qubits` (void)
- void `print` ()
- void `set_btn` (std::shared\_ptr< `Button` > btn\_ptr\_in)
- bool `selected` (void)
- `Operator` ()

### Data Fields

- int `num_qubits`
- std::string `name`
- Eigen::Matrix2cd `matrix`
- std::shared\_ptr< `Button` > `btn_ptr`

## 4.6.1 Constructor & Destructor Documentation

### 4.6.1.1 Operator()

```
Operator::Operator ( ) [inline]
```

## 4.6.2 Member Function Documentation

#### 4.6.2.1 `get_num_qubits()`

```
int Operator::get_num_qubits (
    void )
```

#### 4.6.2.2 `print()`

```
void Operator::print (
    void )
```

#### 4.6.2.3 `selected()`

```
bool Operator::selected (
    void )
```

#### 4.6.2.4 `set_btn()`

```
void Operator::set_btn (
    std::shared_ptr< Button > btn_ptr_in )
```

### 4.6.3 Field Documentation

#### 4.6.3.1 `btn_ptr`

```
std::shared_ptr<Button> Operator::btn_ptr
```

#### 4.6.3.2 `matrix`

```
Eigen::Matrix2cd Operator::matrix
```

#### 4.6.3.3 `name`

```
std::string Operator::name
```

#### 4.6.3.4 num\_qubits

```
int Operator::num_qubits
```

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

- </home/oliver/Documents/piquantum/src/final/state.hpp>
- </home/oliver/Documents/piquantum/src/final/state.cpp>

## 4.7 PIN Class Reference

```
#include <pin_mappings.hpp>
```

### Static Public Attributes

- static const int [LE](#) = 0
- static const int [OE](#) = 1
- static const int [SHLD](#) = 23

### 4.7.1 Field Documentation

#### 4.7.1.1 LE

```
const int PIN::LE = 0 [static]
```

#### 4.7.1.2 OE

```
const int PIN::OE = 1 [static]
```

#### 4.7.1.3 SHLD

```
const int PIN::SHLD = 23 [static]
```

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

- [/home/oliver/Documents/piquantum/src/final/pin\\_mappings.hpp](/home/oliver/Documents/piquantum/src/final/pin_mappings.hpp)

## 4.8 Position Struct Reference

Chip/line structure.

```
#include <io.hpp>
```

### Data Fields

- int [chip](#)
- int [line](#)

### 4.8.1 Detailed Description

Chip/line structure.

Store the chip and line number for a particular LED or button.

### 4.8.2 Field Documentation

#### 4.8.2.1 chip

```
int Position::chip
```

#### 4.8.2.2 line

```
int Position::line
```

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

- [/home/oliver/Documents/piquantum/src/final/io.hpp](#)

## 4.9 Qubit Class Reference

```
#include <state.hpp>
```

### Data Structures

- struct [Qubit\\_state](#)  
*define a struct to hold the qubit properties*

## Public Member Functions

- `Qubit` (`std::vector< Position > led_rgb_loc, Position btn_loc, int pos=-1`)
- `void set_led` (`void`)
- `bool selected` (`void`)
- `void set_amps` (`double zero, double one, double phases`)
- `void set_amps` (`const Qubit_state &qubit_vals`)
- `void set_zero` (`double amp`)
- `void set_one` (`double amp`)
- `void set_phase` (`double phi`)
- `void set_uptodate` (`bool true_false`)
- `double get_zero_amp` (`()`)
- `double get_one_amp` (`()`)
- `double get_phase` (`()`)
- `bool check_uptodate` (`()`)

### 4.9.1 Constructor & Destructor Documentation

#### 4.9.1.1 Qubit()

```
Qubit::Qubit (
    std::vector< Position > led_rgb_loc,
    Position btn_loc,
    int pos = -1 )
```

### 4.9.2 Member Function Documentation

#### 4.9.2.1 check\_uptodate()

```
bool Qubit::check_uptodate (
    void )
```

#### 4.9.2.2 get\_one\_amp()

```
double Qubit::get_one_amp (
    void )
```

#### 4.9.2.3 `get_phase()`

```
double Qubit::get_phase (
    void )
```

#### 4.9.2.4 `get_zero_amp()`

```
double Qubit::get_zero_amp (
    void )
```

#### 4.9.2.5 `selected()`

```
bool Qubit::selected (
    void )
```

#### 4.9.2.6 `set_amps()` [1/2]

```
void Qubit::set_amps (
    double zero,
    double one,
    double phases )
```

#### 4.9.2.7 `set_amps()` [2/2]

```
void Qubit::set_amps (
    const Qubit\_state & qubit_vals )
```

#### 4.9.2.8 `set_led()`

```
void Qubit::set_led (
    void )
```

#### 4.9.2.9 `set_one()`

```
void Qubit::set_one (
    double amp )
```



#### 4.9.2.10 set\_phase()

```
void Qubit::set_phase (
    double phi )
```

#### 4.9.2.11 set\_uptodate()

```
void Qubit::set_uptodate (
    bool true_false )
```

#### 4.9.2.12 set\_zero()

```
void Qubit::set_zero (
    double amp )
```

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

- [/home/oliver/Documents/piquantum/src/final/state.hpp](#)
- [/home/oliver/Documents/piquantum/src/final/state.cpp](#)

## 4.10 Qubit::Qubit\_state Struct Reference

define a struct to hold the qubit properties

```
#include <state.hpp>
```

### Data Fields

- double [zero\\_amp](#)
- double [one\\_amp](#)
- double [phase](#)

#### 4.10.1 Detailed Description

define a struct to hold the qubit properties

#### 4.10.2 Field Documentation

#### 4.10.2.1 one\_amp

```
double Qubit::Qubit_state::one_amp
```

#### 4.10.2.2 phase

```
double Qubit::Qubit_state::phase
```

#### 4.10.2.3 zero\_amp

```
double Qubit::Qubit_state::zero_amp
```

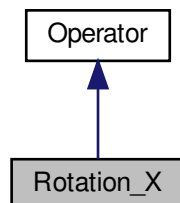
The documentation for this struct was generated from the following file:

- </home/oliver/Documents/piquantum/src/final/state.hpp>

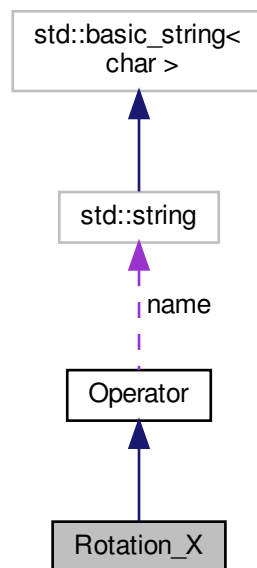
## 4.11 Rotation\_X Class Reference

```
#include <state.hpp>
```

Inheritance diagram for Rotation\_X:



Collaboration diagram for Rotation\_X:



## Public Member Functions

- [Rotation\\_X](#) (`std::shared_ptr< Button > btn_ptr_in=nullptr`, `int num_qubits_act_on=1`, `double theta=PI`)

## Additional Inherited Members

### 4.11.1 Constructor & Destructor Documentation

#### 4.11.1.1 Rotation\_X()

```

Rotation_X::Rotation_X (
    std::shared_ptr< Button > btn_ptr_in = nullptr,
    int num_qubits_act_on = 1,
    double theta = PI )

```

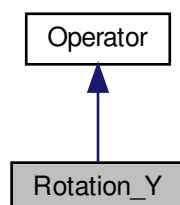
The documentation for this class was generated from the following files:

- `/home/oliver/Documents/piquantum/src/final/state.hpp`
- `/home/oliver/Documents/piquantum/src/final/state.cpp`

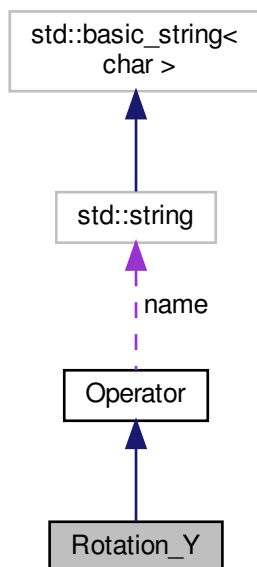
## 4.12 Rotation\_Y Class Reference

```
#include <state.hpp>
```

Inheritance diagram for Rotation\_Y:



Collaboration diagram for Rotation\_Y:



### Public Member Functions

- [Rotation\\_Y](#) (std::shared\_ptr< [Button](#) > btn\_ptr\_in=nullptr, int num\_qubits\_act\_on=1, double theta=[PI](#))

## Additional Inherited Members

### 4.12.1 Constructor & Destructor Documentation

#### 4.12.1.1 Rotation\_Y()

```
Rotation_Y::Rotation_Y (
    std::shared_ptr< Button > btn_ptr_in = nullptr,
    int num_qubits_act_on = 1,
    double theta = PI )
```

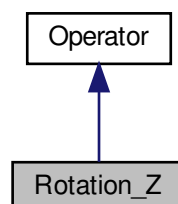
The documentation for this class was generated from the following files:

- </home/oliver/Documents/piquantum/src/final/state.hpp>
- </home/oliver/Documents/piquantum/src/final/state.cpp>

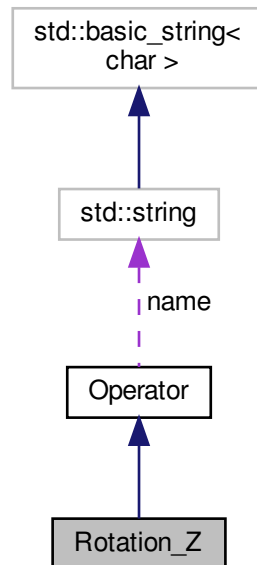
## 4.13 Rotation\_Z Class Reference

```
#include <state.hpp>
```

Inheritance diagram for Rotation\_Z:



Collaboration diagram for `Rotation_Z`:



## Public Member Functions

- [Rotation\\_Z](#) (`std::shared_ptr< Button > btn_ptr_in=nullptr, int num_qubits_act_on=1, double theta=PI`)

## Additional Inherited Members

### 4.13.1 Constructor & Destructor Documentation

#### 4.13.1.1 Rotation\_Z()

```

Rotation_Z::Rotation_Z (
    std::shared_ptr< Button > btn_ptr_in = nullptr,
    int num_qubits_act_on = 1,
    double theta = PI )

```

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

- `/home/oliver/Documents/piquantum/src/final/state.hpp`
- `/home/oliver/Documents/piquantum/src/final/state.cpp`

## 4.14 SpiChannel Class Reference

[SpiChannel](#) class.

```
#include <spi.hpp>
```

### Public Member Functions

- [SpiChannel](#) ()
- void [change\\_frequency](#) (int frequency)
- std::vector< [byte](#) > [read](#) (int num\_bytes)  
*Read spi data.*
- void [write](#) (const std::vector< [byte](#) > &write)  
*Write spi data.*

### 4.14.1 Detailed Description

[SpiChannel](#) class.

This class provides SPI read/write functions on a particular SPI channel using the wiringPi SPI library. The channel is fixed for a particular object when it is constructed, but the frequency can be modified (not sure if that's very helpful).

Use case: as part of an SPI device class to actually read and write data to the device. Other details (such as managing chip selects) would be handled by the higher level class.

### 4.14.2 Constructor & Destructor Documentation

#### 4.14.2.1 SpiChannel()

```
SpiChannel::SpiChannel ( )
```

### 4.14.3 Member Function Documentation

#### 4.14.3.1 change\_frequency()

```
void SpiChannel::change_frequency (
    int frequency )
```

#### 4.14.3.2 read()

```
std::vector< byte > SpiChannel::read (
    int num_bytes )
```

Read spi data.

Function for reading

## Parameters

<i>num_bytes</i>	from an spi device. Data is returned as a standard vector.
------------------	--

## 4.14.3.3 write()

```
void SpiChannel::write (
    const std::vector< byte > & write )
```

Write spi data.

Function for writing

## Parameters

<i>write</i>	from an spi device. Data is returned as a standard vector.
--------------	--

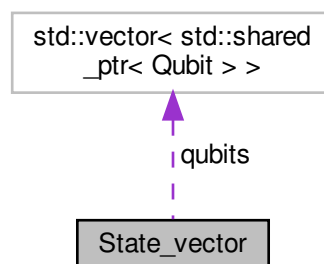
The documentation for this class was generated from the following files:

- </home/oliver/Documents/piquantum/src/final/spi.hpp>
- </home/oliver/Documents/piquantum/src/final/spi.cpp>

## 4.15 State\_vector Class Reference

```
#include <state.hpp>
```

Collaboration diagram for State\_vector:





## Public Member Functions

- void `display_avg` (`Qubits_type` &`qubits`, const Eigen::VectorXcd &`vect`)
- int `get_num_qubits` (void)
- int `get_size` (void)
- void `print` (void)
- `State_vector` ()
- `State_vector` (int `num`, std::vector< std::vector< `Position` > > `qubit_leds`, std::vector< `Position` > `qubit_btns`)
- int `get_qubit` (int `time`=1)
- void `set_vacuum` ()
- void `set_superpos` ()
- void `apply` (const `Operator` &`op`, int `qubit`)
- void `apply` (const `Operator` &`op`, int `ctrl`, int `targ`)
- void `disp` (void)
- int `disp_cycle` (int `n`=0)

## Data Fields

- `Qubits_type` `qubits`

### 4.15.1 Constructor & Destructor Documentation

#### 4.15.1.1 `State_vector()` [1/2]

```
State_vector::State_vector ( ) [inline]
```

#### 4.15.1.2 `State_vector()` [2/2]

```
State_vector::State_vector (
    int num,
    std::vector< std::vector< Position > > qubit_leds,
    std::vector< Position > qubit_btns )
```

### 4.15.2 Member Function Documentation

#### 4.15.2.1 `apply()` [1/2]

```
void State_vector::apply (
    const Operator & op,
    int qubit )
```

#### 4.15.2.2 `apply()` [2/2]

```
void State_vector::apply (
    const Operator & op,
    int ctrl,
    int targ )
```

#### 4.15.2.3 `disp()`

```
void State_vector::disp (
    void )
```

#### 4.15.2.4 `disp_cycle()`

```
int State_vector::disp_cycle (
    int n = 0 )
```

#### 4.15.2.5 `display_avg()`

```
void State_vector::display_avg (
    Qubits_type & qubits,
    const Eigen::VectorXcd & vect )
```

#### 4.15.2.6 `get_num_qubits()`

```
int State_vector::get_num_qubits (
    void )
```

#### 4.15.2.7 `get_qubit()`

```
int State_vector::get_qubit (
    int time = 1 )
```

#### 4.15.2.8 get\_size()

```
int State_vector::get_size (
    void )
```

#### 4.15.2.9 print()

```
void State_vector::print (
    void )
```

#### 4.15.2.10 set\_superpos()

```
void State_vector::set_superpos (
    void )
```

#### 4.15.2.11 set\_vacuum()

```
void State_vector::set_vacuum (
    void )
```

### 4.15.3 Field Documentation

#### 4.15.3.1 qubits

```
Qubits_type State_vector::qubits
```

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

- [/home/oliver/Documents/piquantum/src/final/state.hpp](#)
- [/home/oliver/Documents/piquantum/src/final/state.cpp](#)

## 4.16 WiringPi Class Reference

Class for initialising wiringPi.

```
#include <wpi.hpp>
```

## Public Member Functions

- [WiringPi](#) ()

### 4.16.1 Detailed Description

Class for initialising wiringPi.

Everything that uses input/output using wiringPi should include a [WiringPi](#) private data member. That will make sure that the necessary setup routines get called before anything starts using input/output functions

### 4.16.2 Constructor & Destructor Documentation

#### 4.16.2.1 WiringPi()

```
WiringPi::WiringPi ( ) [inline]
```

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

- [/home/oliver/Documents/piquantum/src/final/wpi.hpp](#)
- [/home/oliver/Documents/piquantum/src/final/wpi.cpp](#)

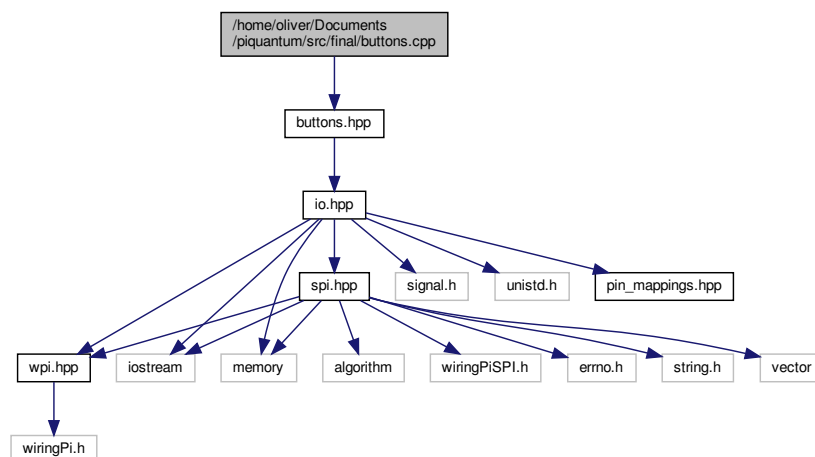
## Chapter 5

# File Documentation

### 5.1 /home/oliver/Documents/piquantum/src/final/buttons.cpp File Reference

```
#include "buttons.hpp"
```

Include dependency graph for buttons.cpp:



#### 5.1.1 Detailed Description

##### Authors

J Scott, O Thomas

##### Date

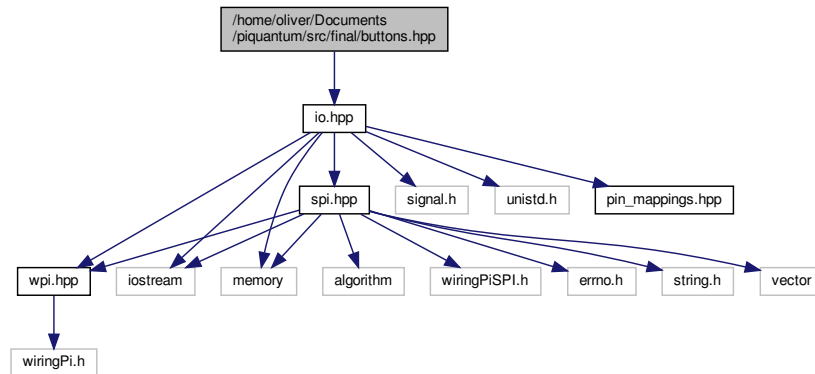
Feb 2019

LED control.

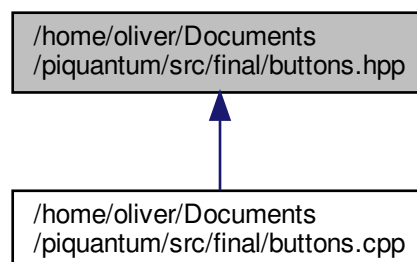
## 5.2 /home/oliver/Documents/piquantum/src/final/buttons.hpp File Reference

```
#include "io.hpp"
```

Include dependency graph for buttons.hpp:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [Button](#)

### 5.2.1 Detailed Description

#### Authors

J Scott, O Thomas

#### Date

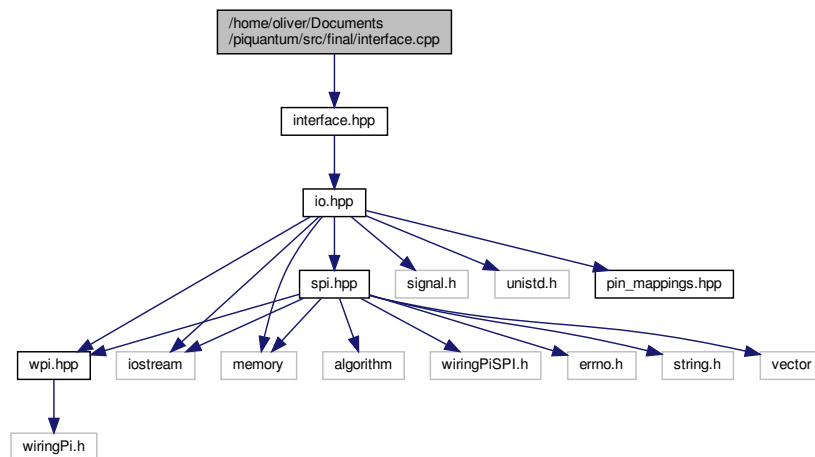
Feb 2019

Header file for Rasp Pi button I/O using SPI.

## 5.3 /home/oliver/Documents/piquantum/src/final/interface.cpp File Reference

```
#include "interface.hpp"
```

Include dependency graph for interface.cpp:



### 5.3.1 Detailed Description

#### Authors

J Scott, O Thomas

#### Date

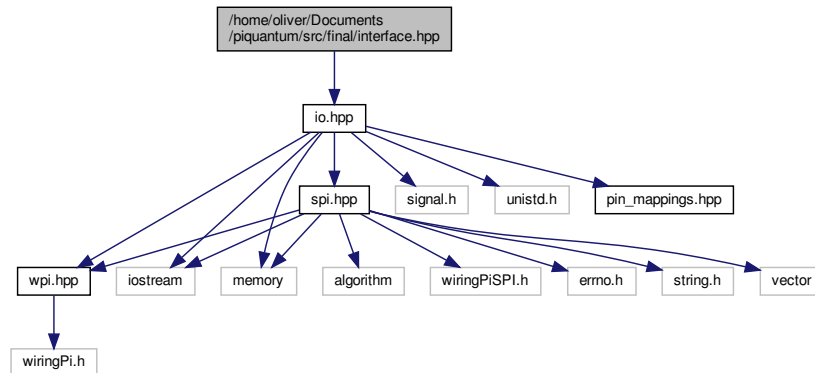
Feb 2019

LED control.

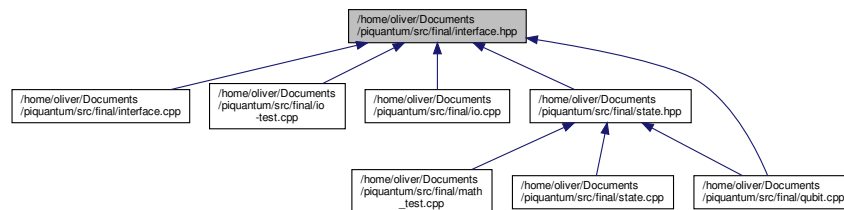
## 5.4 /home/oliver/Documents/piquantum/src/final/interface.hpp File Reference

```
#include "io.hpp"
```

Include dependency graph for `interface.hpp`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [Led](#)  
*RGB [Led](#) class.*
- class [Button](#)

### 5.4.1 Detailed Description

#### Authors

J Scott, O Thomas

#### Date

Feb 2019

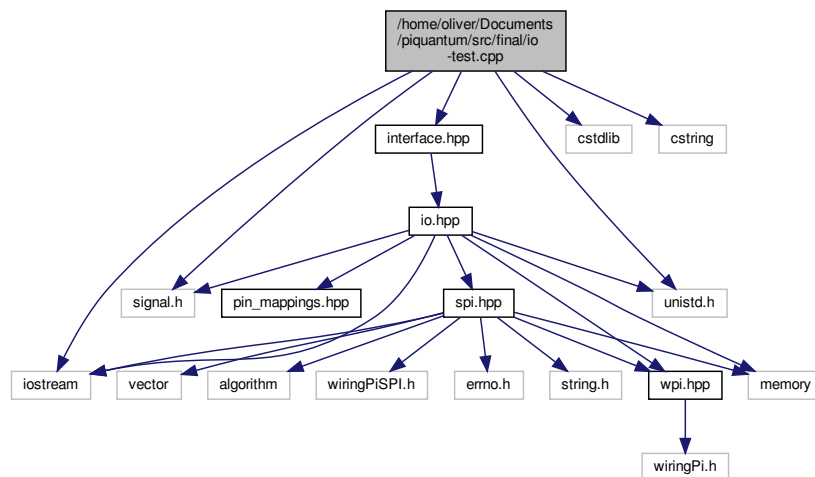
LED and [Button](#) control.



## 5.5 /home/oliver/Documents/piquantum/src/final/io-test.cpp File Reference

```
#include <iostream>
#include "interface.hpp"
#include <cstdlib>
#include <cstring>
#include <signal.h>
#include <unistd.h>
```

Include dependency graph for io-test.cpp:



### Functions

- `int main ()`

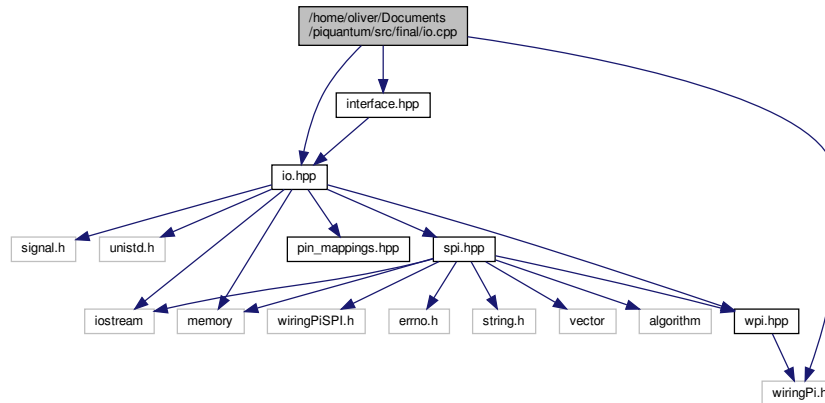
#### 5.5.1 Function Documentation

##### 5.5.1.1 main()

```
int main (
    void )
```

## 5.6 /home/oliver/Documents/piquantum/src/final/io.cpp File Reference

```
#include <wiringPi.h>
#include "io.hpp"
#include "interface.hpp"
Include dependency graph for io.cpp:
```



### Functions

- `std::shared_ptr< InputOutput > getInputOutput ()`  
*Wrapper to return [InputOutput](#) class.*
- `std::ostream & operator<< (std::ostream &stream, Position lines)`  
*Print the chip/lines for an LED.*

### 5.6.1 Detailed Description

#### Authors

J Scott, O Thomas

#### Date

Feb 2019

IO control

### 5.6.2 Function Documentation

## 5.6.2.1 getInputOutput()

```
std::shared_ptr<InputOutput> getInputOutput ( )
```

Wrapper to return [InputOutput](#) class.

Return a shared\_ptr object to an [InputOutput](#). Pass a channel to indicate which SPI channel to use

## 5.6.2.2 operator&lt;&lt;()

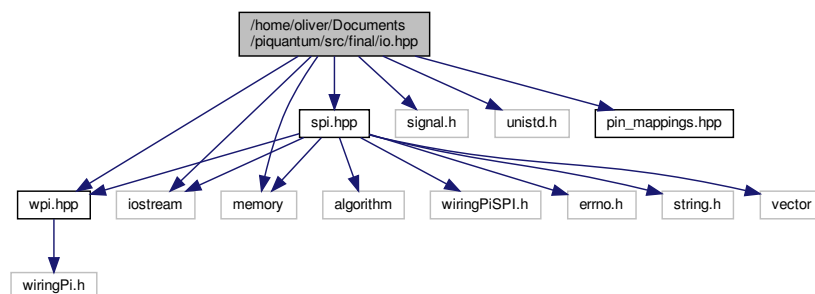
```
std::ostream& operator<< (
    std::ostream & stream,
    Position lines )
```

Print the chip/lines for an LED.

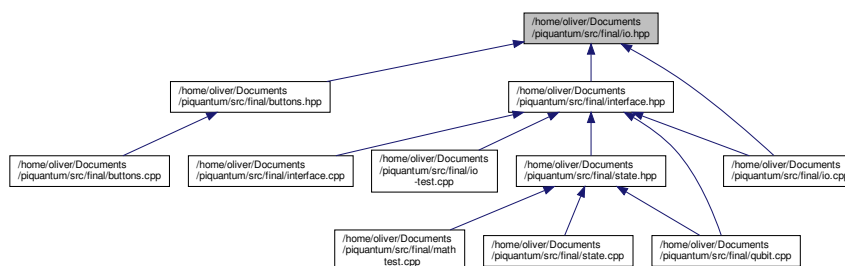
Useful for easily printing out [Position](#) structs in an easy to read format.

## 5.7 /home/oliver/Documents/piquantum/src/final/io.hpp File Reference

```
#include "wpi.hpp"
#include <signal.h>
#include <unistd.h>
#include <iostream>
#include <memory>
#include "spi.hpp"
#include "pin_mappings.hpp"
Include dependency graph for io.hpp:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [Alarm](#)  
*Interrupts in Linux.*
- class [InputOutput](#)  
*Class for reading buttons and writing to LEDs.*
- struct [Position](#)  
*Chip/line structure.*

## Functions

- `std::shared_ptr< class InputOutput > getInputOutput ( )`  
*Wrapper to return [InputOutput](#) class.*
- `std::ostream & operator<< (std::ostream &stream, Position lines)`  
*Print the chip/lines for an LED.*

### 5.7.1 Detailed Description

#### Authors

J Scott, O Thomas

#### Date

Feb 2019

IO control.

### 5.7.2 Function Documentation

#### 5.7.2.1 `getInputOutput()`

```
std::shared_ptr<class InputOutput> getInputOutput ( )
```

Wrapper to return [InputOutput](#) class.

Return a `shared_ptr` object to an [InputOutput](#). Functions should use this function to get a shared pointer to the [InputOutput](#) object. This function ensures that there is only ever one. Functions should not instantiate [InputOutput](#) objects directly

Return a `shared_ptr` object to an [InputOutput](#). Pass a channel to indicate which SPI channel to use

## 5.7.2.2 operator&lt;&lt;()

```
std::ostream& operator<< (
    std::ostream & stream,
    Position lines )
```

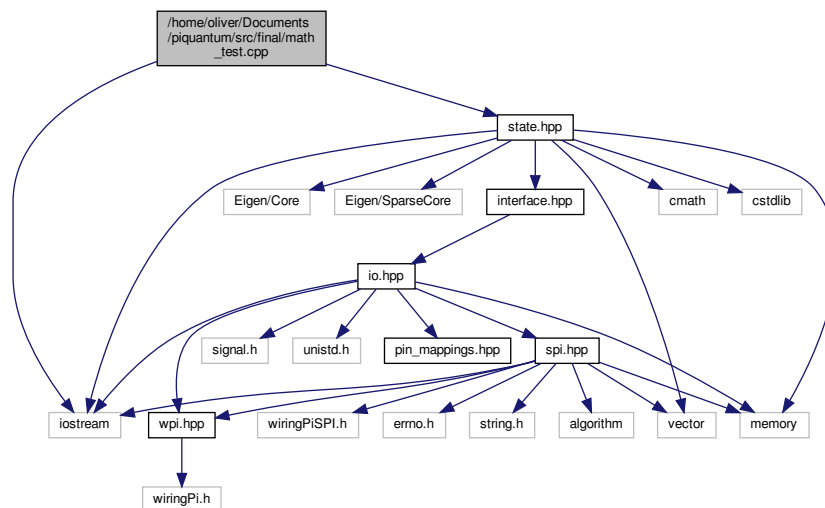
Print the chip/lines for an LED.

Useful for easily printing out `Position` structs in an easy to read format.

## 5.8 /home/oliver/Documents/piquantum/src/final/math\_test.cpp File Reference

```
#include <iostream>
#include "state.hpp"
```

Include dependency graph for math\_test.cpp:



## Functions

- `std::string delay ()`
- `void make_leds_light_up (const std::vector< State_vector::Qubit_states > &states, Led &led0, Led &led1, Led &led2, Led &led3)`
- `int get_qubit_btn (Button &btn_q0, Button &btn_q1, Button &btn_q2, Button &btn_q3)`
- `int main (void)`

## 5.8.1 Function Documentation

### 5.8.1.1 delay()

```
std::string delay ( )
```

### 5.8.1.2 get\_qubit\_btn()

```
int get_qubit_btn (
    Button & btn_q0,
    Button & btn_q1,
    Button & btn_q2,
    Button & btn_q3 )
```

### 5.8.1.3 main()

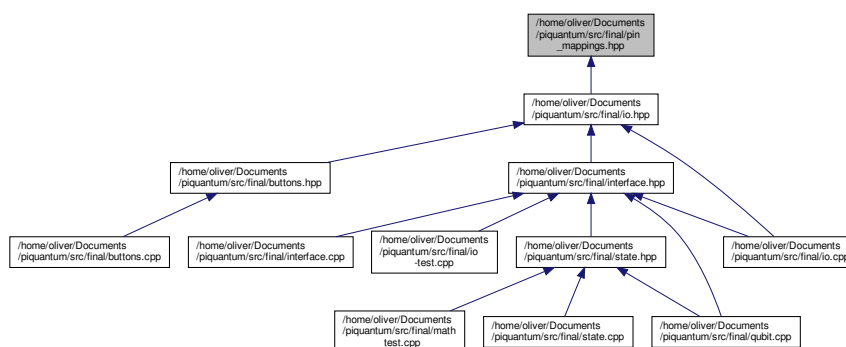
```
int main (
    void )
```

### 5.8.1.4 make\_leds\_light\_up()

```
void make_leds_light_up (
    const std::vector< State_vector::Qubit_states > & states,
    Led & led0,
    Led & led1,
    Led & led2,
    Led & led3 )
```

## 5.9 /home/oliver/Documents/piquantum/src/final/pin\_mappings.hpp File Reference

This graph shows which files directly or indirectly include this file:

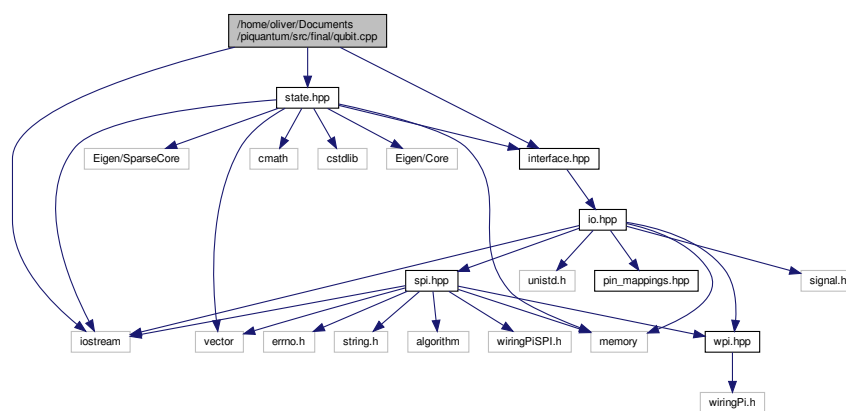


## Data Structures

- class [PIN](#)

## 5.10 /home/oliver/Documents/piquantum/src/final/qubit.cpp File Reference

```
#include <iostream>
#include "state.hpp"
#include "interface.hpp"
Include dependency graph for qubit.cpp:
```



## Functions

- int [main](#) (void)

### 5.10.1 Function Documentation

#### 5.10.1.1 main()

```
int main (
    void )
```

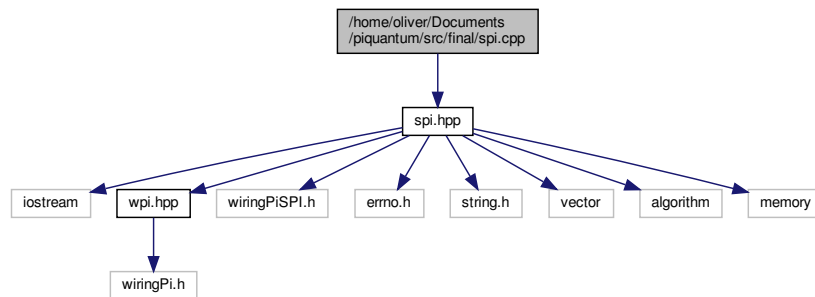
qubit leds RGB positions

vector of qubit btns

## 5.11 /home/oliver/Documents/piquantum/src/final/spi.cpp File Reference

```
#include "spi.hpp"
```

Include dependency graph for spi.cpp:



### Functions

- `std::shared_ptr< SpiChannel > getSpiChannel ()`  
*Wrapper to return SPI class.*

#### 5.11.1 Detailed Description

##### Authors

J Scott, O Thomas

##### Date

Feb 2019

Implementations relating to SPI communications

#### 5.11.2 Function Documentation

##### 5.11.2.1 getSpiChannel()

```
std::shared_ptr<SpiChannel> getSpiChannel ( )
```

Wrapper to return SPI class.

Return a `shared_ptr` object to an SPI channel. This function is a kind of singleton implementation, preventing multiple copies of the SPI class. The actual SPI objects are stored in static variables so that they retain their value between function calls.





## Functions

- `std::shared_ptr< class SpiChannel > getSpiChannel ()`  
*Wrapper to return SPI class.*

### 5.12.1 Detailed Description

#### Authors

J Scott, O Thomas

#### Date

Feb 2019

Header file for SPI (serial peripheral interface) control using wiringPi. The wiringPi library is available by cloning `git://git.drogon.net/wiringPi` and running `./build`. It has two functions: a setup function and a read/write function. The program needs sudo rights to run. Also the SPI needs to be enabled in `raspi-config`.

### 5.12.2 Typedef Documentation

#### 5.12.2.1 byte

```
typedef unsigned char byte
```

Byte (8 bits) type.

Alias for unsigned char. Type to store 8 bits of data for hardware reads and writes.

### 5.12.3 Function Documentation

#### 5.12.3.1 `getSpiChannel()`

```
std::shared_ptr<class SpiChannel> getSpiChannel ( )
```

Wrapper to return SPI class.

Return a `shared_ptr` object to an SPI channel. This function is a kind of singleton implementation, preventing multiple copies of the SPI class. The actual SPI object is stored in a static variables so that it retains its value between function calls.

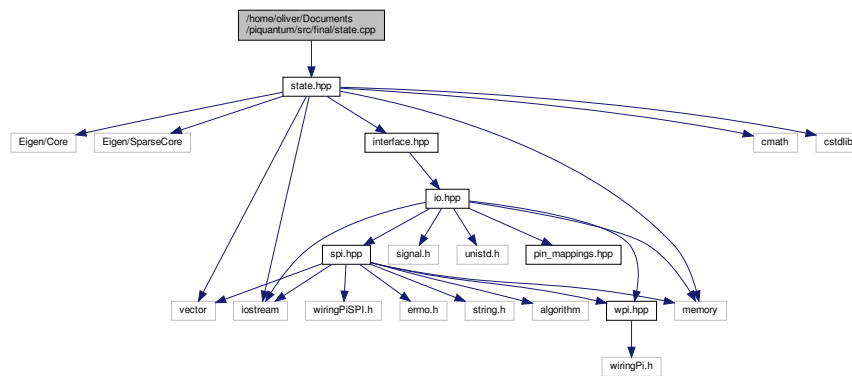
Return a `shared_ptr` object to an SPI channel. This function is a kind of singleton implementation, preventing multiple copies of the SPI class. The actual SPI objects are stored in static variables so that they retain their value between function calls.

## 5.13 /home/oliver/Documents/piquantum/src/final/state.cpp File Reference

State vector class.

```
#include "state.hpp"
```

Include dependency graph for state.cpp:



### Variables

- `std::vector< std::vector< Qubit::Qubit\_state > > qubit_disp_cycle`

#### 5.13.1 Detailed Description

State vector class.

#### Authors

J Scott, O Thomas

#### Date

Feb 2019

#### 5.13.2 Variable Documentation

##### 5.13.2.1 qubit\_disp\_cycle

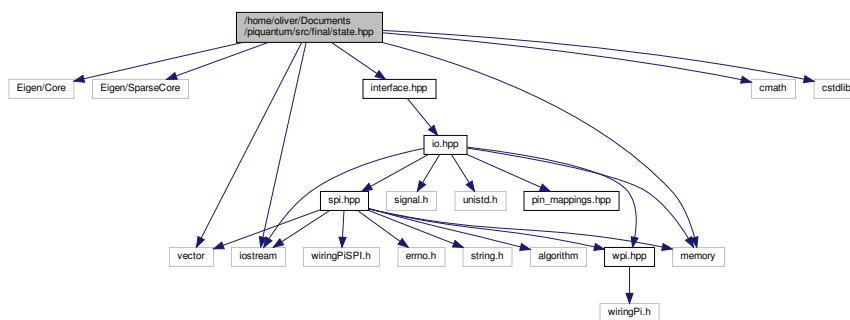
```
std::vector<std::vector<Qubit::Qubit\_state> > qubit_disp_cycle
```

## 5.14 /home/oliver/Documents/piquantum/src/final/state.hpp File Reference

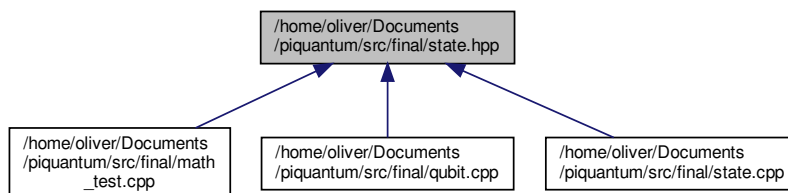
Header for state vector class and operators.

```
#include <Eigen/Core>
#include <Eigen/SparseCore>
#include <iostream>
#include <vector>
#include <memory>
#include <cmath>
#include <cstdlib>
#include "interface.hpp"
```

Include dependency graph for state.hpp:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [Operator](#)
- class [Rotation\\_X](#)
- class [Rotation\\_Y](#)
- class [Rotation\\_Z](#)
- class [Hadamard](#)
- class [Qubit](#)
- struct [Qubit::Qubit\\_state](#)
  - *define a struct to hold the qubit properties*
- class [State\\_vector](#)

## Typedefs

- typedef std::vector< std::shared\_ptr< [Qubit](#) > > [Qubits\\_type](#)

## Functions

- const std::complex< double > [I\\_unit](#) (0.0, 1.0)

## Variables

- const double [PI](#) =4.0\*atan(1.0)

### 5.14.1 Detailed Description

Header for state vector class and operators.

#### Authors

J Scott, O Thomas

#### Date

Feb 2019

### 5.14.2 Typedef Documentation

#### 5.14.2.1 Qubits\_type

```
typedef std::vector<std::shared_ptr<Qubit> > Qubits\_type
```

### 5.14.3 Function Documentation

#### 5.14.3.1 I\_unit()

```
const std::complex<double> I_unit (  
    0.  0,  
    1.  0 )
```

### 5.14.4 Variable Documentation

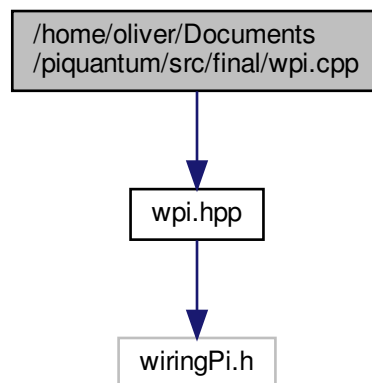
#### 5.14.4.1 PI

```
const double PI =4.0*atan(1.0)
```

## 5.15 /home/oliver/Documents/piquantum/src/final/wpi.cpp File Reference

```
#include "wpi.hpp"
```

Include dependency graph for wpi.cpp:



### 5.15.1 Detailed Description

#### Authors

J Scott, O Thomas

#### Date

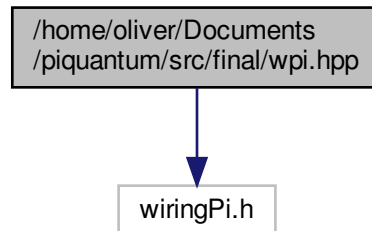
Feb 2019

Source file for [WiringPi](#) functions

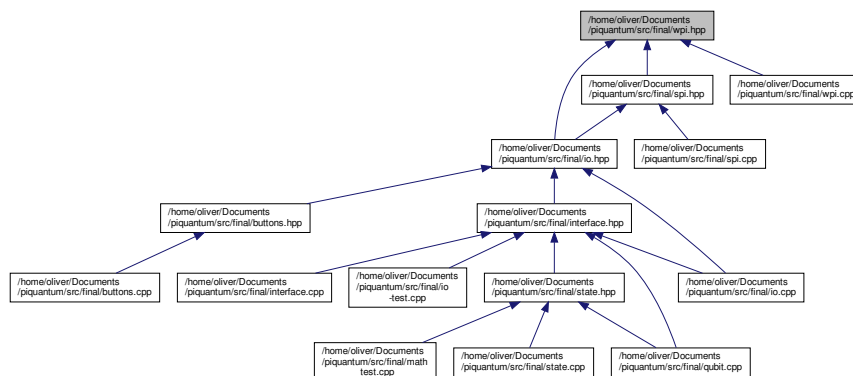
## 5.16 /home/oliver/Documents/piquantum/src/final/wpi.hpp File Reference

```
#include <wiringPi.h>
```

Include dependency graph for wpi.hpp:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [WiringPi](#)  
*Class for initialising wiringPi.*

#### 5.16.1 Detailed Description

##### Authors

J Scott, O Thomas

##### Date

Feb 2019

Header file for [WiringPi](#) functions





# Index

/home/oliver/Documents/piquantum/src/final/buttons.↵  
cpp, 35  
/home/oliver/Documents/piquantum/src/final/buttons.↵  
hpp, 36  
/home/oliver/Documents/piquantum/src/final/interface.↵  
cpp, 37  
/home/oliver/Documents/piquantum/src/final/interface.↵  
hpp, 37  
/home/oliver/Documents/piquantum/src/final/io-test.cpp,  
39  
/home/oliver/Documents/piquantum/src/final/io.cpp, 40  
/home/oliver/Documents/piquantum/src/final/io.hpp, 41  
/home/oliver/Documents/piquantum/src/final/math\_↵  
test.cpp, 43  
/home/oliver/Documents/piquantum/src/final/pin\_↵  
mappings.hpp, 44  
/home/oliver/Documents/piquantum/src/final/qubit.cpp,  
45  
/home/oliver/Documents/piquantum/src/final/spi.cpp, 46  
/home/oliver/Documents/piquantum/src/final/spi.hpp, 47  
/home/oliver/Documents/piquantum/src/final/state.cpp,  
49  
/home/oliver/Documents/piquantum/src/final/state.hpp,  
50  
/home/oliver/Documents/piquantum/src/final/wpi.cpp, 52  
/home/oliver/Documents/piquantum/src/final/wpi.hpp,  
53  
~Button  
Button, 8  
Alarm, 7  
Alarm, 8  
apply  
State\_vector, 31  
btn\_ptr  
Operator, 18  
Button, 8  
~Button, 8  
Button, 8, 9  
get\_position, 9  
get\_state, 9  
InputOutput, 10  
rgb, 9  
byte  
spi.hpp, 48  
change\_frequency  
SpiChannel, 29  
check\_uptodate  
Qubit, 21  
chip  
Position, 20  
delay  
math\_test.cpp, 43  
deregister\_button  
InputOutput, 13  
deregister\_led  
InputOutput, 13  
disp  
State\_vector, 32  
disp\_cycle  
State\_vector, 32  
display\_avg  
State\_vector, 32  
get\_num\_qubits  
Operator, 17  
State\_vector, 32  
get\_one\_amp  
Qubit, 21  
get\_phase  
Qubit, 21  
get\_position  
Button, 9  
get\_positions  
Led, 16  
get\_qubit  
State\_vector, 32  
get\_qubit\_btn  
math\_test.cpp, 44  
get\_rgb  
Led, 16  
get\_size  
State\_vector, 32  
get\_state  
Button, 9  
get\_zero\_amp  
Qubit, 22  
getInputOutput  
io.cpp, 40  
io.hpp, 42  
getSpiChannel  
spi.cpp, 46  
spi.hpp, 48  
Hadamard, 10  
Hadamard, 11  
I\_unit

- state.hpp, 51
- InputOutput, 12
  - Button, 10
  - deregister\_button, 13
  - deregister\_led, 13
  - InputOutput, 13
  - print, 13
  - read\_button\_states, 13
  - register\_button, 14
  - register\_led, 14
  - set\_leds, 14
- io-test.cpp
  - main, 39
- io.cpp
  - getInputOutput, 40
  - operator<<, 41
- io.hpp
  - getInputOutput, 42
  - operator<<, 42
- LE
  - PIN, 19
- Led, 15
  - get\_positions, 16
  - get\_rgb, 16
  - Led, 15
  - set\_rgb, 16
- line
  - Position, 20
- main
  - io-test.cpp, 39
  - math\_test.cpp, 44
  - qubit.cpp, 45
- make\_leds\_light\_up
  - math\_test.cpp, 44
- math\_test.cpp
  - delay, 43
  - get\_qubit\_btn, 44
  - main, 44
  - make\_leds\_light\_up, 44
- matrix
  - Operator, 18
- name
  - Operator, 18
- num\_qubits
  - Operator, 18
- OE
  - PIN, 19
- one\_amp
  - Qubit::Qubit\_state, 23
- Operator, 16
  - btn\_ptr, 18
  - get\_num\_qubits, 17
  - matrix, 18
  - name, 18
  - num\_qubits, 18
  - Operator, 17
  - print, 18
  - selected, 18
  - set\_btn, 18
- operator<<
  - io.cpp, 41
  - io.hpp, 42
- PIN, 19
  - LE, 19
  - OE, 19
  - SHLD, 19
- phase
  - Qubit::Qubit\_state, 24
- PI
  - state.hpp, 52
- Position, 20
  - chip, 20
  - line, 20
- print
  - InputOutput, 13
  - Operator, 18
  - State\_vector, 33
- Qubit, 20
  - check\_uptodate, 21
  - get\_one\_amp, 21
  - get\_phase, 21
  - get\_zero\_amp, 22
  - Qubit, 21
  - selected, 22
  - set\_amps, 22
  - set\_led, 22
  - set\_one, 22
  - set\_phase, 22
  - set\_uptodate, 23
  - set\_zero, 23
- qubit.cpp
  - main, 45
- Qubit::Qubit\_state, 23
  - one\_amp, 23
  - phase, 24
  - zero\_amp, 24
- qubit\_disp\_cycle
  - state.cpp, 49
- qubits
  - State\_vector, 33
- Qubits\_type
  - state.hpp, 51
- read
  - SpiChannel, 29
- read\_button\_states
  - InputOutput, 13
- register\_button
  - InputOutput, 14
- register\_led
  - InputOutput, 14
- rgb

- Button, [9](#)
- Rotation\_X, [24](#)
  - Rotation\_X, [25](#)
- Rotation\_Y, [26](#)
  - Rotation\_Y, [27](#)
- Rotation\_Z, [27](#)
  - Rotation\_Z, [28](#)
- SHLD
  - PIN, [19](#)
- selected
  - Operator, [18](#)
  - Qubit, [22](#)
- set\_amps
  - Qubit, [22](#)
- set\_btn
  - Operator, [18](#)
- set\_led
  - Qubit, [22](#)
- set\_leds
  - InputOutput, [14](#)
- set\_one
  - Qubit, [22](#)
- set\_phase
  - Qubit, [22](#)
- set\_rgb
  - Led, [16](#)
- set\_superpos
  - State\_vector, [33](#)
- set\_uptodate
  - Qubit, [23](#)
- set\_vacuum
  - State\_vector, [33](#)
- set\_zero
  - Qubit, [23](#)
- spi.cpp
  - getSpiChannel, [46](#)
- spi.hpp
  - byte, [48](#)
  - getSpiChannel, [48](#)
- SpiChannel, [29](#)
  - change\_frequency, [29](#)
  - read, [29](#)
  - SpiChannel, [29](#)
  - write, [30](#)
- state.cpp
  - qubit\_disp\_cycle, [49](#)
- state.hpp
  - I\_unit, [51](#)
  - PI, [52](#)
  - Qubits\_type, [51](#)
- State\_vector, [30](#)
  - apply, [31](#)
  - disp, [32](#)
  - disp\_cycle, [32](#)
  - display\_avg, [32](#)
  - get\_num\_qubits, [32](#)
  - get\_qubit, [32](#)
  - get\_size, [32](#)
  - print, [33](#)
  - qubits, [33](#)
  - set\_superpos, [33](#)
  - set\_vacuum, [33](#)
  - State\_vector, [31](#)
- WiringPi, [33](#)
  - WiringPi, [34](#)
- write
  - SpiChannel, [30](#)
- zero\_amp
  - Qubit::Qubit\_state, [24](#)