

**BFIT**

Generated by Doxygen 1.16.1



---

<b>1 BFIT</b>	<b>1</b>
1.1 Features . . . . .	1
1.2 Repository structure . . . . .	1
1.3 Documentation . . . . .	2
<b>2 Directory Hierarchy</b>	<b>3</b>
2.1 Directories . . . . .	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List . . . . .	5
<b>4 File Index</b>	<b>7</b>
4.1 File List . . . . .	7
<b>5 Directory Documentation</b>	<b>9</b>
5.1 include Directory Reference . . . . .	9
5.2 src Directory Reference . . . . .	10
<b>6 Class Documentation</b>	<b>11</b>
6.1 inventory Struct Reference . . . . .	11
6.1.1 Member Data Documentation . . . . .	11
6.1.1.1 number_of_products_stocked . . . . .	11
6.1.1.2 products_in_inventory . . . . .	11
6.1.1.3 room_number . . . . .	11
6.2 product Struct Reference . . . . .	12
6.2.1 Member Data Documentation . . . . .	12
6.2.1.1 beverage_variant . . . . .	12
6.2.1.2 name . . . . .	12
6.2.1.3 price . . . . .	12
6.2.1.4 weight . . . . .	12
6.3 products_stocked Struct Reference . . . . .	13
6.3.1 Member Data Documentation . . . . .	13
6.3.1.1 beverage . . . . .	13
6.3.1.2 current_quantity . . . . .	13
6.3.1.3 original_quantity . . . . .	13
6.4 User Struct Reference . . . . .	13
6.4.1 Member Data Documentation . . . . .	14
6.4.1.1 balance . . . . .	14
6.4.1.2 roomNumber . . . . .	14
6.4.1.3 uid . . . . .	14
<b>7 File Documentation</b>	<b>15</b>
7.1 include/admin_html.h File Reference . . . . .	15
7.1.1 Detailed Description . . . . .	15

---

7.1.2 Variable Documentation . . . . .	15
7.1.2.1 PROGMEM . . . . .	15
7.2 admin_html.h . . . . .	16
7.3 include/buzzer.h File Reference . . . . .	16
7.3.1 Detailed Description . . . . .	16
7.3.2 Macro Definition Documentation . . . . .	17
7.3.2.1 BUZZER_H . . . . .	17
7.3.3 Function Documentation . . . . .	17
7.3.3.1 play_lock() . . . . .	17
7.3.3.2 play_unlock() . . . . .	17
7.3.3.3 play_warning() . . . . .	17
7.4 buzzer.h . . . . .	17
7.5 include/fridge_state.h File Reference . . . . .	17
7.5.1 Detailed Description . . . . .	18
7.5.2 Variable Documentation . . . . .	18
7.5.2.1 fridge . . . . .	18
7.6 fridge_state.h . . . . .	18
7.7 include/graph_data.h File Reference . . . . .	18
7.7.1 Detailed Description . . . . .	19
7.7.2 Macro Definition Documentation . . . . .	19
7.7.2.1 ROOM_COUNT . . . . .	19
7.7.3 Function Documentation . . . . .	19
7.7.3.1 graph_add_to_room_clasic() . . . . .	19
7.7.3.2 graph_add_to_room_green() . . . . .	19
7.7.3.3 print_graph_arrays() . . . . .	20
7.7.4 Variable Documentation . . . . .	20
7.7.4.1 classicHeight . . . . .	20
7.7.4.2 greenHeight . . . . .	20
7.8 graph_data.h . . . . .	20
7.9 include/index_html.h File Reference . . . . .	20
7.9.1 Detailed Description . . . . .	21
7.9.2 Variable Documentation . . . . .	21
7.9.2.1 PROGMEM . . . . .	21
7.10 index_html.h . . . . .	21
7.11 include/init_users_and_sale.h File Reference . . . . .	22
7.11.1 Detailed Description . . . . .	22
7.11.2 Macro Definition Documentation . . . . .	22
7.11.2.1 number_of_users . . . . .	22
7.11.3 Function Documentation . . . . .	23
7.11.3.1 init_users_and_products() . . . . .	23
7.11.3.2 perform_sale() . . . . .	23
7.12 init_users_and_sale.h . . . . .	23

---

7.13 include/inventory.h File Reference . . . . .	23
7.13.1 Detailed Description . . . . .	24
7.13.2 Macro Definition Documentation . . . . .	24
7.13.2.1 INVENTORY_CAPACITY . . . . .	24
7.13.3 Enumeration Type Documentation . . . . .	25
7.13.3.1 beverage_type . . . . .	25
7.13.4 Function Documentation . . . . .	26
7.13.4.1 inventory_add_beverage() . . . . .	26
7.13.4.2 inventory_add_product() . . . . .	26
7.13.4.3 inventory_init() . . . . .	27
7.13.4.4 inventory_make_product() . . . . .	27
7.13.4.5 inventory_print() . . . . .	27
7.13.4.6 inventory_remove_beverage() . . . . .	27
7.13.4.7 inventory_remove_product() . . . . .	28
7.14 inventory.h . . . . .	28
7.15 include/lock_ctrl.h File Reference . . . . .	29
7.15.1 Detailed Description . . . . .	30
7.15.2 Macro Definition Documentation . . . . .	30
7.15.2.1 CLOSED_THRESHOLD . . . . .	30
7.15.2.2 LIGHT_PIN . . . . .	30
7.15.2.3 OPEN_THRESHOLD . . . . .	30
7.15.2.4 SERVO_PIN . . . . .	30
7.15.3 Function Documentation . . . . .	31
7.15.3.1 is_box_closed() . . . . .	31
7.15.3.2 lock_ctrl_init() . . . . .	31
7.15.3.3 lock_door() . . . . .	31
7.15.3.4 play_close() . . . . .	31
7.15.3.5 play_open() . . . . .	31
7.15.3.6 unlock_door() . . . . .	31
7.16 lock_ctrl.h . . . . .	32
7.17 include/login_html.h File Reference . . . . .	32
7.17.1 Detailed Description . . . . .	32
7.17.2 Variable Documentation . . . . .	32
7.17.2.1 PROGMEM . . . . .	32
7.18 login_html.h . . . . .	33
7.19 include/rfid_access.h File Reference . . . . .	33
7.19.1 Detailed Description . . . . .	35
7.19.2 Macro Definition Documentation . . . . .	35
7.19.2.1 MAX_ROOMS . . . . .	35
7.19.2.2 RST_PIN . . . . .	35
7.19.2.3 SS_PIN . . . . .	35
7.19.2.4 UID_LENGTH . . . . .	35

---

7.19.3 Enumeration Type Documentation . . . . .	35
7.19.3.1 RFIDcommand . . . . .	35
7.19.4 Function Documentation . . . . .	36
7.19.4.1 add_user() . . . . .	36
7.19.4.2 check_command() . . . . .	36
7.19.4.3 compare_UID() . . . . .	36
7.19.4.4 count_rooms() . . . . .	37
7.19.4.5 display_commands() . . . . .	37
7.19.4.6 display_commands_um() . . . . .	37
7.19.4.7 find_empty_index() . . . . .	37
7.19.4.8 get_users_db() . . . . .	37
7.19.4.9 print_all_users() . . . . .	38
7.19.4.10 print_single_user() . . . . .	38
7.19.4.11 print_uid() . . . . .	38
7.19.4.12 read_confirmation() . . . . .	38
7.19.4.13 read_integer() . . . . .	38
7.19.4.14 read_RFID_tag() . . . . .	39
7.19.4.15 remove_user() . . . . .	40
7.19.4.16 rfid_get_last_uid() . . . . .	40
7.19.4.17 rfid_set_last_uid() . . . . .	40
7.19.4.18 setup_RFID_reader() . . . . .	41
7.19.4.19 user_management() . . . . .	41
7.19.4.20 validate_rfid() . . . . .	41
7.19.5 Variable Documentation . . . . .	41
7.19.5.1 userCount . . . . .	41
7.19.5.2 users . . . . .	41
7.20 rfid_access.h . . . . .	42
7.21 include/sale_html.h File Reference . . . . .	43
7.21.1 Detailed Description . . . . .	44
7.21.2 Function Documentation . . . . .	44
7.21.2.1 send_sale_html_graph() . . . . .	44
7.21.2.2 send_sale_html_page() . . . . .	44
7.21.3 Variable Documentation . . . . .	44
7.21.3.1 PROGMEM . . . . .	44
7.22 sale_html.h . . . . .	45
7.23 include/style_css.h File Reference . . . . .	46
7.23.1 Detailed Description . . . . .	46
7.23.2 Variable Documentation . . . . .	46
7.23.2.1 PROGMEM . . . . .	46
7.24 style_css.h . . . . .	46
7.25 include/weight_scale.h File Reference . . . . .	47
7.25.1 Detailed Description . . . . .	48

---

7.25.2 Macro Definition Documentation . . . . .	48
7.25.2.1 BEER_WEIGHT . . . . .	48
7.25.2.2 HX711_DOUT . . . . .	48
7.25.2.3 HX711_SCK . . . . .	49
7.25.2.4 SCALE_DEFAULT_SETTLE_TIME_MS . . . . .	49
7.25.2.5 SCALE_TOL . . . . .	49
7.25.3 Function Documentation . . . . .	49
7.25.3.1 get_beer_cans_taken() . . . . .	49
7.25.3.2 get_weight() . . . . .	49
7.25.3.3 get_weight_reference() . . . . .	49
7.25.3.4 reset_weight_reference() . . . . .	50
7.25.3.5 set_weight_reference() . . . . .	50
7.25.3.6 setup_scale() . . . . .	50
7.25.3.7 tare_complete() . . . . .	50
7.25.3.8 tare_scale() . . . . .	50
7.25.3.9 update_scale() . . . . .	50
7.25.3.10 weight_reference_is_set() . . . . .	51
7.25.4 Variable Documentation . . . . .	51
7.25.4.1 scale . . . . .	51
7.26 weight_scale.h . . . . .	51
7.27 README.md File Reference . . . . .	52
7.28 src/buzzer.cpp File Reference . . . . .	52
7.28.1 Function Documentation . . . . .	52
7.28.1.1 play_lock() . . . . .	52
7.28.1.2 play_unlock() . . . . .	53
7.28.1.3 play_warning() . . . . .	53
7.28.2 Variable Documentation . . . . .	53
7.28.2.1 BUZZERPIN . . . . .	53
7.28.2.2 HIGH_TONE . . . . .	53
7.28.2.3 LOW_TONE . . . . .	53
7.28.2.4 TONE_LENGTH . . . . .	53
7.29 src/database_management.cpp File Reference . . . . .	53
7.29.1 Detailed Description . . . . .	54
7.29.2 Function Documentation . . . . .	54
7.29.2.1 count_rooms() . . . . .	54
7.29.2.2 find_empty_index() . . . . .	55
7.29.2.3 get_users_db() . . . . .	55
7.29.2.4 print_all_users() . . . . .	55
7.29.2.5 print_single_user() . . . . .	55
7.29.2.6 print_uid() . . . . .	56
7.29.2.7 read_confirmation() . . . . .	56
7.29.2.8 read_integer() . . . . .	56

7.29.2.9 remove_user()	56
7.29.2.10 user_management()	56
7.30 src/fridge_state.cpp File Reference	57
7.30.1 Detailed Description	57
7.30.2 Variable Documentation	57
7.30.2.1 fridge	57
7.31 src/graph_data.cpp File Reference	57
7.31.1 Detailed Description	58
7.31.2 Function Documentation	58
7.31.2.1 graph_add_to_room_clasic()	58
7.31.2.2 graph_add_to_room_green()	58
7.31.3 Variable Documentation	58
7.31.3.1 classicHeight	58
7.31.3.2 greenHeight	59
7.32 src/init_users_and_sale.cpp File Reference	59
7.32.1 Function Documentation	59
7.32.1.1 init_users_and_products()	59
7.32.1.2 perform_sale()	59
7.32.1.3 read_current_weight_blocking()	60
7.33 src/inventory.cpp File Reference	60
7.33.1 Detailed Description	60
7.33.2 Function Documentation	60
7.33.2.1 inventory_add_beverage()	60
7.33.2.2 inventory_add_product()	60
7.33.2.3 inventory_init()	61
7.33.2.4 inventory_make_product()	61
7.33.2.5 inventory_print()	61
7.33.2.6 inventory_remove_beverage()	62
7.33.2.7 inventory_remove_product()	62
7.34 src/lock_ctrl.cpp File Reference	62
7.34.1 Detailed Description	63
7.34.2 Function Documentation	63
7.34.2.1 is_box_closed()	63
7.34.2.2 lock_ctrl_init()	64
7.34.2.3 lock_door()	64
7.34.2.4 play_close()	64
7.34.2.5 play_open()	64
7.34.2.6 unlock_door()	64
7.34.3 Variable Documentation	64
7.34.3.1 boxClosed	64
7.34.3.2 BUZZER	64
7.34.3.3 HIGH_TONE	64

---

7.34.3.4 LOCK_POS . . . . .	64
7.34.3.5 lockServo . . . . .	65
7.34.3.6 LOW_TONE . . . . .	65
7.34.3.7 TONE_LENGTH . . . . .	65
7.34.3.8 UNLOCK_POS . . . . .	65
7.35 src/main.cpp File Reference . . . . .	65
7.35.1 Detailed Description . . . . .	66
7.35.2 Function Documentation . . . . .	66
7.35.2.1 connect_wifi_and_start_mdns() . . . . .	66
7.35.2.2 loop() . . . . .	66
7.35.2.3 rfid() . . . . .	66
7.35.2.4 server() . . . . .	66
7.35.2.5 setup() . . . . .	67
7.35.2.6 setup_inventory_and_scale() . . . . .	67
7.35.2.7 setup_rfid_and_lock() . . . . .	67
7.35.2.8 setup_web_routes() . . . . .	67
7.35.3 Variable Documentation . . . . .	67
7.35.3.1 activeCommand . . . . .	67
7.35.3.2 CAL_FACTOR . . . . .	67
7.35.3.3 classicHeight . . . . .	67
7.35.3.4 demo_beer . . . . .	67
7.35.3.5 doorCloseTimer . . . . .	68
7.35.3.6 doorUnlocked . . . . .	68
7.35.3.7 greenHeight . . . . .	68
7.35.3.8 START_BEER_QTY . . . . .	68
7.35.3.9 WIFI_PASS . . . . .	68
7.35.3.10 WIFI_SSID . . . . .	68
7.36 src/rfid_access.cpp File Reference . . . . .	68
7.36.1 Detailed Description . . . . .	69
7.36.2 Function Documentation . . . . .	69
7.36.2.1 add_user() . . . . .	69
7.36.2.2 check_command() . . . . .	70
7.36.2.3 compare_UID() . . . . .	70
7.36.2.4 display_commands() . . . . .	70
7.36.2.5 display_commands_um() . . . . .	70
7.36.2.6 read_RFID_tag() . . . . .	70
7.36.2.7 rfid_get_last_uid() . . . . .	71
7.36.2.8 rfid_set_last_uid() . . . . .	71
7.36.2.9 setup_RFID_reader() . . . . .	71
7.36.2.10 validate_rfid() . . . . .	71
7.36.3 Variable Documentation . . . . .	72
7.36.3.1 hasUID . . . . .	72

7.36.3.2 lastUID . . . . .	72
7.36.3.3 userCount . . . . .	72
7.36.3.4 users . . . . .	72
7.37 src/sale_html.cpp File Reference . . . . .	72
7.37.1 Detailed Description . . . . .	72
7.37.2 Function Documentation . . . . .	73
7.37.2.1 send_sale_html_graph() . . . . .	73
7.37.2.2 send_sale_html_page() . . . . .	74
7.37.3 Variable Documentation . . . . .	74
7.37.3.1 PROGMEM . . . . .	74
7.38 src/weight_scale.cpp File Reference . . . . .	75
7.38.1 Detailed Description . . . . .	75
7.38.2 Function Documentation . . . . .	75
7.38.2.1 get_beer_cans_taken() . . . . .	75
7.38.2.2 get_weight() . . . . .	76
7.38.2.3 get_weight_reference() . . . . .	76
7.38.2.4 reset_weight_reference() . . . . .	76
7.38.2.5 scale() . . . . .	76
7.38.2.6 set_weight_reference() . . . . .	76
7.38.2.7 setup_scale() . . . . .	77
7.38.2.8 tare_complete() . . . . .	77
7.38.2.9 tare_scale() . . . . .	77
7.38.2.10 update_scale() . . . . .	77
7.38.2.11 weight_reference_is_set() . . . . .	77
7.38.3 Variable Documentation . . . . .	77
7.38.3.1 g_referenceWeight . . . . .	77
Index . . . . .	79

# Chapter 1

## BFIT

Beer Fridge Inventory Tracking (BFIT) is a system that logs who takes drinks from a fridge, how much they consume, and provides statistics on current stock and usage. The project was developed as part of the DTU course **34338 – Telecommunication Programming Project with Arduino**.

This repository contains the embedded code, server/backend logic, documentation, and supporting files for the project.

### 1.1 Features

- RFID-based user authentication
- Automatic detection of items removed using scale
- Servo-controlled locking mechanism
- Local web server displaying usage and inventory data
- [User](#) and inventory database

### 1.2 Repository structure

Folder / File	Description
src/↔	Source code
include/↔	Header files
docs/↔	Doxygen generated documentation
figs/↔	Schematics
parts/↔	3D drawings

Folder / File	Description
platformio.ini	PlatformIO configuration
Doxyfile	Doxygen documentation config
LICENSE	GPL-2.0 license

## 1.3 Documentation

The Doxygen generated document is `final_doxxygen_doc.pdf`.

# Chapter 2

## Directory Hierarchy

### 2.1 Directories

include . . . . .	9
admin_html.h . . . . .	15
buzzer.h . . . . .	16
fridge_state.h . . . . .	17
graph_data.h . . . . .	18
index_html.h . . . . .	20
init_users_and_sale.h . . . . .	22
inventory.h . . . . .	23
lock_ctrl.h . . . . .	29
login_html.h . . . . .	32
rfid_access.h . . . . .	33
sale_html.h . . . . .	43
style_css.h . . . . .	46
weight_scale.h . . . . .	47
src . . . . .	10
buzzer.cpp . . . . .	52
database_management.cpp . . . . .	53
fridge_state.cpp . . . . .	57
graph_data.cpp . . . . .	57
init_users_and_sale.cpp . . . . .	59
inventory.cpp . . . . .	60
lock_ctrl.cpp . . . . .	62
main.cpp . . . . .	65
rfid_access.cpp . . . . .	68
sale_html.cpp . . . . .	72
weight_scale.cpp . . . . .	75



# Chapter 3

## Class Index

### 3.1 Class List

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

<a href="#">inventory</a>	Represents a user's beverage inventory . . . . .	11
<a href="#">product</a>	Describes a single beverage product . . . . .	12
<a href="#">products_stocked</a>	Tracks inventory quantities for a single product . . . . .	13
<a href="#">User</a>	User record stored in the RFID user database . . . . .	13



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

include/admin_html.h	HTML for the admin page . . . . .	15
include/buzzer.h	Declaring functions to play sounds using a buzzer . . . . .	16
include/fridge_state.h	File for declearing the fridge inventory, user inventory shuld also be moved to this file, and the file renamed to reflekt the new content . . . . .	17
include/graph_data.h	File used for setting up the graph section of the web server . . . . .	18
include/index_html.h	HTML for the start page . . . . .	20
include/init_users_and_sale.h	Function used for declearing the system users and products . . . . .	22
include/inventory.h	Inventory system for tracking the inventory of both the fridge and the induvidual users . . . . .	23
include/lock_ctrl.h	Door lock control interface using a servo and light sensor . . . . .	29
include/login_html.h	HTML file for the login page . . . . .	32
include/rfid_access.h	Declare functions related to RFID reading and user database . . . . .	33
include/sale_html.h	Headerfile for displaying the graph of sales on the main page . . . . .	43
include/style_css.h	CSS served at /style.css used for setting the style for the web server . . . . .	46
include/weight_scale.h	. . . . .	47
src/buzzer.cpp	. . . . .	52
src/database_management.cpp	Functions for user management and read/write of non-volatile memory on ESP8266 . . . . .	53
src/fridge_state.cpp	File used for defining the fridge inventory, definition of user inventory shuld also be placed here . . . . .	57
src/graph_data.cpp	File used to set up the graph section in the web server . . . . .	57
src/init_users_and_sale.cpp	. . . . .	59
src/inventory.cpp	Functions responsible for keeping track of the fridge inventory . . . . .	60

src/lock_ctrl.cpp	Implements servo-based locking control and buzzer feedback for the fridge door . . . . .	62
src/main.cpp	Combined: Web server + Graph + Scale + RFID access + Lock control . . . . .	65
src/rfid_access.cpp	File used to setup the RFID . . . . .	68
src/sale_html.cpp	File for setting up the graph section of the web server . . . . .	72
src/weight_scale.cpp	File used for setting up the scale . . . . .	75

# Chapter 5

## Directory Documentation

### 5.1 include Directory Reference

#### Files

- file [admin\\_html.h](#)  
*HTML for the admin page.*
- file [buzzer.h](#)  
*Declaring functions to play sounds using a buzzer.*
- file [fridge\\_state.h](#)  
*File for declearing the fridge inventory, user inventory shuld also be moved to this file, and the file renamed to reflekt the new content.*
- file [graph\\_data.h](#)  
*File used for setting up the graph section of the web server.*
- file [index\\_html.h](#)  
*HTML for the start page.*
- file [init\\_users\\_and\\_sale.h](#)  
*Function used for declearing the system users and products.*
- file [inventory.h](#)  
*Inventory system for tracking the inventory of both the fridge and the individual users.*
- file [lock\\_ctrl.h](#)  
*Door lock control interface using a servo and light sensor.*
- file [login\\_html.h](#)  
*HTML file for the login page.*
- file [rfid\\_access.h](#)  
*Declare functions related to RFID reading and user database.*
- file [sale\\_html.h](#)  
*Headerfile for displaying the graph of sales on the main page.*
- file [style\\_css.h](#)  
*CSS served at /style.css used for setting the style for the web server.*
- file [weight\\_scale.h](#)

## 5.2 src Directory Reference

### Files

- file [buzzer.cpp](#)
- file [database\\_management.cpp](#)

*Functions for user management and read/write of non-volatile memory on ESP8266.*

- file [fridge\\_state.cpp](#)

*File used for defining the fridge inventory, definition of user inventory shuld also be placed here.*

- file [graph\\_data.cpp](#)

*File used to set up the graph section in the web server.*

- file [init\\_users\\_and\\_sale.cpp](#)

- file [inventory.cpp](#)

*Functions responsible for keeping track of the fridge inventory.*

- file [lock\\_ctrl.cpp](#)

*Implements servo-based locking control and buzzer feedback for the fridge door.*

- file [main.cpp](#)

*Combined: Web server + Graph + Scale + RFID access + Lock control.*

- file [rfid\\_access.cpp](#)

*File used to setup the RFID.*

- file [sale\\_html.cpp](#)

*File for setting up the graph section of the web server.*

- file [weight\\_scale.cpp](#)

*File used for setting up the scale.*

# Chapter 6

## Class Documentation

### 6.1 inventory Struct Reference

Represents a user's beverage inventory.

```
#include <inventory.h>
```

#### Public Attributes

- `products_stocked` `products_in_inventory` [INVENTORY\_CAPACITY]  
*Products currently in inventory.*
- `uint8_t number_of_products_stocked`  
*Number of valid entries in products\_in\_inventory.*
- `uint8_t room_number`  
*Room number associated with the inventory (currently unused).*

#### 6.1.1 Member Data Documentation

##### 6.1.1.1 `number_of_products_stocked`

```
uint8_t inventory::number_of_products_stocked
```

##### 6.1.1.2 `products_in_inventory`

```
products_stocked inventory::products_in_inventory[INVENTORY_CAPACITY]
```

##### 6.1.1.3 `room_number`

```
uint8_t inventory::room_number
```

FRID of the user owning the inventory (not implemented)

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

- `include/inventory.h`

## 6.2 product Struct Reference

Describes a single beverage product.

```
#include <inventory.h>
```

### Public Attributes

- char `name` [20]  
*Display name of the beverage.*
- `beverage_type beverage_variant`  
*Beverage type.*
- `uint16_t weight`  
*Weight of the beverage (e.g.*
- `uint8_t price`  
*Price of the beverage.*

### 6.2.1 Member Data Documentation

#### 6.2.1.1 `beverage_variant`

```
beverage_type product::beverage_variant
```

#### 6.2.1.2 `name`

```
char product::name[20]
```

#### 6.2.1.3 `price`

```
uint8_t product::price
```

#### 6.2.1.4 `weight`

```
uint16_t product::weight
```

grams)

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

- `include/inventory.h`

## 6.3 products\_stocked Struct Reference

Tracks inventory quantities for a single product.

```
#include <inventory.h>
```

### Public Attributes

- `product beverage`  
*Product being tracked.*
- `uint8_t original_quantity`  
*Initial stock quantity.*
- `uint8_t current_quantity`  
*Current stock quantity.*

### 6.3.1 Member Data Documentation

#### 6.3.1.1 `beverage`

```
product products_stocked::beverage
```

#### 6.3.1.2 `current_quantity`

```
uint8_t products_stocked::current_quantity
```

#### 6.3.1.3 `original_quantity`

```
uint8_t products_stocked::original_quantity
```

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

- `include/inventory.h`

## 6.4 User Struct Reference

`User` record stored in the RFID user database.

```
#include <rfid_access.h>
```

### Public Attributes

- byte `uid [UID_LENGTH]`
- int `roomNumber`
- int `balance`

## 6.4.1 Member Data Documentation

### 6.4.1.1 balance

```
int User::balance
```

### 6.4.1.2 roomNumber

```
int User::roomNumber
```

### 6.4.1.3 uid

```
byte User::uid[UID\_LENGTH]
```

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

- [include/rfid\\_access.h](#)

# Chapter 7

## File Documentation

### 7.1 `include/admin_html.h` File Reference

HTML for the admin page.

```
#include <pgmspace.h>
```

#### Variables

- `const char ADMIN_HTML[] PROGMEM`  
*HTML content for the admin web page.*

#### 7.1.1 Detailed Description

##### Authors

Baldur G. Toftegaard

#### 7.1.2 Variable Documentation

##### 7.1.2.1 PROGMEM

```
const char ADMIN_HTML [ ] PROGMEM
```

###### Initial value:

```
= R"rawliteral(
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Beer fridge online services</title>
    <link rel="stylesheet" href="/style.css">
  </head>
  <body>
    <div class="topbar">
      <div class="left">
        Start -> Admin
      </div>
      <div class="right">
        <a href="/">Log Out</a>
      </div>
    </div>
    <p> Verry important stuff goes here! </p>
  </body>
</html>
)rawliteral"
```

Opening container for the sales graph.

Interperated by compiler as a string.

## 7.2 admin\_html.h

[Go to the documentation of this file.](#)

```

00001
00006
00007 #ifndef ADMIN_HTML_H
00008 #define ADMIN_HTML_H
00009
00010 #include <pgmspace.h>
00011
00015 const char ADMIN_HTML[] PROGMEM = R"rawliteral(
00016     <!DOCTYPE html>
00017     <html>
00018         <head>
00019             <meta charset="utf-8">
00020             <title>Beer fridge online services</title>
00021             <link rel="stylesheet" href="/style.css">
00022         </head>
00023         <body>
00024             <div class="topbar">
00025                 <div class="left">
00026                     Start -> Admin
00027                 </div>
00028                 <div class="right">
00029                     <a href="/">Log Out</a>
00030                 </div>
00031             </div>
00032             <p> Verry important stuff goes here! </p>
00033         </body>
00034     </html>
00035 )rawliteral";
00036
00037 #endif

```

## 7.3 include/buzzer.h File Reference

Declaring functions to play sounds using a buzzer.

```
#include <Arduino.h>
```

### Macros

- `#define BUZZER_H`

### Functions

- void `play_warning` (unsigned long t)  
*Plays short sound when door is closed and about to be locked.*
- void `play_unlock` ()  
*Plays an ascending tone, when the door unlocks.*
- void `play_lock` ()  
*Play a descending tone, when the door locks.*

### 7.3.1 Detailed Description

Implementation of buzzer sound effects.

#### Author

Anssi Sohlman

### 7.3.2 Macro Definition Documentation

#### 7.3.2.1 BUZZER\_H

```
#define BUZZER_H
```

### 7.3.3 Function Documentation

#### 7.3.3.1 play\_lock()

```
void play_lock ()
```

#### 7.3.3.2 play\_unlock()

```
void play_unlock ()
```

#### 7.3.3.3 play\_warning()

```
void play_warning (
    unsigned long t)
```

##### Parameters

<i>t</i>	Time passed since door has been closed
----------	--

## 7.4 buzzer.h

[Go to the documentation of this file.](#)

```
00001
00006
00007 #include <Arduino.h> //Tone functions don't work without this here
00008
00009 #ifndef BUZZER_H
00010 #define BUZZER_H
00011
00017 void play_warning(unsigned long t);
00018
00022 void play_unlock();
00023
00027 void play_lock();
00028
00029 #endif
```

## 7.5 include/fridge\_state.h File Reference

File for declearing the fridge inventory, user inventory shuld also be moved to this file, and the file renamed to reflekt the new content.

```
#include "inventory.h"
```

## Variables

- **inventory\_fridge**

*Used to set up the fridge inventory.*

### 7.5.1 Detailed Description

#### Author

Baldur G. Toftegaard

### 7.5.2 Variable Documentation

#### 7.5.2.1 **fridge**

```
inventory_fridge [extern]
```

## 7.6 **fridge\_state.h**

[Go to the documentation of this file.](#)

```
00001
00002
00003 #ifndef FRIDGE_STATE_H
00004 #define FRIDGE_STATE_H
00005
00006 #include "inventory.h"
00007
00008 extern inventory_fridge;
00009
00010 #endif
```

## 7.7 **include/graph\_data.h** File Reference

File used for setting up the graph section of the web server.

```
#include <stdint.h>
```

## Macros

- **#define ROOM\_COUNT 18**

*Number of rooms supported by the sales graph.*

## Functions

- **void graph\_add\_to\_room\_green (uint8\_t roomNumber, int delta)**  
*Adds a value to the green sales bar of a given room.*
- **void graph\_add\_to\_room\_clasic (uint8\_t roomNumber, int delta)**  
*Adds a value to the classic sales bar of a given room.*
- **void print\_graph\_arrays ()**  
*Prints the current graph height arrays.*

## Variables

- int `greenHeight [ROOM_COUNT]`  
*Height values for green product sales per room.*
- int `classicHeight [ROOM_COUNT]`  
*Height values for classic product sales per room.*

## 7.7.1 Detailed Description

### Author

Baldur G. Toftegaard

## 7.7.2 Macro Definition Documentation

### 7.7.2.1 ROOM\_COUNT

```
#define ROOM_COUNT 18
```

## 7.7.3 Function Documentation

### 7.7.3.1 graph\_add\_to\_room\_clasic()

```
void graph_add_to_room_clasic (
    uint8_t roomNumber,
    int delta)
```

#### Parameters

<code>roomNumber</code>	Room operated on.
<code>delta</code>	Incremental value used to update the bar height.

### 7.7.3.2 graph\_add\_to\_room\_green()

```
void graph_add_to_room_green (
    uint8_t roomNumber,
    int delta)
```

#### Parameters

<code>roomNumber</code>	Room operated on.
<code>delta</code>	Incremental value used to update the bar height.

### 7.7.3.3 print\_graph\_arrays()

```
void print_graph_arrays ()
```

## 7.7.4 Variable Documentation

### 7.7.4.1 classicHeight

```
int classicHeight[ROOM_COUNT] [extern]
```

Height values for classic product sales per room.

Indexed by room number. Used by sale\_html and the /saleHeights endpoint.

### 7.7.4.2 greenHeight

```
int greenHeight[ROOM_COUNT] [extern]
```

Height values for green product sales per room.

Indexed by room number. Used by sale\_html and the /saleHeights endpoint.

## 7.8 graph\_data.h

[Go to the documentation of this file.](#)

```
00001
00006
00007 #ifndef GRAPH_DATA_H
00008 #define GRAPH_DATA_H
00009
00010 #include <stdint.h>
00011
00016 #define ROOM_COUNT 18
00017
00021 extern int greenHeight[ROOM_COUNT];
00022
00026 extern int classicHeight[ROOM_COUNT];
00027
00034 void graph_add_to_room_green(
00035     uint8_t roomNumber,
00036     int delta
00037 );
00038
00045 void graph_add_to_room_clasic(
00046     uint8_t roomNumber,
00047     int delta
00048 );
00049
00053 void print_graph_arrays();
00054
00055 #endif
```

## 7.9 include/index\_html.h File Reference

HTML for the start page.

```
#include <pgmspace.h>
```

## Variables

- const char INDEX\_HTML\_HEAD[] PROGMEM  
*String used for the HTML header, read by the compiler as a string.*

### 7.9.1 Detailed Description

#### Author

Baldur G. Toftegaard

### 7.9.2 Variable Documentation

#### 7.9.2.1 PROGMEM

```
const char INDEX_HTML_FOOT [ ] PROGMEM
```

##### Initial value:

```
= R"rawliteral(
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Beer fridge online services</title>
    <link rel="stylesheet" href="/style.css">
  </head>
  <body>
    <div class="topbar">
      <div class="left">Welcome!</div>
      <div class="right"><a href="/login">Login</a></div>
    </div>
)rawliteral"
```

Opening container for the sales graph.

String used for the HTML footer.

This is responsible for updating the graphs. read by the compiler as a string

## 7.10 index\_html.h

[Go to the documentation of this file.](#)

```
00001
00006
00007 #ifndef INDEX_HTML_H
00008 #define INDEX_HTML_H
00009
00010 #include <pgmspace.h>
00011
00015 const char INDEX_HTML_HEAD[] PROGMEM = R"rawliteral(
00016 <!DOCTYPE html>
00017 <html>
00018   <head>
00019     <meta charset="utf-8">
00020     <title>Beer fridge online services</title>
00021     <link rel="stylesheet" href="/style.css">
00022   </head>
00023   <body>
00024     <div class="topbar">
00025       <div class="left">Welcome!</div>
00026       <div class="right"><a href="/login">Login</a></div>
00027     </div>
```

```

00028 )rawliteral";
00029
00030 const char INDEX_HTML_FOOT[] PROGMEM = R"rawliteral(
00031     <script>
00032         async function refreshGraphs() {
00033             try {
00034                 /* Send a HTML GET request (no cashe to prevent old data from displaying) */
00035                 const res = await fetch('/saleHeights', { cache: 'no-store' });
00036                 /* Convert the HTML respons into a JavaScript object */
00037                 const data = await res.json();
00038
00039                 /* Loop through all rooms key in JSON object*/
00040                 for (const room in data) {
00041                     const green = document.getElementById(room + "_green");
00042                     const clasic = document.getElementById(room + "_clasic");
00043
00044                     if (green) green.style.height = data[room].green + "px";
00045                     if (clasic) clasic.style.height = data[room].clasic + "px";
00046
00047                 }
00048             } catch (e) {
00049                 console.error(e);
00050             }
00051         }
00052         refreshGraphs();
00053         setInterval(refreshGraphs, 200);
00054     </script>
00055     </body>
00056 </html>
00057 )rawliteral";
00058
00059
00060
00061
00062 #endif

```

## 7.11 include/init\_users\_and\_sale.h File Reference

Function used for declearing the system users and products.

```
#include "inventory.h"
```

### Macros

- `#define number_of_users 18`

### Functions

- `void init_users_and_products ()`  
*Old function for initializing users and products.*
- `void perform_sale (inventory *fridge_inventory)`  
*Performs a sale between a user and the fridge.*

#### 7.11.1 Detailed Description

##### Author

Baldur G. Toftegaard

#### 7.11.2 Macro Definition Documentation

##### 7.11.2.1 number\_of\_users

```
#define number_of_users 18
```

### 7.11.3 Function Documentation

#### 7.11.3.1 init\_users\_and\_products()

```
void init_users_and_products ()
```

Old function for initializing users and products.

#### 7.11.3.2 perform\_sale()

```
void perform_sale (
    inventory * fridge_inventory)
```

##### Parameters

<i>weight</i>	- Read the value from the weight.
<i>user_id</i>	- The user the sale shuld be registered to.
<i>fridge_inventory</i>	- The inventory we wish to remove the beverage from.

##### Parameters

<i>fridge_inventory</i>	Inventory that the sale shuld remove item from
-------------------------	--

## 7.12 init\_users\_and\_sale.h

[Go to the documentation of this file.](#)

```
00001
00006
00007 #ifndef INIT_USERS_AND_SALE_H
00008 #define INIT_USERS_AND_SALE_H
00009
00010 #include "inventory.h"
00011 #define number_of_users 18
00012
00016 void init_users_and_products();
00024 void perform_sale(
00025     inventory *fridge_inventory
00026 );
00027
00028 #endif
```

## 7.13 include/inventory.h File Reference

Inventory system for tracking the inventory of both the fridge and the individual users.

```
#include <stdint.h>
#include <stdbool.h>
#include <Arduino.h>
```

## Classes

- struct `product`  
*Describes a single beverage product.*
- struct `products_stocked`  
*Tracks inventory quantities for a single product.*
- struct `inventory`  
*Represents a user's beverage inventory.*

## Macros

- `#define INVENTORY_CAPACITY 6`  
*Maximum number of products the inventory can hold.*

## Enumerations

- enum `beverage_type` {  
  `beer` , `cider` , `soda` , `limfjords_porter` ,  
  `other` }  
*Types of beverages supported by the system.*

## Functions

- void `inventory_init (inventory *inventory)`  
*Initializes an inventory structure.*
- `product inventory_make_product (const char *name, beverage_type type, uint16_t weight, uint8_t price)`  
*Creates a new product instance.*
- bool `inventory_add_product (inventory *inventory, product product, uint16_t quantity)`  
*Adds a new product to the inventory.*
- bool `inventory_remove_product (inventory *inventory, product beverage)`  
*Removes a product entirely from the inventory.*
- bool `inventory_add_beverage (inventory *inventory, product beverage, uint16_t amount)`  
*Increases the quantity of a beverage in the inventory.*
- bool `inventory_remove_beverage (inventory *inventory, product beverage, uint8_t amount)`  
*Function for removing from the amount of a beverage in an inventory.*
- void `inventory_print (inventory *inventory)`  
*Function to print a users inventory.*

### 7.13.1 Detailed Description

#### Author

Baldur G. Toftegaard

### 7.13.2 Macro Definition Documentation

#### 7.13.2.1 INVENTORY\_CAPACITY

```
#define INVENTORY_CAPACITY 6
```

### 7.13.3 Enumeration Type Documentation

#### 7.13.3.1 beverage\_type

```
enum beverage_type
```

**Enumerator**

beer	Beer.
------	-------

cider	Cider.
soda	Soda.
limfjords_porter	Limfjords porter.
other	Other.

## 7.13.4 Function Documentation

### 7.13.4.1 inventory\_add\_beverage()

```
bool inventory_add_beverage (
    inventory * inventory,
    product beverage,
    uint16_t amount)
```

#### Parameters

<i>inventory</i>	Inventory to modify.
<i>beverage</i>	Beverage to update.
<i>amount</i>	Quantity to add.

#### Returns

true - Quantity updated successfully.  
false - Product not found or overflow.

### 7.13.4.2 inventory\_add\_product()

```
bool inventory_add_product (
    inventory * inventory,
    product product,
    uint16_t quantity)
```

#### Parameters

<i>inventory</i>	Inventory to add the product to.
<i>product</i>	Product to add.
<i>quantity</i>	Initial quantity of the product.

#### Returns

true - Product successfully added.  
false - Error occurred (inventory full or duplicate product).

#### 7.13.4.3 inventory\_init()

```
void inventory_init (
    inventory * inventory)
```

Clears internal state and prepares the inventory for use.

##### Parameters

<i>inventory</i>	Pointer to inventory instance to initialize.
------------------	--

#### 7.13.4.4 inventory\_make\_product()

```
product inventory_make_product (
    const char * name,
    beverage_type type,
    uint16_t weight,
    uint8_t price)
```

##### Parameters

<i>name</i>	Display name of the product.
<i>type</i>	Beverage type.
<i>weight</i>	Weight of the product.
<i>price</i>	Price of the product.

##### Returns

Initialized product instance.

#### 7.13.4.5 inventory\_print()

```
void inventory_print (
    inventory * inventory)
```

##### Parameters

<i>inventory</i>	Inventory to print.
------------------	---------------------

#### 7.13.4.6 inventory\_remove\_beverage()

```
bool inventory_remove_beverage (
    inventory * inventory,
    product beverage,
    uint8_t amount)
```

##### Parameters

<i>inventory</i>	Inventory to modify.
------------------	----------------------

<i>beverage</i>	Beverage to update.
<i>amount</i>	Quantity to remove.

**Returns**

true - Quantity updated successfully.  
 false - Product not found or insufficient stock.

**7.13.4.7 inventory\_remove\_product()**

```
bool inventory_remove_product (
    inventory * inventory,
    product beverage)
```

**Parameters**

<i>inventory</i>	Inventory to remove the product from.
<i>beverage</i>	Product to remove.

**Returns**

true - Product removed successfully.  
 false - Product not found.

**7.14 inventory.h**

[Go to the documentation of this file.](#)

```
00001
00006
00007 #ifndef INVENTORY_H
00008 #define INVENTORY_H
00009
00010 #include <stdint.h>
00011 #include <stdbool.h>
00012 #include <Arduino.h>
00013
00017 #define INVENTORY_CAPACITY 6
00018
00023 typedef enum {
00024     beer,
00025     cider,
00026     soda,
00027     limfjords_porter,
00028     other
00029 } beverage_type;
00030
00031
00036 typedef struct {
00037     char name[20];
00038     beverage_type beverage_variant;
00039     uint16_t weight;
00040     uint8_t price;
00041 } product;
00042
00043
00048 typedef struct {
00049     product beverage;
00050     uint8_t original_quantity;
00051     uint8_t current_quantity;
```

```

00052 } products_stocked;
00053
00054
00055 typedef struct {
00056     products_stocked products_in_inventory[INVENTORY_CAPACITY];
00057     uint8_t number_of_products_stocked;
00058     uint8_t room_number;
00059 } inventory;
00060
00061
00062
00063
00064
00065
00066
00067
00068 void inventory_init(
00069     inventory *inventory
00070 );
00071
00072
00073 product inventory_make_product(
00074     const char *name,
00075     beverage_type type,
00076     uint16_t weight,
00077     uint8_t price
00078 );
00079
00080
00081 bool inventory_add_product(
00082     inventory *inventory,
00083     product product,
00084     uint16_t quantity
00085 );
00086
00087
00088
00089
00090
00091
00092
00093 );
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109 );
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124 );
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141 );
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157 );
00158
00159
00160
00161
00162
00163
00164
00165
00166 );
00167
00168 #endif

```

## 7.15 include/lock\_ctrl.h File Reference

Door lock control interface using a servo and light sensor.

```
#include <Servo.h>
```

### Macros

- **#define SERVO\_PIN 16**  
*GPIO pin connected to the servo motor.*
- **#define CLOSED\_THRESHOLD 70**  
*Light threshold indicating a closed box.*
- **#define LIGHT\_PIN A0**  
*GPIO pin connected to light detector.*
- **#define OPEN\_THRESHOLD 100**  
*Light threshold indicating an open box.*

## Functions

- void `lock_ctrl_init ()`  
*Initializes the lock control module.*
- void `lock_door ()`  
*Locks the door using the servo.*
- void `unlock_door ()`  
*Unlocks the door using the servo.*
- bool `is_box_closed ()`  
*Checks whether the box is closed.*
- void `play_open ()`  
*Plays the sound effect for door opening.*
- void `play_close ()`  
*Play sound effect when the door closes.*

### 7.15.1 Detailed Description

#### Author

Amal Araweelo Almis

### 7.15.2 Macro Definition Documentation

#### 7.15.2.1 CLOSED\_THRESHOLD

```
#define CLOSED_THRESHOLD 70
```

#### 7.15.2.2 LIGHT\_PIN

```
#define LIGHT_PIN A0
```

#### 7.15.2.3 OPEN\_THRESHOLD

```
#define OPEN_THRESHOLD 100
```

#### 7.15.2.4 SERVO\_PIN

```
#define SERVO_PIN 16
```

### 7.15.3 Function Documentation

#### 7.15.3.1 `is_box_closed()`

```
bool is_box_closed ()
```

Uses the light sensor to determine door state.

##### Returns

true Box is closed.

false Box is open.

#### 7.15.3.2 `lock_ctrl_init()`

```
void lock_ctrl_init ()
```

Must be called once during system startup before any other lock control functions are used.

#### 7.15.3.3 `lock_door()`

```
void lock_door ()
```

#### 7.15.3.4 `play_close()`

```
void play_close ()
```

#### 7.15.3.5 `play_open()`

```
void play_open ()
```

#### 7.15.3.6 `unlock_door()`

```
void unlock_door ()
```

## 7.16 lock\_ctrl.h

[Go to the documentation of this file.](#)

```

00001
00006
00007 #ifndef LOCK_CTRL_H
00008 #define LOCK_CTRL_H
00009
00010 #include <Servo.h>
00011
00012 // Pins
00014 #define SERVO_PIN 16
00015
00017 #define CLOSED_THRESHOLD 70
00018
00020 #define LIGHT_PIN A0
00021
00023 #define OPEN_THRESHOLD 100      // to avoid issues
00024
00031 void lock_ctrl_init();
00032
00033 // Actions
00037 void lock_door();
00038
00042 void unlock_door();
00043
00044 // Photosensor state
00053 bool is_box_closed();
00054
00058 void play_open();
00059
00063 void play_close();
00064
00065 #endif

```

## 7.17 include/login\_html.h File Reference

HTML file for the login page.

```
#include <pgmspace.h>
```

### Variables

- const char LOGIN\_HTML[] PROGMEM  
*HTML for the login page.*

### 7.17.1 Detailed Description

#### Authors

Baldur G. Toftegaard

### 7.17.2 Variable Documentation

#### 7.17.2.1 PROGMEM

```
const char LOGIN_HTML [ ] PROGMEM
```

Opening container for the sales graph.

Interpreted as a string by the compiler.

## 7.18 login\_html.h

[Go to the documentation of this file.](#)

```

00001
00006
00007 #ifndef LOGIN_HTML_H
00008 #define LOGIN_HTML_H
00009
00010 #include <pgmspace.h>
00011
00015 const char LOGIN_HTML[] PROGMEM = R"rawliteral(
00016     <!DOCTYPE html>
00017     <html>
00018         <head>
00019             <meta charset="utf-8">
00020             <title>Login</title>
00021             <link rel="stylesheet" href="/style.css">
00022     </head>
00023     <body>
00024         <div class="login_box">
00025             <h2>
00026                 Login
00027             </h2>
00028             <form action="/login" method="POST">
00029                 <label>
00030                     Username
00031                 </label>
00032                 <br>
00033                     <input type="text" name="user">
00034                 <br>
00035                 <br>
00036                     <label>
00037                         Password
00038                     </label>
00039                     <br>
00040                         <input type="password" name="pass">
00041                     <br>
00042                     <br>
00043                     <a href="/"><input type="button" value="Back"></a>
00044                     <a href="/admin"><input type="button" value="Login"></a>
00045                 </form>
00046             </div>
00047         </body>
00048     </html>
00049 )rawliteral";
00050
00051 #endif

```

## 7.19 include/rfid\_access.h File Reference

Declare functions related to RFID reading and user database.

```
#include <SPI.h>
#include <MFRC522.h>
```

### Classes

- struct **User**

*User record stored in the RFID user database.*

### Macros

- #define **SS\_PIN** 15  
*GPIO pin connected to SS.*
- #define **RST\_PIN** 0  
*GPIO pin connected to RST.*
- #define **MAX\_ROOMS** 17  
*Maximum number of rooms supported by the system.*
- #define **UID\_LENGTH** 4  
*Length of an RFID UID in bytes.*

## Enumerations

- enum `RFIDcommand` {  
  `CMD_NONE` , `CMD_ADD_USER` , `CMD_OPEN` , `CMD_LOCK` ,  
  `CMD_REMOVE_USER` , `CMD_CONFIRM` , `CMD_PRINT` }

*Supported serial commands for the RFID management interface.*

## Functions

- `RFIDcommand check_command (void)`  
*brief Reads a command from the serial interface and maps it to an `RFIDcommand`.*
- `void setup_RFID_reader (MFRC522 &rfid)`  
*Initializes SPI and the MFRC522 RFID reader.*
- `bool add_user (MFRC522 &rfid)`  
*Adds a new user by reading room number and scanning an RFID tag.*
- `bool remove_user ()`  
*Removes a user from the database.*
- `bool compare_UID (byte *uid1, byte *uid2)`  
*Compares two RFID UIDs.*
- `bool read_RFID_tag (MFRC522 &rfid, byte *uidBuffer)`  
*Reads an RFID tag UID from the MFRC522 reader.*
- `void display_commands (void)`  
*Prints the available serial commands.*
- `void display_commands_um ()`  
*Prints the available serial commands for user-management mode.*
- `void get_users_db (User *ptr)`  
*Copies the current user database to a provided buffer.*
- `void user_management (RFIDcommand cmd, User *ptr, MFRC522 &rfid)`  
*Executes user-management actions based on the provided command.*
- `bool validate_rfid (MFRC522 myRFID)`  
*Validates an RFID tag against the registered user database.*
- `void print_single_user (User *ptr, int idx)`  
*brief Prints a single user entry to the serial interface.*
- `void print_all_users (User *ptr)`  
*Prints all users in the database to the serial interface.*
- `void print_uid (byte *ptr)`  
*Prints a UID buffer to the serial interface.*
- `int read_integer ()`  
*Reads an integer from the serial interface.*
- `bool read_confirmation ()`  
*Reads a confirmation input from the serial interface.*
- `int find_empty_index (User *ptr)`  
*brief Finds an empty slot in the user database.*
- `int count_rooms (User *ptr)`  
*brief Counts the number of occupied user entries in the database.*
- `void rfid_set_last_uid (const byte *uidIn)`  
*Function for storing the last used RFID.*
- `bool rfid_get_last_uid (byte *uidOut)`  
*Function for restoring the last used RFID.*

## Variables

- User users [MAX\_ROOMS]  
*Global user database array.*
- int userCount  
*Number of currently registered users.*

## 7.19.1 Detailed Description

### Author

Amal Araweelo Almis  
Anssi Sohlman  
Baldur G. Toftegaard

## 7.19.2 Macro Definition Documentation

### 7.19.2.1 MAX\_ROOMS

```
#define MAX_ROOMS 17
```

### 7.19.2.2 RST\_PIN

```
#define RST_PIN 0
```

### 7.19.2.3 SS\_PIN

```
#define SS_PIN 15
```

### 7.19.2.4 UID\_LENGTH

```
#define UID_LENGTH 4
```

## 7.19.3 Enumeration Type Documentation

### 7.19.3.1 RFIDcommand

```
enum RFIDcommand
```

#### Enumerator

CMD_NONE	
----------	--

CMD_ADD_USER	
CMD_OPEN	
CMD_LOCK	
CMD_REMOVE_USER	
CMD_CONFIRM	
CMD_PRINT	

## 7.19.4 Function Documentation

### 7.19.4.1 add\_user()

```
bool add_user (
    MFRC522 & rfid)
```

#### Parameters

<i>rfid</i>	An instance of a MFRC522 object.
-------------	----------------------------------

#### Returns

true [User](#) was added.  
false [User](#) was not added.

### 7.19.4.2 check\_command()

```
RFIDcommand check_command (
    void )
```

#### Returns

[RFIDcommand](#)

### 7.19.4.3 compare\_UID()

```
bool compare_UID (
    byte * uid1,
    byte * uid2)
```

#### Parameters

<i>uid1</i>	The first uid number.
<i>uid2</i>	The uid number we want to compare it to.

#### Returns

true Numbers match.  
false Numbers do not match.

#### 7.19.4.4 count\_rooms()

```
int count_rooms (
    User * ptr)
```

##### Parameters

<i>ptr</i>	A pointer to the users[] array
------------	--------------------------------

##### Returns

int Number of rooms

#### 7.19.4.5 display\_commands()

```
void display_commands (
    void )
```

#### 7.19.4.6 display\_commands\_um()

```
void display_commands_um ()
```

#### 7.19.4.7 find\_empty\_index()

```
int find_empty_index (
    User * ptr)
```

##### Parameters

<i>ptr</i>	A pointer to the users[] array
------------	--------------------------------

##### Returns

int The empty index.

#### 7.19.4.8 get\_users\_db()

```
void get_users_db (
    User * ptr)
```

##### Parameters

<i>ptr</i>	A pointer to the users[] array
------------	--------------------------------

#### 7.19.4.9 print\_all\_users()

```
void print_all_users (
    User * ptr)
```

##### Parameters

<i>ptr</i>	A pointer to the users[] array
------------	--------------------------------

#### 7.19.4.10 print\_single\_user()

```
void print_single_user (
    User * ptr,
    int idx)
```

##### Parameters

<i>ptr</i>	A pointer to the users[] array
<i>idx</i>	The index to be printed

#### 7.19.4.11 print\_uid()

```
void print_uid (
    byte * ptr)
```

##### Parameters

<i>ptr</i>	A pointer to member of the users[] array
------------	--

#### 7.19.4.12 read\_confirmation()

```
bool read_confirmation ()
```

##### Returns

true Confirmed.

false Not confirmed.

#### 7.19.4.13 read\_integer()

```
int read_integer ()
```

##### Returns

int Integer read from serial interface.

#### 7.19.4.14 `read_RFID_tag()`

```
bool read_RFID_tag (
    MFRC522 & rfid,
    byte * uidBuffer)
```

##### Parameters

<i>rfid</i>	An instance of a MFRC522 object
-------------	---------------------------------

<i>uidBuffer</i>	An array where the UID is saved to
------------------	------------------------------------

**Returns**

true RFID tag was read.  
false No RFID tag was read.

**7.19.4.15 remove\_user()**

```
bool remove_user ()
```

**Returns**

true [User](#) was removed.  
false [User](#) was not removed.

**7.19.4.16 rfid\_get\_last\_uid()**

```
bool rfid_get_last_uid (
    byte * uidOut)
```

**Parameters**

<i>uidOut</i>	Pointer to user it.
---------------	---------------------

**Returns**

true There was an last uid to retrive.  
false No last uid was retrived.

**7.19.4.17 rfid\_set\_last\_uid()**

```
void rfid_set_last_uid (
    const byte * uidIn)
```

**Parameters**

<i>uidOut</i>	Pointer to user it.
---------------	---------------------

**Returns**

true Stored last used uid.  
false Last uid was not stored

#### 7.19.4.18 setup\_RFID\_reader()

```
void setup_RFID_reader (
    MFRC522 & rfid)
```

##### Parameters

<i>rfid</i>	An instance of a MFRC522 object
-------------	---------------------------------

#### 7.19.4.19 user\_management()

```
void user_management (
    RFIDcommand cmd,
    User * ptr,
    MFRC522 & rfid)
```

##### Parameters

<i>cmd</i>	The current <a href="#">RFIDcommand</a> (add or remove user)
<i>ptr</i>	A pointer to the users[] array
<i>rfid</i>	An instance of a MFRC522 object

#### 7.19.4.20 validate\_rfid()

```
bool validate_rfid (
    MFRC522 myRFID)
```

##### Parameters

<i>myRFID</i>	An instance of a MFRC522 object
---------------	---------------------------------

##### Returns

true The RFID was a match.  
false There was no matching RFID in the database.

### 7.19.5 Variable Documentation

#### 7.19.5.1 userCount

```
int userCount [extern]
```

#### 7.19.5.2 users

```
User users[MAX_ROOMS] [extern]
```

## 7.20 rfid\_access.h

Go to the documentation of this file.

```
00001
00008
00009 #ifndef RFID_ACCESS_H
00010 #define RFID_ACCESS_H
00011
00012 #include <SPI.h>
00013 #include <MFRC522.h>
00014
00015 // Pins
00017 #define SS_PIN 15 // Use GPIO pins for HUZZAH instead of D8
00018
00020 #define RST_PIN 0 // Instead of D3
00021
00022 // Constants
00024 #define MAX_ROOMS 17
00025
00027 #define UID_LENGTH 4
00028
00032 enum RFIDcommand {
00033     CMD_NONE,
00034     CMD_ADD_USER,
00035     CMD_OPEN,
00036     CMD_LOCK,
00037     CMD_REMOVE_USER,
00038     CMD_CONFIRM,
00039     CMD_PRINT
00040 };
00041
00045 struct User {
00046     byte uid[UID_LENGTH];
00047     int roomNumber;
00048     int balance;
00049 };
00050
00054 extern User users[MAX_ROOMS];
00055
00059 extern int userCount;
00060
00066 RFIDcommand check_command(
00067     void
00068 );
00069
00075 void setup_RFID_reader(
00076     MFRC522 &rfid
00077 );
00078
00086 bool add_user(
00087     MFRC522 &rfid
00088 );
00089
00096 bool remove_user(
00097 );
00098
00107 bool compare_UID(
00108     byte *uid1,
00109     byte *uid2
00110 );
00111
00120 bool read_RFID_tag(
00121     MFRC522 &rfid,
00122     byte *uidBuffer
00123 );
00124
00128 void display_commands(
00129     void
00130 );
00131
00135 void display_commands_um(
00136 );
00137
00143 void get_users_db(
00144     User* ptr
00145 );
00146
00154 void user_management(
00155     RFIDcommand cmd,
00156     User* ptr,
00157     MFRC522 &rfid
00158 );
00159
00167 bool validate_rfid(
00168     MFRC522 myRFID
```

```

00169 );
00170
00177 void print_single_user(
00178     User* ptr,
00179     int idx
00180 );
00181
00187 void print_all_users(
00188     User* ptr
00189 );
00190
00196 void print_uid(
00197     byte* ptr
00198 );
00199
00205 int read_integer(
00206 );
00207
00214 bool read_confirmation(
00215 );
00216
00223 int find_empty_index(
00224     User* ptr
00225 );
00226
00233 int count_rooms(
00234     User* ptr
00235 );
00236
00243 void rfid_set_last_uid(
00244     const byte *uidIn
00245 );
00246
00253 bool rfid_get_last_uid(
00254     byte *uidOut
00255 );
00256
00257
00258 #endif

```

## 7.21 include/sale\_html.h File Reference

Headerfile for displaying the graph of sales on the main page.

```
#include <Arduino.h>
#include <ESP8266WebServer.h>
#include <pgmspace.h>
```

### Functions

- void **send\_sale\_html\_graph** (ESP8266WebServer &**server**, uint8\_t room\_number, const char \*bar\_type, int bar\_height)  
*Sends a single sales bar element to the client.*
- void **send\_sale\_html\_page** (ESP8266WebServer &**server**, uint8\_t room\_count, const int \*greenHeight, const int \*classicHeights)  
*Sends the complete sales graph page.*

### Variables

- const char SALE\_BOX\_START[] PROGMEM  
*Opening container for the sales graph.*

## 7.21.1 Detailed Description

### Author

Baldur G. Toftegaard

## 7.21.2 Function Documentation

### 7.21.2.1 send\_sale\_html\_graph()

```
void send_sale_html_graph (
    ESP8266WebServer & server,
    uint8_t room_number,
    const char * bar_type,
    int bar_height)
```

#### Parameters

<i>server</i>	The server instance.
<i>room_number</i>	The room number operated on.
<i>bar_type</i>	Type of the bar operated on.
<i>bar_height</i>	Height of the bre operated on.

### 7.21.2.2 send\_sale\_html\_page()

```
void send_sale_html_page (
    ESP8266WebServer & server,
    uint8_t room_count,
    const int * greenHeight,
    const int * classicHeights)
```

#### Parameters

<i>server</i>	The server instance.
<i>room_count</i>	Number of registered rooms.
<i>greenHeight</i>	The height of green bars.
<i>classicHeights</i>	The height of clasic bars.

## 7.21.3 Variable Documentation

### 7.21.3.1 PROGMEM

```
const char SALE_BOX_STOP [ ] PROGMEM [extern]
```

Closing container for the complete sales graph.

Closing container for a single room graph.  
Closing fragment for a single sales bar.  
HTML fragment defining the height style of a bar.  
HTML fragment defining the CSS class type for a bar.  
HTML fragment for the room identifier.  
Opening container for a single room graph.  
Opening container for the sales graph.  
Interperated by compiler as a string.  
Opening container for the sales graph.  
String used for the HTML footer.  
This is responsible for updating the graphs. read by the compiler as a string  
Opening container for the sales graph.  
Interpreted as a string by the compiler.  
Opening container for the sales graph.  
is handled like a string by the compiler

## 7.22 sale\_html.h

[Go to the documentation of this file.](#)

```
00001
00006
00007 #ifndef SALE_HTML_H
00008 #define SALE_HTML_H
00009
00010 #include <Arduino.h>
00011 #include <ESP8266WebServer.h>
00012 #include <pgmspace.h>
00013
00017 extern const char SALE_BOX_START[] PROGMEM;
00018
00022 extern const char SALE_BOX_ROOM_START[] PROGMEM;
00023
00027 extern const char SALE_BOX_ROOM_ID[] PROGMEM;
00028
00032 extern const char SALE_BOX_ROOM_CLASS_TYPE[] PROGMEM;
00033
00037 extern const char SALE_BOX_ROOM_CLASS_HEIGHT[] PROGMEM;
00038
00042 extern const char SALE_BOX_ROOM_END[] PROGMEM;
00043
00047 extern const char SALE_BOX_ROOM_STOP[] PROGMEM;
00048
00052 extern const char SALE_BOX_STOP[] PROGMEM;
00053
00061 void send_sale_html_graph(
00062     ESP8266WebServer &server,
00063     uint8_t room_number,
00064     const char *bar_type,
00065     int bar_height
00066 );
00067
00075 void send_sale_html_page(
00076     ESP8266WebServer &server,
00077     uint8_t room_count,
00078     const int *greenHeight,
00079     const int *classicHeights
00080 );
00081
00082 #endif
```

## 7.23 include/style\_css.h File Reference

CSS served at /style.css used for setting the style for the web server.

```
#include <pgmspace.h>
```

### Variables

- const char STYLE\_CSS[] PROGMEM  
*String containing the CSS styling of the webserver.*

### 7.23.1 Detailed Description

The css is handled as a string by the compiler.

#### Author

Baldur G. Toftegaard

### 7.23.2 Variable Documentation

#### 7.23.2.1 PROGMEM

```
const char STYLE_CSS [ ] PROGMEM
```

Opening container for the sales graph.

is handled like a string by the compiler

## 7.24 style\_css.h

[Go to the documentation of this file.](#)

```
00001
00009
00010 #ifndef STYLE_CSS_H
00011 #define STYLE_CSS_H
00012 #include <pgmspace.h>
00013
00017 const char STYLE_CSS[] PROGMEM = R"rawliteral(
00018 /* ----- Global page styling ----- */
00019 body {
00020   margin: 0;
00021   padding: 0;
00022   font-family: Arial, Helvetica, sans-serif;
00023   background-color: #f0f0f0;
00024 }
00025
00026 /* Bar on top of page for displaying message/path and log in */
00027 .topbar {
00028   display: flex;
00029   justify-content: space-between;
00030   align-items: center;
00031   background-color: #2c3e50;
00032   color: #fff;
00033   padding: 12px 16px;
```

```

00034   box-sizing: border-box;
00035 }
00036
00037 /* message/path display box */
00038 .topbar .right a {
00039   color: #fff;
00040   text-decoration: none;
00041   border: 1px solid rgba(255,255,255,0.5);
00042   padding: 6px 10px;
00043   border-radius: 6px;
00044 }
00045
00046 /* login button */
00047 .topbar .right a:hover {
00048   background: rgba(255,255,255,0.15);
00049 }
00050
00051 /* box for the graph to be placed inside */
00052 .sale_box {
00053   margin: 40px auto;
00054   width: calc(100% - 100px);
00055   height: 500px;
00056
00057   background-color: #ffffff;
00058   border: 2px solid #000000;
00059   box-sizing: border-box;
00060
00061   display: flex;
00062   justify-content: space-evenly;
00063   align-items: flex-end;
00064   padding: 15px;
00065   gap: 0;
00066 }
00067
00068 /* graph box for each room */
00069 .sale_room {
00070   display: flex;
00071   align-items: flex-end;
00072   gap: 4px;
00073   height: 100%;
00074 }
00075
00076 /* Bars */
00077 .sale_pole_green {
00078   width: 22px;
00079   border: 1px solid #000000;
00080   box-sizing: border-box;
00081 }
00082
00083 .sale_pole_clasic {
00084   width: 22px;
00085   border: 1px solid #ffffff;
00086   box-sizing: border-box;
00087 }
00088
00089 .sale_pole_green { background-color: #2e7d32; }
00090 .sale_pole_clasic { background-color: #ffffff; }
00091 )rawliteral";
00092
00093 #endif

```

## 7.25 include/weight\_scale.h File Reference

```
#include <Arduino.h>
#include <HX711_ADC.h>
#include <math.h>
```

### Macros

- **#define BEER\_WEIGHT 350**  
*Define the weight of a beer.*
- **#define SCALE\_TOL 25**  
*Define the uncertainty of the weight measurement.*

- #define HX711\_DOUT 4
- #define HX711\_SCK 5
- #define SCALE\_DEFAULT\_SETTLE\_TIME\_MS 3000

## Functions

- float `get_weight_reference` (void)  
*Function to get the weight reference.*
- void `set_weight_reference` (float value)  
*Function to set the weight reference.*
- void `reset_weight_reference` (void)  
*Function to reset the weight reference.*
- bool `weight_reference_is_set` (void)  
*Function to send confirmation that the weight reference is set.*
- void `setup_scale` (float calFactor)  
*Function to seting up the scale, is called in the begining of the program.*
- bool `update_scale` (void)  
*Function for updating the scale value.*
- float `get_weight` (void)  
*Function to get the scale reading.*
- void `tare_scale` (void)  
*Function to tare the scale.*
- bool `tare_complete` (void)  
*Function to signal that the scale has been tared.*
- int `get_beer_cans_taken` (float referencWeight, float currentWeight)  
*Function to get the number of beer cans taken.*

## Variables

- HX711\_ADC `scale`

### 7.25.1 Detailed Description

#### Author

Amal Araweelo Almis

### 7.25.2 Macro Definition Documentation

#### 7.25.2.1 BEER\_WEIGHT

```
#define BEER_WEIGHT 350
```

#### 7.25.2.2 HX711\_DOUT

```
#define HX711_DOUT 4
```

### 7.25.2.3 HX711\_SCK

```
#define HX711_SCK 5
```

### 7.25.2.4 SCALE\_DEFAULT\_SETTLE\_TIME\_MS

```
#define SCALE_DEFAULT_SETTLE_TIME_MS 3000
```

### 7.25.2.5 SCALE\_TOL

```
#define SCALE_TOL 25
```

## 7.25.3 Function Documentation

### 7.25.3.1 get\_beer\_cans\_taken()

```
int get_beer_cans_taken (
    float referencWeight,
    float currentWeight)
```

#### Parameters

<i>referencWeight</i>	The set reference weight [g]
<i>currentWeight</i>	The current mesured weight [g]

#### Returns

int number of cans taken

### 7.25.3.2 get\_weight()

```
float get_weight (
    void )
```

#### Returns

float Value read from weight.

### 7.25.3.3 get\_weight\_reference()

```
float get_weight_reference (
    void )
```

#### Returns

float The weight reference.

#### 7.25.3.4 reset\_weight\_reference()

```
void reset_weight_reference (
    void )
```

#### 7.25.3.5 set\_weight\_reference()

```
void set_weight_reference (
    float value)
```

##### Parameters

value	Reference value.
-------	------------------

#### 7.25.3.6 setup\_scale()

```
void setup_scale (
    float calFactor)
```

##### Parameters

calFactor	The calibration factor.
-----------	-------------------------

#### 7.25.3.7 tare\_complete()

```
bool tare_complete (
    void )
```

##### Returns

- true The scale was tared.
- false The scale was not tared.

#### 7.25.3.8 tare\_scale()

```
void tare_scale (
    void )
```

#### 7.25.3.9 update\_scale()

```
bool update_scale (
    void )
```

##### Returns

- true Scale was updated.
- false Scale was not updated.

### 7.25.3.10 weight\_reference\_is\_set()

```
bool weight_reference_is_set (
    void )
```

#### Returns

true Weight reference is set.  
false Weight reference is not set.

## 7.25.4 Variable Documentation

### 7.25.4.1 scale

```
HX711_ADC scale [extern]
```

## 7.26 weight\_scale.h

[Go to the documentation of this file.](#)

```
00001
00005
00006 #ifndef WEIGHT_SCALE_H
00007 #define WEIGHT_SCALE_H
00008
00009 #include <Arduino.h>
00010 #include <HX711_ADC.h>
00011 #include <math.h>
00012
00013 float get_weight_reference(
00014     void
00015 );
00016
00017 void set_weight_reference(
00018     float value
00019 );
00020
00021
00022 void reset_weight_reference(
00023     void
00024 );
00025
00026 bool weight_reference_is_set(
00027     void
00028 );
00029
00030
00031
00032
00033
00034
00035
00036
00037
00038
00039
00040
00041
00042
00043
00044
00045
00046
00047
00048
00049
00050
00051 #define BEER_WEIGHT 350
00052
00053
00054
00055
00056 #define SCALE_TOL 25
00057
00058 // Pins
00059 #define HX711_DOUT 4 // GPIO4=D2
00060 #define HX711_SCK 5 // GPIO5=D1
00061
00062 // Config scale
00063 #define SCALE_DEFAULT_SETTLE_TIME_MS 3000
00064
00065 // Globals
00066 extern HX711_ADC scale;
00067
00068
00069 void setup_scale(
00070     float calFactor
00071 );
00072
00073 bool update_scale(
00074     void
00075 );
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089 float get_weight(
```

```

00094     void
00095 );
00096
00100 void tare_scale(
00101     void
00102 );
00103
00110 bool tare_complete(
00111     void
00112 );
00113
00121 int get_beer_cans_taken(
00122     float referencWeight,
00123     float currentWeight
00124 );
00125
00126 #endif

```

## 7.27 README.md File Reference

## 7.28 src/buzzer.cpp File Reference

```
#include "buzzer.h"
```

### Functions

- void **play\_warning** (unsigned long t)  
*Plays short sound when door is closed and about to be locked.*
- void **play\_unlock** ()  
*Plays an ascending tone, when the door unlocks.*
- void **play\_lock** ()  
*Play a descending tone, when the door locks.*

### Variables

- const int **BUZZERPIN** = 2  
*GPIO pin connected to the buzzer.*
- const double **HIGH\_TONE** = 1000  
*Frequency (Hz) used for high-tone buzzer sound.*
- const double **LOW\_TONE** = 600  
*Frequency (Hz) used for low-tone buzzer sound.*
- const unsigned long **TONE\_LENGTH** = 200  
*Duration of buzzer tone in milliseconds.*

### 7.28.1 Function Documentation

#### 7.28.1.1 play\_lock()

```
void play_lock ()
```

### 7.28.1.2 play\_unlock()

```
void play_unlock ()
```

### 7.28.1.3 play\_warning()

```
void play_warning (
    unsigned long t)
```

#### Parameters

<i>t</i>	Time passed since door has been closed
----------	--

## 7.28.2 Variable Documentation

### 7.28.2.1 BUZZERPIN

```
const int BUZZERPIN = 2
```

### 7.28.2.2 HIGH\_TONE

```
const double HIGH_TONE = 1000
```

### 7.28.2.3 LOW\_TONE

```
const double LOW_TONE = 600
```

### 7.28.2.4 TONE\_LENGTH

```
const unsigned long TONE_LENGTH = 200
```

## 7.29 src/database\_management.cpp File Reference

Functions for user management and read/write of non-volatile memory on ESP8266.

```
#include "rfid_access.h"
#include <EEPROM.h>
```

## Functions

- void `user_management` (`RFIDcommand` incomingCommand, `User` \*ptr, `MFRC522` &`rfid`)  
*Executes user-management actions based on the provided command.*
- bool `remove_user` ()  
*Removes a user from the database.*
- void `get_users_db` (`User` \*ptr)  
*Copies the current user database to a provided buffer.*
- void `print_all_users` (`User` \*ptr)  
*Prints all users in the database to the serial interface.*
- void `print_single_user` (`User` \*ptr, int idx)  
*brief Prints a single user entry to the serial interface.*
- void `print_uid` (byte \*ptr)  
*Prints a UID buffer to the serial interface.*
- int `read_integer` ()  
*Reads an integer from the serial interface.*
- bool `read_confirmation` ()  
*Reads a confirmation input from the serial interface.*
- int `find_empty_index` (`User` \*ptr)  
*brief Finds an empty slot in the user database.*
- int `count_rooms` (`User` \*ptr)  
*brief Counts the number of occupied user entries in the database.*

### 7.29.1 Detailed Description

#### Authors

Anssi Sohlman,

### 7.29.2 Function Documentation

#### 7.29.2.1 `count_rooms()`

```
int count_rooms (
    User * ptr)
```

#### Parameters

<code>ptr</code>	A pointer to the <code>users[]</code> array
------------------	---

#### Returns

`int` Number of rooms

### 7.29.2.2 find\_empty\_index()

```
int find_empty_index (
    User * ptr)
```

#### Parameters

<i>ptr</i>	A pointer to the users[] array
------------	--------------------------------

#### Returns

int The empty index.

### 7.29.2.3 get\_users\_db()

```
void get_users_db (
    User * ptr)
```

#### Parameters

<i>ptr</i>	A pointer to the users[] array
------------	--------------------------------

### 7.29.2.4 print\_all\_users()

```
void print_all_users (
    User * ptr)
```

#### Parameters

<i>ptr</i>	A pointer to the users[] array
------------	--------------------------------

### 7.29.2.5 print\_single\_user()

```
void print_single_user (
    User * ptr,
    int idx)
```

#### Parameters

<i>ptr</i>	A pointer to the users[] array
<i>idx</i>	The index to be printed

### 7.29.2.6 print\_uid()

```
void print_uid (
    byte * ptr)
```

#### Parameters

<i>ptr</i>	A pointer to member of the users[] array
------------	--

### 7.29.2.7 read\_confirmation()

```
bool read_confirmation ()
```

#### Returns

- true Confirmed.
- false Not confirmed.

### 7.29.2.8 read\_integer()

```
int read_integer ()
```

#### Returns

- int Integer read from serial interface.

### 7.29.2.9 remove\_user()

```
bool remove_user ()
```

#### Returns

- true [User](#) was removed.
- false [User](#) was not removed.

### 7.29.2.10 user\_management()

```
void user_management (
    RFIDcommand cmd,
    User * ptr,
    MFRC522 & rfid)
```

#### Parameters

<i>cmd</i>	The current <a href="#">RFIDcommand</a> (add or remove user)
------------	--

<i>ptr</i>	A pointer to the users[] array
<i>rfid</i>	An instance of a MFRC522 object

## 7.30 src/fridge\_state.cpp File Reference

File used for defining the fridge inventory, definition of user inventory shuld also be placed here.

```
#include "fridge_state.h"
```

### Variables

- **inventory\_fridge**

*Used to set up the fridge inventory.*

### 7.30.1 Detailed Description

#### Author

Baldur G. Toftegaard

### 7.30.2 Variable Documentation

#### 7.30.2.1 **fridge**

```
inventory_fridge
```

## 7.31 src/graph\_data.cpp File Reference

File used to set up the graph section in the web server.

```
#include "graph_data.h"
```

### Functions

- void **graph\_add\_to\_room\_green** (uint8\_t roomNumber, int delta)
 

*Adds a value to the green sales bar of a given room.*
- void **graph\_add\_to\_room\_clasic** (uint8\_t roomNumber, int delta)
 

*Adds a value to the classic sales bar of a given room.*

## Variables

- int `greenHeight [ROOM_COUNT]`  
*Sales graph height data for green beer.*
- int `classicHeight [ROOM_COUNT]`  
*Sales graph height data for classic beer.*

### 7.31.1 Detailed Description

#### Author

Baldur G. Toftegaard

### 7.31.2 Function Documentation

#### 7.31.2.1 `graph_add_to_room_clasic()`

```
void graph_add_to_room_clasic (
    uint8_t roomNumber,
    int delta)
```

#### Parameters

<code>roomNumber</code>	Room operated on.
<code>delta</code>	Incremental value used to update the bar height.

#### 7.31.2.2 `graph_add_to_room_green()`

```
void graph_add_to_room_green (
    uint8_t roomNumber,
    int delta)
```

#### Parameters

<code>roomNumber</code>	Room operated on.
<code>delta</code>	Incremental value used to update the bar height.

### 7.31.3 Variable Documentation

#### 7.31.3.1 `classicHeight`

```
int classicHeight[ROOM_COUNT] [extern]
```

Height values for classic product sales per room.

Indexed by room number. Used by `sale_html` and the `/saleHeights` endpoint.

### 7.31.3.2 greenHeight

```
int greenHeight[ROOM_COUNT] [extern]
```

Height values for green product sales per room.

Indexed by room number. Used by sale\_html and the /saleHeights endpoint.

## 7.32 src/init\_users\_and\_sale.cpp File Reference

```
#include "init_users_and_sale.h"
#include <math.h>
#include "weight_scale.h"
#include "rfid_access.h"
#include "graph_data.h"
```

### Functions

- void [init\\_users\\_and\\_products\(\)](#)  
*Not used in prototype, shuld be moved to [inventory.cpp](#).*
- static float [read\\_current\\_weight\\_blocking](#) (uint32\_t timeoutMs=1200)
- void [perform\\_sale](#) ([inventory](#) \*fridge\_inventory)  
*Performs a sale between a user and the fridge.*

### 7.32.1 Function Documentation

#### 7.32.1.1 init\_users\_and\_products()

```
void init_users_and_products ()
```

Old function for initializing users and products.

#### 7.32.1.2 perform\_sale()

```
void perform_sale (
    inventory * fridge_inventory)
```

### Parameters

<i>weight</i>	- Read the value from the weight.
<i>user_id</i>	- The user the sale shuld be registered to.
<i>fridge_inventory</i>	- The inventory we wish to remove the beverage from.

### Parameters

<i>fridge_inventory</i>	Inventory that the sale shuld remove item from
-------------------------	--

### 7.32.1.3 `read_current_weight_blocking()`

```
float read_current_weight_blocking (
    uint32_t timeoutMs = 1200) [static]
```

## 7.33 src/inventory.cpp File Reference

Functions responsible for keeping track of the fridge inventory.

```
#include "inventory.h"
#include <string.h>
```

### Functions

- void `inventory_init (inventory *inventory)`  
*Initializes an inventory structure.*
- `product inventory_make_product (const char *product_name, beverage_type type, uint16_t weight, uint8_t price)`  
*Creates a new product instance.*
- bool `inventory_add_product (inventory *inventory, product product, uint16_t quantity)`  
*Adds a new product to the inventory.*
- bool `inventory_remove_product (inventory *inventory, product product)`  
*Removes a product entirely from the inventory.*
- bool `inventory_add_beverage (inventory *inventory, product beverag, uint8_t amount)`
- bool `inventory_remove_beverage (inventory *inventory, product beverag, uint8_t amount)`  
*Function for removing from the amount of a beverage in an inventory.*
- void `inventory_print (inventory *inventory)`  
*Function to print a users inventory.*

### 7.33.1 Detailed Description

#### Author

Baldur G. Toftegaard

### 7.33.2 Function Documentation

#### 7.33.2.1 `inventory_add_beverage()`

```
bool inventory_add_beverage (
    inventory * inventory,
    product beverag,
    uint8_t amount)
```

#### 7.33.2.2 `inventory_add_product()`

```
bool inventory_add_product (
    inventory * inventory,
    product product,
    uint16_t quantity)
```

#### Parameters

<code>inventory</code>	Inventory to add the product to.
------------------------	----------------------------------

<i>product</i>	Product to add.
<i>quantity</i>	Initial quantity of the product.

**Returns**

true - Product successfully added.  
 false - Error occurred (inventory full or duplicate product).

**7.33.2.3 inventory\_init()**

```
void inventory_init (
    inventory * inventory)
```

Clears internal state and prepares the inventory for use.

**Parameters**

<i>inventory</i>	Pointer to inventory instance to initialize.
------------------	--

**7.33.2.4 inventory\_make\_product()**

```
product inventory_make_product (
    const char * name,
    beverage_type type,
    uint16_t weight,
    uint8_t price)
```

**Parameters**

<i>name</i>	Display name of the product.
<i>type</i>	Beverage type.
<i>weight</i>	Weight of the product.
<i>price</i>	Price of the product.

**Returns**

Initialized product instance.

**7.33.2.5 inventory\_print()**

```
void inventory_print (
    inventory * inventory)
```

**Parameters**

<i>inventory</i>	Inventory to print.
------------------	---------------------

### 7.33.2.6 inventory\_remove\_beverage()

```
bool inventory_remove_beverage (
    inventory * inventory,
    product beverage,
    uint8_t amount)
```

#### Parameters

<i>inventory</i>	Inventory to modify.
<i>beverage</i>	Beverage to update.
<i>amount</i>	Quantity to remove.

#### Returns

true - Quantity updated successfully.  
false - Product not found or insufficient stock.

### 7.33.2.7 inventory\_remove\_product()

```
bool inventory_remove_product (
    inventory * inventory,
    product beverage)
```

#### Parameters

<i>inventory</i>	Inventory to remove the product from.
<i>beverage</i>	Product to remove.

#### Returns

true - Product removed successfully.  
false - Product not found.

## 7.34 src/lock\_ctrl.cpp File Reference

Implements servo-based locking control and buzzer feedback for the fridge door.

```
#include "lock_ctrl.h"
#include "buzzer.h"
```

## Functions

- void `lock_ctrl_init ()`  
*Initializes the lock control module.*
- void `lock_door ()`  
*Locks the door using the servo.*
- void `unlock_door ()`  
*Unlocks the door using the servo.*
- bool `is_box_closed ()`  
*Checks whether the box is closed.*
- void `play_open ()`  
*Plays the sound effect for door opening.*
- void `play_close ()`  
*Play sound effect when the door closes.*

## Variables

- static Servo `lockServo`
- static const int `UNLOCK_POS` = 0  
*Servo position corresponding to the unlocked state (mechanically fixed).*
- static const int `LOCK_POS` = 100  
*Servo position corresponding to the locked state.*
- const int `BUZZER` = 2  
*GPIO pin connected to the buzzer.*
- const double `HIGH_TONE` = 1000  
*Frequency (Hz) used for the high-tone buzzer sound.*
- const double `LOW_TONE` = 600  
*Frequency (Hz) used for the low-tone buzzer sound.*
- const unsigned long `TONE_LENGTH` = 200  
*Duration of buzzer tone in milliseconds.*
- static bool `boxClosed` = false  
*Internal latched state indicating whether the box is closed.*

### 7.34.1 Detailed Description

#### Authors

Amal Araweelo Almis

### 7.34.2 Function Documentation

#### 7.34.2.1 `is_box_closed()`

```
bool is_box_closed ()
```

Uses the light sensor to determine door state.

#### Returns

true Box is closed.

false Box is open.

#### 7.34.2.2 lock\_ctrl\_init()

```
void lock_ctrl_init ()
```

Must be called once during system startup before any other lock control functions are used.

#### 7.34.2.3 lock\_door()

```
void lock_door ()
```

#### 7.34.2.4 play\_close()

```
void play_close ()
```

#### 7.34.2.5 play\_open()

```
void play_open ()
```

#### 7.34.2.6 unlock\_door()

```
void unlock_door ()
```

### 7.34.3 Variable Documentation

#### 7.34.3.1 boxClosed

```
bool boxClosed = false [static]
```

#### 7.34.3.2 BUZZER

```
const int BUZZER = 2
```

##### Note

This pin choice may cause reset issues on some boards.

#### 7.34.3.3 HIGH\_TONE

```
const double HIGH_TONE = 1000
```

#### 7.34.3.4 LOCK\_POS

```
const int LOCK_POS = 100 [static]
```

### 7.34.3.5 lockServo

```
Servo lockServo [static]
```

### 7.34.3.6 LOW\_TONE

```
const double LOW_TONE = 600
```

### 7.34.3.7 TONE\_LENGTH

```
const unsigned long TONE_LENGTH = 200
```

### 7.34.3.8 UNLOCK\_POS

```
const int UNLOCK_POS = 0 [static]
```

## 7.35 src/main.cpp File Reference

Combined: Web server + Graph + Scale + RFID access + Lock control.

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <ESP8266WiFiMulti.h>
#include "index_html.h"
#include "sale_html.h"
#include "STYLE_CSS.h"
#include "LOGIN_HTML.h"
#include "ADMIN_HTML.h"
#include "graph_data.h"
#include "inventory.h"
#include "init_users_and_sale.h"
#include "weight_scale.h"
#include "fridge_state.h"
#include "rfid_access.h"
#include "lock_ctrl.h"
#include "buzzer.h"
```

## Functions

- ESP8266WebServer `server (80)`
- MFRC522 `rfid (SS_PIN, RST_PIN)`
- static void `connect_wifi_and_start_mdns ()`
- static void `setup_web_routes ()`
- static void `setup_inventory_and_scale ()`
- static void `setup_rfid_and_lock ()`
- void `setup ()`
- void `loop ()`

## Variables

- static const float **CAL\_FACTOR** = 22.9f  
*Calibration factor used for weight or sensor calculations.*
- static const uint16\_t **START\_BEER\_QTY** = 20  
*Initial quantity of beer available at system startup.*
- const char \* **WIFI\_SSID** = "Baldur's A56"  
*Wi-Fi network SSID used by the device.*
- const char \* **WIFI\_PASS** = "MyPasskeyA56"  
*Wi-Fi network password used by the device.*
- int **greenHeight** [ROOM\_COUNT]  
*Sales graph height data for green beer.*
- int **classicHeight** [ROOM\_COUNT]  
*Sales graph height data for classic beer.*
- product **demo\_beer**
- RFIDCommand **activeCommand** = **CMD\_NONE**
- bool **doorUnlocked** = false
- unsigned long **doorCloseTimer** = 0

## 7.35.1 Detailed Description

### Author

Baldur G. Toftegaard

Anssi Sohlman

## 7.35.2 Function Documentation

### 7.35.2.1 connect\_wifi\_and\_start\_mdns()

```
void connect_wifi_and_start_mdns () [static]
```

### 7.35.2.2 loop()

```
void loop ()
```

### 7.35.2.3 rfid()

```
MFRC522 rfid (
    SS_PIN ,
    RST_PIN )
```

### 7.35.2.4 server()

```
ESP8266WebServer server (
    80 )
```

### 7.35.2.5 setup()

```
void setup ()
```

### 7.35.2.6 setup\_inventory\_and\_scale()

```
void setup_inventory_and_scale () [static]
```

### 7.35.2.7 setup\_rfid\_and\_lock()

```
void setup_rfid_and_lock () [static]
```

### 7.35.2.8 setup\_web\_routes()

```
void setup_web_routes () [static]
```

## 7.35.3 Variable Documentation

### 7.35.3.1 activeCommand

```
RFIDcommand activeCommand = CMD_NONE
```

### 7.35.3.2 CAL\_FACTOR

```
const float CAL_FACTOR = 22.9f [static]
```

### 7.35.3.3 classicHeight

```
int classicHeight[ROOM_COUNT]
```

#### Initial value:

```
    = {  
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1  
    }
```

Height values for classic product sales per room.

Indexed by room number. Used by sale\_html and the /saleHeights endpoint.

### 7.35.3.4 demo\_beer

```
product demo_beer
```

### 7.35.3.5 doorCloseTimer

```
unsigned long doorCloseTimer = 0
```

### 7.35.3.6 doorUnlocked

```
bool doorUnlocked = false
```

### 7.35.3.7 greenHeight

```
int greenHeight [ROOM_COUNT]
```

### Initial value:

Height values for green product sales per room.

Indexed by room number. Used by sale\_html and the /saleHeights endpoint.

#### **7.35.3.8 START\_BEER\_QTY**

```
const uint16_t START_BEER_QTY = 20 [static]
```

### **7.35.3.9 WIFI\_PASS**

```
const char* WIFI_PASS = "MyPasskeyA56";
```

### **7.35.3.10 WIFI\_SSID**

```
const char* WIFI_SSID = "Baldrus A56"
```

## 7.36 src/rfid\_access.cpp File Reference

File used to setup the RFID.

```
#include "rfid access.h"
```

## Functions

- `RFIDcommand check_command ()`  
*brief Reads a command from the serial interface and maps it to an `RFIDcommand`.*
- `void setup_RFID_reader (MFRC522 &rfid)`  
*Initializes SPI and the MFRC522 RFID reader.*
- `bool add_user (MFRC522 &rfid)`  
*Adds a new user by reading room number and scanning an RFID tag.*
- `bool validate_rfid (MFRC522 myRFID)`  
*Validates an RFID tag against the registered user database.*
- `bool compare_UID (byte *uid1, byte *uid2)`  
*Compares two RFID UIDs.*
- `bool read_RFID_tag (MFRC522 &rfid, byte *uidBuffer)`  
*Reads an RFID tag UID from the MFRC522 reader.*
- `void display_commands ()`  
*Prints the available serial commands.*
- `void display_commands_um ()`  
*Prints the available serial commands for user-management mode.*
- `void rfid_set_last_uid (const byte *uidIn)`  
*Function for storing the last used RFID.*
- `bool rfid_get_last_uid (byte *uidOut)`  
*Function for restoring the last used RFID.*

## Variables

- `User users [MAX_ROOMS]`  
*Global user database array.*
- `int userCount = 0`  
*Number of currently registered users.*
- `static byte lastUID [UID_LENGTH]`
- `static bool hasUID = false`

### 7.36.1 Detailed Description

#### Author

Amal Araweelo Almis  
 Baldur G. Toftegaard

### 7.36.2 Function Documentation

#### 7.36.2.1 add\_user()

```
bool add_user (
    MFRC522 & rfid)
```

#### Parameters

<code>rfid</code>	An instance of a MFRC522 object.
-------------------	----------------------------------

#### Returns

true `User` was added.  
 false `User` was not added.

### 7.36.2.2 check\_command()

```
RFIDcommand check_command (
    void )
```

Returns

`RFIDcommand`

### 7.36.2.3 compare\_UID()

```
bool compare_UID (
    byte * uid1,
    byte * uid2)
```

Parameters

<code>uid1</code>	The first uid number.
<code>uid2</code>	The uid number we want to compare it to.

Returns

true Numbers match.  
false Numbers do not match.

### 7.36.2.4 display\_commands()

```
void display_commands (
    void )
```

### 7.36.2.5 display\_commands\_um()

```
void display_commands_um ()
```

### 7.36.2.6 read\_RFID\_tag()

```
bool read_RFID_tag (
    MFRC522 & rfid,
    byte * uidBuffer)
```

Parameters

<code>rfid</code>	An instance of a MFRC522 object
<code>uidBuffer</code>	An array where the UID is saved to

Returns

true RFID tag was read.  
false No RFID tag was read.

### 7.36.2.7 rfid\_get\_last\_uid()

```
bool rfid_get_last_uid (
    byte * uidOut)
```

#### Parameters

<i>uidOut</i>	Pointer to user it.
---------------	---------------------

#### Returns

true There was an last uid to retrive.  
false No last uid was retrived.

### 7.36.2.8 rfid\_set\_last\_uid()

```
void rfid_set_last_uid (
    const byte * uidIn)
```

#### Parameters

<i>uidOut</i>	Pointer to user it.
---------------	---------------------

#### Returns

true Stored last used uid.  
false Last uid was not stored

### 7.36.2.9 setup\_RFID\_reader()

```
void setup_RFID_reader (
    MFRC522 & rfid)
```

#### Parameters

<i>rfid</i>	An instance of a MFRC522 object
-------------	---------------------------------

### 7.36.2.10 validate\_rfid()

```
bool validate_rfid (
    MFRC522 myRFID)
```

#### Parameters

<i>myRFID</i>	An instance of a MFRC522 object
---------------	---------------------------------

#### Returns

true The RFID was a match.  
false There was no matching RFID in the database.

### 7.36.3 Variable Documentation

#### 7.36.3.1 hasUID

```
bool hasUID = false [static]
```

#### 7.36.3.2 lastUID

```
byte lastUID[UID\_LENGTH] [static]
```

#### 7.36.3.3 userCount

```
int userCount = 0
```

#### 7.36.3.4 users

```
User users[MAX\_ROOMS]
```

## 7.37 src/sale\_html.cpp File Reference

File for setting up the graph section of the web server.

```
#include "index_html.h"  
#include "sale_html.h"
```

### Functions

- void [send\\_sale\\_html\\_graph](#) (ESP8266WebServer &[server](#), uint8\_t room\_number, const char \*bar\_type, int bar\_height)  
*Sends a single sales bar element to the client.*
- void [send\\_sale\\_html\\_page](#) (ESP8266WebServer &[server](#), uint8\_t room\_count, const int \*greenHeight, const int \*classicHeights)  
*Sends the complete sales graph page.*

### Variables

- const char SALE\_BOX\_START[] [PROGMEM](#) = R"rawliteral( <div class="sale\_box">)rawliteral"  
*Opening container for the sales graph.*

### 7.37.1 Detailed Description

#### Author

Baldur G. Toftegaard

## 7.37.2 Function Documentation

### 7.37.2.1 send\_sale\_html\_graph()

```
void send_sale_html_graph (
    ESP8266WebServer & server,
    uint8_t room_number,
    const char * bar_type,
    int bar_height)
```

#### Parameters

<i>server</i>	The server instance.
---------------	----------------------

<i>room_number</i>	The room number operated on.
<i>bar_type</i>	Type of the bar operated on.
<i>bar_height</i>	Height of the bar operated on.

### 7.37.2.2 send\_sale\_html\_page()

```
void send_sale_html_page (
    ESP8266WebServer & server,
    uint8_t room_count,
    const int * greenHeight,
    const int * classicHeights)
```

#### Parameters

<i>server</i>	The server instance.
<i>room_count</i>	Number of registered rooms.
<i>greenHeight</i>	The height of green bars.
<i>classicHeights</i>	The height of classic bars.

## 7.37.3 Variable Documentation

### 7.37.3.1 PROGMEM

const char SALE\_BOX\_STOP [ ] PROGMEM = R"rawliteral( <div class="sale\_box">)rawliteral"

Closing container for the complete sales graph.

Closing container for a single room graph.

Closing fragment for a single sales bar.

HTML fragment defining the height style of a bar.

HTML fragment defining the CSS class type for a bar.

HTML fragment for the room identifier.

Opening container for a single room graph.

Opening container for the sales graph.

Interpreted by compiler as a string.

Opening container for the sales graph.

String used for the HTML footer.

This is responsible for updating the graphs. read by the compiler as a string

Opening container for the sales graph.

Interpreted as a string by the compiler.

Opening container for the sales graph.

is handled like a string by the compiler

## 7.38 src/weight\_scale.cpp File Reference

File used for setting up the scale.

```
#include "weight_scale.h"
```

### Functions

- HX711\_ADC `scale` (HX711\_DOUT, HX711\_SCK)
   
*Function to get the weight reference.*
- float `get_weight_reference` (void)
   
*Function to set the weight reference.*
- void `set_weight_reference` (float value)
   
*Function to reset the weight reference.*
- void `reset_weight_reference` (void)
   
*Function to send confirmation that the weight reference is set.*
- bool `weight_reference_is_set` (void)
   
*Function to setup the scale, is called in the begining of the program.*
- bool `update_scale` ()
   
*Function for updating the scale value.*
- float `get_weight` ()
   
*Function to get the scale reading.*
- void `tare_scale` ()
   
*Function to tare the scale.*
- bool `tare_complete` ()
   
*Function to signal that the scale has been tared.*
- int `get_beer_cans_taken` (float referenceWeight, float currentWeight)
   
*Function to get the number of beer cans taken.*

### Variables

- static float `g_referenceWeight` = NAN

### 7.38.1 Detailed Description

#### Author

Amal Araweelo Almis

### 7.38.2 Function Documentation

#### 7.38.2.1 `get_beer_cans_taken()`

```
int get_beer_cans_taken (
    float referencWeight,
    float currentWeight)
```

#### Parameters

<code>referencWeight</code>	The set reference weight [g]
-----------------------------	------------------------------

<i>currentWeight</i>	The current measured weight [g]
----------------------	------------------------------------

**Returns**

```
int number of cans taken
```

**7.38.2.2 get\_weight()**

```
float get_weight (
    void )
```

**Returns**

```
float Value read from weight.
```

**7.38.2.3 get\_weight\_reference()**

```
float get_weight_reference (
    void )
```

**Returns**

```
float The weight reference.
```

**7.38.2.4 reset\_weight\_reference()**

```
void reset_weight_reference (
    void )
```

**7.38.2.5 scale()**

```
HX711_ADC scale (
    HX711_DOUT ,
    HX711_SCK )
```

**7.38.2.6 set\_weight\_reference()**

```
void set_weight_reference (
    float value)
```

**Parameters**

<i>value</i>	Reference value.
--------------	------------------

### 7.38.2.7 setup\_scale()

```
void setup_scale (
    float calFactor)
```

#### Parameters

<i>calFactor</i>	The calibration factor.
------------------	-------------------------

### 7.38.2.8 tare\_complete()

```
bool tare_complete (
    void )
```

#### Returns

- true The scale was tared.
- false The scale was not tared.

### 7.38.2.9 tare\_scale()

```
void tare_scale (
    void )
```

### 7.38.2.10 update\_scale()

```
bool update_scale (
    void )
```

#### Returns

- true Scale was updated.
- false Scale was not updated.

### 7.38.2.11 weight\_reference\_is\_set()

```
bool weight_reference_is_set (
    void )
```

#### Returns

- true Weight reference is set.
- false Weight reference is not set.

## 7.38.3 Variable Documentation

### 7.38.3.1 g\_referenceWeight

```
float g_referenceWeight = NAN [static]
```



# Index

activeCommand  
    main.cpp, 67  
add\_user  
    rfid\_access.cpp, 69  
    rfid\_access.h, 36  
admin\_html.h  
    PROGMEM, 15  
  
balance  
    User, 14  
beer  
    inventory.h, 25  
BEER\_WEIGHT  
    weight\_scale.h, 48  
beverage  
    products\_stocked, 13  
beverage\_type  
    inventory.h, 25  
beverage\_variant  
    product, 12  
BFIT, 1  
boxClosed  
    lock\_ctrl.cpp, 64  
BUZZER  
    lock\_ctrl.cpp, 64  
buzzer.cpp  
    BUZZERPIN, 53  
    HIGH\_TONE, 53  
    LOW\_TONE, 53  
    play\_lock, 52  
    play\_unlock, 52  
    play\_warning, 53  
    TONE\_LENGTH, 53  
buzzer.h  
    BUZZER\_H, 17  
    play\_lock, 17  
    play\_unlock, 17  
    play\_warning, 17  
BUZZER\_H  
    buzzer.h, 17  
BUZZERPIN  
    buzzer.cpp, 53  
  
CAL\_FACTOR  
    main.cpp, 67  
check\_command  
    rfid\_access.cpp, 69  
    rfid\_access.h, 36  
cider  
    inventory.h, 26  
  
classicHeight  
    graph\_data.cpp, 58  
    graph\_data.h, 20  
    main.cpp, 67  
CLOSED\_THRESHOLD  
    lock\_ctrl.h, 30  
CMD\_ADD\_USER  
    rfid\_access.h, 36  
CMD\_CONFIRM  
    rfid\_access.h, 36  
CMD\_LOCK  
    rfid\_access.h, 36  
CMD\_NONE  
    rfid\_access.h, 35  
CMD\_OPEN  
    rfid\_access.h, 36  
CMD\_PRINT  
    rfid\_access.h, 36  
CMD\_REMOVE\_USER  
    rfid\_access.h, 36  
compare\_UID  
    rfid\_access.cpp, 70  
    rfid\_access.h, 36  
connect\_wifi\_and\_start\_mdns  
    main.cpp, 66  
count\_rooms  
    database\_management.cpp, 54  
    rfid\_access.h, 36  
current\_quantity  
    products\_stocked, 13  
  
database\_management.cpp  
    count\_rooms, 54  
    find\_empty\_index, 54  
    get\_users\_db, 55  
    print\_all\_users, 55  
    print\_single\_user, 55  
    print\_uid, 55  
    read\_confirmation, 56  
    read\_integer, 56  
    remove\_user, 56  
    user\_management, 56  
demo\_beer  
    main.cpp, 67  
display\_commands  
    rfid\_access.cpp, 70  
    rfid\_access.h, 37  
display\_commands\_um  
    rfid\_access.cpp, 70  
    rfid\_access.h, 37

doorCloseTimer  
     main.cpp, 67  
 doorUnlocked  
     main.cpp, 68  
  
 find\_empty\_index  
     database\_management.cpp, 54  
     rfid\_access.h, 37  
 fridge  
     fridge\_state.cpp, 57  
     fridge\_state.h, 18  
 fridge\_state.cpp  
     fridge, 57  
 fridge\_state.h  
     fridge, 18  
  
 g\_referenceWeight  
     weight\_scale.cpp, 77  
 get\_beer\_cans\_taken  
     weight\_scale.cpp, 75  
     weight\_scale.h, 49  
 get\_users\_db  
     database\_management.cpp, 55  
     rfid\_access.h, 37  
 get\_weight  
     weight\_scale.cpp, 76  
     weight\_scale.h, 49  
 get\_weight\_reference  
     weight\_scale.cpp, 76  
     weight\_scale.h, 49  
 graph\_add\_to\_room\_clasic  
     graph\_data.cpp, 58  
     graph\_data.h, 19  
 graph\_add\_to\_room\_green  
     graph\_data.cpp, 58  
     graph\_data.h, 19  
 graph\_data.cpp  
     classicHeight, 58  
     graph\_add\_to\_room\_clasic, 58  
     graph\_add\_to\_room\_green, 58  
     greenHeight, 58  
 graph\_data.h  
     classicHeight, 20  
     graph\_add\_to\_room\_clasic, 19  
     graph\_add\_to\_room\_green, 19  
     greenHeight, 20  
     print\_graph\_arrays, 19  
     ROOM\_COUNT, 19  
 greenHeight  
     graph\_data.cpp, 58  
     graph\_data.h, 20  
     main.cpp, 68  
  
 hasUID  
     rfid\_access.cpp, 72  
 HIGH\_TONE  
     buzzer.cpp, 53  
     lock\_ctrl.cpp, 64  
 HX711\_DOUT  
     weight\_scale.h, 48  
 HX711\_SCK  
     weight\_scale.h, 48  
  
 include Directory Reference, 9  
 include/admin\_html.h, 15, 16  
 include/buzzer.h, 16, 17  
 include/fridge\_state.h, 17, 18  
 include/graph\_data.h, 18, 20  
 include/index\_html.h, 20, 21  
 include/init\_users\_and\_sale.h, 22, 23  
 include/inventory.h, 23, 28  
 include/lock\_ctrl.h, 29, 32  
 include/login\_html.h, 32, 33  
 include/rfid\_access.h, 33, 42  
 include/sale\_html.h, 43, 45  
 include/style\_css.h, 46  
 include/weight\_scale.h, 47, 51  
 index\_html.h  
     PROGMEM, 21  
 init\_users\_and\_products  
     init\_users\_and\_sale.cpp, 59  
     init\_users\_and\_sale.h, 23  
 init\_users\_and\_sale.cpp  
     init\_users\_and\_products, 59  
     perform\_sale, 59  
     read\_current\_weight\_blocking, 59  
 init\_users\_and\_sale.h  
     init\_users\_and\_products, 23  
     number\_of\_users, 22  
     perform\_sale, 23  
 inventory, 11  
     number\_of\_products\_stocked, 11  
     products\_in\_inventory, 11  
     room\_number, 11  
 inventory.cpp  
     inventory\_add\_beverage, 60  
     inventory\_add\_product, 60  
     inventory\_init, 61  
     inventory\_make\_product, 61  
     inventory\_print, 61  
     inventory\_remove\_beverage, 61  
     inventory\_remove\_product, 62  
 inventory.h  
     beer, 25  
     beverage\_type, 25  
     cider, 26  
     inventory\_add\_beverage, 26  
     inventory\_add\_product, 26  
     INVENTORY\_CAPACITY, 24  
     inventory\_init, 26  
     inventory\_make\_product, 27  
     inventory\_print, 27  
     inventory\_remove\_beverage, 27  
     inventory\_remove\_product, 28  
     limfjords\_porter, 26  
     other, 26  
     soda, 26  
     inventory\_add\_beverage

inventory.cpp, 60  
inventory.h, 26  
inventory\_add\_product  
    inventory.cpp, 60  
    inventory.h, 26  
INVENTORY\_CAPACITY  
    inventory.h, 24  
inventory\_init  
    inventory.cpp, 61  
    inventory.h, 26  
inventory\_make\_product  
    inventory.cpp, 61  
    inventory.h, 27  
inventory\_print  
    inventory.cpp, 61  
    inventory.h, 27  
inventory\_remove\_beverage  
    inventory.cpp, 61  
    inventory.h, 27  
inventory\_remove\_product  
    inventory.cpp, 62  
    inventory.h, 28  
is\_box\_closed  
    lock\_ctrl.cpp, 63  
    lock\_ctrl.h, 31  
  
lastUID  
    rfid\_access.cpp, 72  
LIGHT\_PIN  
    lock\_ctrl.h, 30  
limfjords\_porter  
    inventory.h, 26  
lock\_ctrl.cpp  
    boxClosed, 64  
    BUZZER, 64  
    HIGH\_TONE, 64  
    is\_box\_closed, 63  
    lock\_ctrl\_init, 63  
    lock\_door, 64  
    LOCK\_POS, 64  
    lockServo, 64  
    LOW\_TONE, 65  
    play\_close, 64  
    play\_open, 64  
    TONE\_LENGTH, 65  
    unlock\_door, 64  
    UNLOCK\_POS, 65  
lock\_ctrl.h  
    CLOSED\_THRESHOLD, 30  
    is\_box\_closed, 31  
    LIGHT\_PIN, 30  
    lock\_ctrl\_init, 31  
    lock\_door, 31  
    OPEN\_THRESHOLD, 30  
    play\_close, 31  
    play\_open, 31  
    SERVO\_PIN, 30  
    unlock\_door, 31  
lock\_ctrl\_init  
    lock\_ctrl.cpp, 63  
    lock\_ctrl.h, 31  
lock\_door  
    lock\_ctrl.cpp, 64  
    lock\_ctrl.h, 31  
LOCK\_POS  
    lock\_ctrl.cpp, 64  
lockServo  
    lock\_ctrl.cpp, 64  
login\_html.h  
    PROGMEM, 32  
loop  
    main.cpp, 66  
LOW\_TONE  
    buzzer.cpp, 53  
    lock\_ctrl.cpp, 65  
  
main.cpp  
    activeCommand, 67  
    CAL\_FACTOR, 67  
    classicHeight, 67  
    connect\_wifi\_and\_start\_mdns, 66  
    demo\_beer, 67  
    doorCloseTimer, 67  
    doorUnlocked, 68  
    greenHeight, 68  
    loop, 66  
    rfid, 66  
    server, 66  
    setup, 66  
    setup\_inventory\_and\_scale, 67  
    setup\_rfid\_and\_lock, 67  
    setup\_web\_routes, 67  
    START\_BEER\_QTY, 68  
    WIFI\_PASS, 68  
    WIFI\_SSID, 68  
MAX\_ROOMS  
    rfid\_access.h, 35  
  
name  
    product, 12  
number\_of\_products\_stocked  
    inventory, 11  
number\_of\_users  
    init\_users\_and\_sale.h, 22  
  
OPEN\_THRESHOLD  
    lock\_ctrl.h, 30  
original\_quantity  
    products\_stocked, 13  
other  
    inventory.h, 26  
  
perform\_sale  
    init\_users\_and\_sale.cpp, 59  
    init\_users\_and\_sale.h, 23  
play\_close  
    lock\_ctrl.cpp, 64  
    lock\_ctrl.h, 31

play\_lock  
  buzzer.cpp, 52  
  buzzer.h, 17  
play\_open  
  lock\_ctrl.cpp, 64  
  lock\_ctrl.h, 31  
play\_unlock  
  buzzer.cpp, 52  
  buzzer.h, 17  
play\_warning  
  buzzer.cpp, 53  
  buzzer.h, 17  
price  
  product, 12  
print\_all\_users  
  database\_management.cpp, 55  
  rfid\_access.h, 37  
print\_graph\_arrays  
  graph\_data.h, 19  
print\_single\_user  
  database\_management.cpp, 55  
  rfid\_access.h, 38  
print\_uid  
  database\_management.cpp, 55  
  rfid\_access.h, 38  
product, 12  
  beverage\_variant, 12  
  name, 12  
  price, 12  
  weight, 12  
products\_in\_inventory  
  inventory, 11  
products\_stocked, 13  
  beverage, 13  
  current\_quantity, 13  
  original\_quantity, 13  
PROGMEM  
  admin\_html.h, 15  
  index\_html.h, 21  
  login\_html.h, 32  
  sale\_html.cpp, 74  
  sale\_html.h, 44  
  style\_css.h, 46  
read\_confirmation  
  database\_management.cpp, 56  
  rfid\_access.h, 38  
read\_current\_weight\_blocking  
  init\_users\_and\_sale.cpp, 59  
read\_integer  
  database\_management.cpp, 56  
  rfid\_access.h, 38  
read\_RFID\_tag  
  rfid\_access.cpp, 70  
  rfid\_access.h, 38  
README.md, 52  
remove\_user  
  database\_management.cpp, 56  
  rfid\_access.h, 40  
reset\_weight\_reference  
  weight\_scale.cpp, 76  
  weight\_scale.h, 49  
rfid  
  main.cpp, 66  
rfid\_access.cpp  
  add\_user, 69  
  check\_command, 69  
  compare\_UID, 70  
  display\_commands, 70  
  display\_commands\_um, 70  
  hasUID, 72  
  lastUID, 72  
  read\_RFID\_tag, 70  
  rfid\_get\_last\_uid, 70  
  rfid\_set\_last\_uid, 71  
  setup\_RFID\_reader, 71  
  userCount, 72  
  users, 72  
  validate\_rfid, 71  
rfid\_access.h  
  add\_user, 36  
  check\_command, 36  
  CMD\_ADD\_USER, 36  
  CMD\_CONFIRM, 36  
  CMD\_LOCK, 36  
  CMD\_NONE, 35  
  CMD\_OPEN, 36  
  CMD\_PRINT, 36  
  CMD\_REMOVE\_USER, 36  
  compare\_UID, 36  
  count\_rooms, 36  
  display\_commands, 37  
  display\_commands\_um, 37  
  find\_empty\_index, 37  
  get\_users\_db, 37  
  MAX\_ROOMS, 35  
  print\_all\_users, 37  
  print\_single\_user, 38  
  print\_uid, 38  
  read\_confirmation, 38  
  read\_integer, 38  
  read\_RFID\_tag, 38  
  remove\_user, 40  
  rfid\_get\_last\_uid, 40  
  rfid\_set\_last\_uid, 40  
  RFIDCommand, 35  
  RST\_PIN, 35  
  setup\_RFID\_reader, 40  
  SS\_PIN, 35  
  UID\_LENGTH, 35  
  user\_management, 41  
  userCount, 41  
  users, 41  
  validate\_rfid, 41  
rfid\_get\_last\_uid  
  rfid\_access.cpp, 70  
  rfid\_access.h, 40

rfid\_set\_last\_uid  
    rfid\_access.cpp, 71  
    rfid\_access.h, 40  
RFIDCommand  
    rfid\_access.h, 35  
ROOM\_COUNT  
    graph\_data.h, 19  
room\_number  
    inventory, 11  
roomNumber  
    User, 14  
RST\_PIN  
    rfid\_access.h, 35  
  
sale\_html.cpp  
    PROGMEM, 74  
    send\_sale\_html\_graph, 73  
    send\_sale\_html\_page, 74  
sale\_html.h  
    PROGMEM, 44  
    send\_sale\_html\_graph, 44  
    send\_sale\_html\_page, 44  
scale  
    weight\_scale.cpp, 76  
    weight\_scale.h, 51  
SCALE\_DEFAULT\_SETTLE\_TIME\_MS  
    weight\_scale.h, 49  
SCALE\_TOL  
    weight\_scale.h, 49  
send\_sale\_html\_graph  
    sale\_html.cpp, 73  
    sale\_html.h, 44  
send\_sale\_html\_page  
    sale\_html.cpp, 74  
    sale\_html.h, 44  
server  
    main.cpp, 66  
SERVO\_PIN  
    lock\_ctrl.h, 30  
set\_weight\_reference  
    weight\_scale.cpp, 76  
    weight\_scale.h, 50  
setup  
    main.cpp, 66  
setup\_inventory\_and\_scale  
    main.cpp, 67  
setup\_rfid\_and\_lock  
    main.cpp, 67  
setup\_RFID\_reader  
    rfid\_access.cpp, 71  
    rfid\_access.h, 40  
setup\_scale  
    weight\_scale.cpp, 76  
    weight\_scale.h, 50  
setup\_web\_routes  
    main.cpp, 67  
soda  
    inventory.h, 26  
src Directory Reference, 10  
src/buzzer.cpp, 52  
src/database\_management.cpp, 53  
src/fridge\_state.cpp, 57  
src/graph\_data.cpp, 57  
src/init\_users\_and\_sale.cpp, 59  
src/inventory.cpp, 60  
src/lock\_ctrl.cpp, 62  
src/main.cpp, 65  
src/rfid\_access.cpp, 68  
src/sale\_html.cpp, 72  
src/weight\_scale.cpp, 75  
SS\_PIN  
    rfid\_access.h, 35  
START\_BEER\_QTY  
    main.cpp, 68  
style\_css.h  
    PROGMEM, 46  
  
tare\_complete  
    weight\_scale.cpp, 77  
    weight\_scale.h, 50  
tare\_scale  
    weight\_scale.cpp, 77  
    weight\_scale.h, 50  
TONE\_LENGTH  
    buzzer.cpp, 53  
    lock\_ctrl.cpp, 65  
  
uid  
    User, 14  
UID\_LENGTH  
    rfid\_access.h, 35  
unlock\_door  
    lock\_ctrl.cpp, 64  
    lock\_ctrl.h, 31  
UNLOCK\_POS  
    lock\_ctrl.cpp, 65  
update\_scale  
    weight\_scale.cpp, 77  
    weight\_scale.h, 50  
User, 13  
    balance, 14  
    roomNumber, 14  
    uid, 14  
user\_management  
    database\_management.cpp, 56  
    rfid\_access.h, 41  
userCount  
    rfid\_access.cpp, 72  
    rfid\_access.h, 41  
users  
    rfid\_access.cpp, 72  
    rfid\_access.h, 41  
validate\_rfid  
    rfid\_access.cpp, 71  
    rfid\_access.h, 41  
weight

product, 12  
weight\_reference\_is\_set  
  weight\_scale.cpp, 77  
  weight\_scale.h, 50  
weight\_scale.cpp  
  g\_referenceWeight, 77  
  get\_beer\_cans\_taken, 75  
  get\_weight, 76  
  get\_weight\_reference, 76  
  reset\_weight\_reference, 76  
  scale, 76  
  set\_weight\_reference, 76  
  setup\_scale, 76  
  tare\_complete, 77  
  tare\_scale, 77  
  update\_scale, 77  
  weight\_reference\_is\_set, 77  
weight\_scale.h  
  BEER\_WEIGHT, 48  
  get\_beer\_cans\_taken, 49  
  get\_weight, 49  
  get\_weight\_reference, 49  
  HX711\_DOUT, 48  
  HX711\_SCK, 48  
  reset\_weight\_reference, 49  
  scale, 51  
  SCALE\_DEFAULT\_SETTLE\_TIME\_MS, 49  
  SCALE\_TOL, 49  
  set\_weight\_reference, 50  
  setup\_scale, 50  
  tare\_complete, 50  
  tare\_scale, 50  
  update\_scale, 50  
  weight\_reference\_is\_set, 50  
WIFI\_PASS  
  main.cpp, 68  
WIFI\_SSID  
  main.cpp, 68