BFIT

Generated by Doxygen 1.16.1

1 BFIT 1

2 Directory Hierarchy 3

2.1 Directories . . . 3

3 Class Index 5

3.1 Class List . . . 5

4 File Index 7

4.1 File List . . . 7

5 Directory Documentation 9

5.1 include Directory Reference . . . 9

5.2 src Directory Reference . . . 9

6 Class Documentation 11

6.1 inventory Struct Reference . . . 11

6.1.1 Detailed Description . . . 11

6.1.2 Member Data Documentation . . . 11

6.1.2.1 number_of_products_stocked . . . 11

6.1.2.2 produckts_in_inventory . . . 11

6.1.2.3 room_number . . . 12

6.2 product Struct Reference . . . 12

6.2.1 Detailed Description . . . 12

6.2.2 Member Data Documentation . . . 12

6.2.2.1 beverage_variant . . . 12

6.2.2.2 name . . . 12

6.2.2.3 price . . . 13

6.2.2.4 weight . . . 13

6.3 products_stocked Struct Reference . . . 13

6.3.1 Detailed Description . . . 13

6.3.2 Member Data Documentation . . . 13

6.3.2.1 beverage . . . 13

6.3.2.2 current_quantity . . . 14

6.3.2.3 original_quantity . . . 14

6.4 User Struct Reference . . . 14

6.4.1 Detailed Description . . . 14

6.4.2 Member Data Documentation . . . 14

6.4.2.1 balance . . . 14

6.4.2.2 roomNumber . . . 14

6.4.2.3 uid . . . 14

7 File Documentation 15

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 Macro Definition Documentation . . . 17
7.3.1.1 BUZZER_H . . . 17
7.3.2 Function Documentation . . . 17
7.3.2.1 play_lock() . . . 17
7.3.2.2 play_unlock() . . . 17
7.3.2.3 play_warning() . . . 17
7.4 buzzer.h . . . 17
7.5 include/fridge_state.h File Reference . . . 18
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 Macro Definition Documentation . . . 19
7.7.1.1 ROOM_COUNT . . . 19
7.7.2 Function Documentation . . . 19
7.7.2.1 graph_add_to_room_clasic() . . . 19
7.7.2.2 graph_add_to_room_green() . . . 19
7.7.2.3 print_graph_arrays() . . . 20
7.7.3 Variable Documentation . . . 20
7.7.3.1 classicHeight . . . 20
7.7.3.2 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 . . . 22
7.11 include/init_users_and_sale.h File Reference . . . 22
7.11.1 Detailed Description . . . 23
7.11.2 Macro Definition Documentation . . . 23
7.11.2.1 number_of_users . . . 23
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 . . . 24
7.13 include/inventory.h File Reference . . . 24

7.13.1 Detailed Description . . . 25
7.13.2 Macro Definition Documentation . . . 25
7.13.2.1 INVENTORY_CAPACITY . . . 25
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() . . . 28
7.13.4.6 inventory_remove_beverage() . . . 28
7.13.4.7 inventory_remove_product() . . . 28
7.14 inventory.h . . . 29
7.15 include/lock_ctrl.h File Reference . . . 30
7.15.1 Detailed Description . . . 30
7.15.2 Macro Definition Documentation . . . 31
7.15.2.1 CLOSED_THRESHOLD . . . 31
7.15.2.2 LIGHT_PIN . . . 31
7.15.2.3 OPEN_THRESHOLD . . . 31
7.15.2.4 SERVO_PIN . . . 31
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 play_warning() . . . 32
7.15.3.7 unlock_door() . . . 32
7.16 lock_ctrl.h . . . 32
7.17 include/login_html.h File Reference . . . 32
7.17.1 Detailed Description . . . 33
7.17.2 Variable Documentation . . . 33
7.17.2.1 PROGMEM . . . 33
7.18 login_html.h . . . 33
7.19 include/rfid_access.h File Reference . . . 34
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 . . . 36

7.19.3 Enumeration Type Documentation . . . 36
7.19.3.1 RFIDcommand . . . 36
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() . . . 37
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() . . . 38
7.19.4.8 get_users_db() . . . 38
7.19.4.9 print_all_users() . . . 38
7.19.4.10 print_single_user() . . . 38
7.19.4.11 print_uid() . . . 39
7.19.4.12 read_confirmation() . . . 39
7.19.4.13 read_integer() . . . 39
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 . . . 42
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 . . . 45
7.21.3.1 PROGMEM . . . 45
7.22 sale_html.h . . . 46
7.23 include/style_css.h File Reference . . . 46
7.23.1 Detailed Description . . . 46
7.23.2 Variable Documentation . . . 47
7.23.2.1 PROGMEM . . . 47
7.24 style_css.h . . . 47
7.25 include/weight_scale.h File Reference . . . 48
7.25.1 Detailed Description . . . 49

7.25.2 Macro Definition Documentation . . . 49
7.25.2.1 BEER_WEIGHT . . . 49
7.25.2.2 HX711_DOUT . . . 49
7.25.2.3 HX711_SCK . . . 49
7.25.2.4 SCALE_DEFAULT_SETTLE_TIME_MS . . . 49
7.25.2.5 SCALE_TOL . . . 50
7.25.3 Enumeration Type Documentation . . . 50
7.25.3.1 weight_recall_action . . . 50
7.25.4 Function Documentation . . . 51
7.25.4.1 get_beer_cans_taken() . . . 51
7.25.4.2 get_weight() . . . 51
7.25.4.3 get_weight_reference() . . . 51
7.25.4.4 reset_weight_reference() . . . 52
7.25.4.5 set_weight_reference() . . . 52
7.25.4.6 setup_scale() . . . 52
7.25.4.7 tare_complete() . . . 52
7.25.4.8 tare_scale() . . . 52
7.25.4.9 update_scale() . . . 53
7.25.4.10 weight_reference_is_set() . . . 53
7.25.5 Variable Documentation . . . 53
7.25.5.1 scale . . . 53
7.26 weight_scale.h . . . 53
7.27 README.md File Reference . . . 54
7.28 src/buzzer.cpp File Reference . . . 54
7.28.1 Function Documentation . . . 55
7.28.1.1 play_lock() . . . 55
7.28.1.2 play_unlock() . . . 55
7.28.1.3 play_warning() . . . 55
7.28.2 Variable Documentation . . . 55
7.28.2.1 BUZZERPIN . . . 55
7.28.2.2 HIGH_TONE . . . 55
7.28.2.3 LOW_TONE . . . 55
7.28.2.4 TONE_LENGTH . . . 55
7.29 src/database_management.cpp File Reference . . . 56
7.29.1 Detailed Description . . . 56
7.29.2 Function Documentation . . . 57
7.29.2.1 count_rooms() . . . 57
7.29.2.2 find_empty_index() . . . 57
7.29.2.3 get_users_db() . . . 57
7.29.2.4 print_all_users() . . . 58
7.29.2.5 print_single_user() . . . 58
7.29.2.6 print_uid() . . . 58

7.29.2.7 read_confirmation() . . . 58
7.29.2.8 read_integer() . . . 59
7.29.2.9 remove_user() . . . 59
7.29.2.10 user_management() . . . 59
7.30 src/fridge_state.cpp File Reference . . . 59
7.30.1 Detailed Description . . . 60
7.30.2 Variable Documentation . . . 60
7.30.2.1 fridge . . . 60
7.31 src/graph_data.cpp File Reference . . . 60
7.31.1 Function Documentation . . . 60
7.31.1.1 graph_add_to_room_clasic() . . . 60
7.31.1.2 graph_add_to_room_green() . . . 61
7.31.2 Variable Documentation . . . 61
7.31.2.1 classicHeight . . . 61
7.31.2.2 greenHeight . . . 61
7.32 src/init_users_and_sale.cpp File Reference . . . 61
7.32.1 Function Documentation . . . 62
7.32.1.1 init_users_and_products() . . . 62
7.32.1.2 perform_sale() . . . 62
7.32.1.3 read_current_weight_blocking() . . . 62
7.33 src/inventory.cpp File Reference . . . 62
7.33.1 Detailed Description . . . 63
7.33.2 Function Documentation . . . 63
7.33.2.1 inventory_add_beverage() . . . 63
7.33.2.2 inventory_add_product() . . . 63
7.33.2.3 inventory_init() . . . 64
7.33.2.4 inventory_make_product() . . . 64
7.33.2.5 inventory_print() . . . 65
7.33.2.6 inventory_remove_beverage() . . . 65
7.33.2.7 inventory_remove_product() . . . 66
7.34 src/lock_ctrl.cpp File Reference . . . 66
7.34.1 Detailed Description . . . 67
7.34.2 Function Documentation . . . 67
7.34.2.1 is_box_closed() . . . 67
7.34.2.2 lock_ctrl_init() . . . 67
7.34.2.3 lock_door() . . . 67
7.34.2.4 play_close() . . . 67
7.34.2.5 play_open() . . . 67
7.34.2.6 unlock_door() . . . 67
7.34.3 Variable Documentation . . . 67
7.34.3.1 boxClosed . . . 67
7.34.3.2 BUZZER . . . 68

7.34.3.3 HIGH_TONE . . . 68
7.34.3.4 LOCK_POS . . . 68
7.34.3.5 lockServo . . . 68
7.34.3.6 LOW_TONE . . . 68
7.34.3.7 TONE_LENGTH . . . 68
7.34.3.8 UNLOCK_POS . . . 68
7.35 src/main.cpp File Reference . . . 68
7.35.1 Detailed Description . . . 69
7.35.2 Function Documentation . . . 69
7.35.2.1 connect_wifi_and_start_mdns() . . . 69
7.35.2.2 loop() . . . 69
7.35.2.3 print_graph_arrays() . . . 70
7.35.2.4 rfid() . . . 70
7.35.2.5 server() . . . 70
7.35.2.6 setup() . . . 70
7.35.2.7 setup_inventory_and_scale() . . . 70
7.35.2.8 setup_rfid_and_lock() . . . 70
7.35.2.9 setup_web_routes() . . . 70
7.35.3 Variable Documentation . . . 70
7.35.3.1 activeCommand . . . 70
7.35.3.2 CAL_FACTOR . . . 70
7.35.3.3 classicHeight . . . 71
7.35.3.4 demo_beer . . . 71
7.35.3.5 doorCloseTimer . . . 71
7.35.3.6 doorUnlocked . . . 71
7.35.3.7 greenHeight . . . 71
7.35.3.8 START_BEER_QTY . . . 71
7.35.3.9 WIFI_PASS . . . 71
7.35.3.10 WIFI_SSID . . . 71
7.36 src/rfid_access.cpp File Reference . . . 71
7.36.1 Detailed Description . . . 72
7.36.2 Function Documentation . . . 72
7.36.2.1 add_user() . . . 72
7.36.2.2 check_command() . . . 73
7.36.2.3 compare_UID() . . . 73
7.36.2.4 display_commands() . . . 73
7.36.2.5 display_commands_um() . . . 73
7.36.2.6 read_RFID_tag() . . . 73
7.36.2.7 rfid_get_last_uid() . . . 74
7.36.2.8 rfid_set_last_uid() . . . 74
7.36.2.9 setup_RFID_reader() . . . 74
7.36.2.10 validate_rfid() . . . 75

7.36.3 Variable Documentation . . . 75
7.36.3.1 hasUID . . . 75
7.36.3.2 lastUID . . . 75
7.36.3.3 userCount . . . 75
7.36.3.4 users . . . 75
7.37 src/sale_html.cpp File Reference . . . 75
7.37.1 Detailed Description . . . 76
7.37.2 Function Documentation . . . 76
7.37.2.1 send_sale_html_graph() . . . 76
7.37.2.2 send_sale_html_page() . . . 77
7.37.3 Variable Documentation . . . 77
7.37.3.1 PROGMEM . . . 77
7.38 src/weight_scale.cpp File Reference . . . 78
7.38.1 Detailed Description . . . 78
7.38.2 Function Documentation . . . 78
7.38.2.1 get_beer_cans_taken() . . . 78
7.38.2.2 get_weight() . . . 79
7.38.2.3 get_weight_reference() . . . 79
7.38.2.4 reset_weight_reference() . . . 79
7.38.2.5 scale() . . . 79
7.38.2.6 set_weight_reference() . . . 80
7.38.2.7 setup_scale() . . . 80
7.38.2.8 tare_complete() . . . 80
7.38.2.9 tare_scale() . . . 80
7.38.2.10 update_scale() . . . 80
7.38.2.11 weight_reference_is_set() . . . 81
7.38.3 Variable Documentation . . . 81
7.38.3.1 g_referenceWeight . . . 81

Index **83**

Chapter 1

BFIT

Beer Fridge Inventory Tracking (who takes, how much do they take, current drink information, statistics and so on and so forth.)

Chapter 2

Directory Hierarchy

2.1 Directories

include . . . 9
admin_html.h . . . 15
buzzer.h . . . 16
fridge_state.h . . . 18
graph_data.h . . . 18
index_html.h . . . 20
init_users_and_sale.h . . . 22
inventory.h . . . 24
lock_ctrl.h . . . 30
login_html.h . . . 32
rfid_access.h . . . 34
sale_html.h . . . 43
style_css.h . . . 46
weight_scale.h . . . 48
src . . . 9
buzzer.cpp . . . 54
database_management.cpp . . . 56
fridge_state.cpp . . . 59
graph_data.cpp . . . 60
init_users_and_sale.cpp . . . 61
inventory.cpp . . . 62
lock_ctrl.cpp . . . 66
main.cpp . . . 68
rfid_access.cpp . . . 71
sale_html.cpp . . . 75
weight_scale.cpp . . . 78

Chapter 3

Class Index

3.1 Class List

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

inventory
: Strukt used for making the inventory 11

product
: Struct holding the information for one item 12

products_stocked
: Struckt for keeping track of the original and current quantity of a beverage 13

User
: User record stored in the RFID user database 14

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 . . . 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 . . . 18
include/graph_data.h . . . 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 . . . 24
include/lock_ctrl.h . . . 30
include/login_html.h
HTML file for the login page . . . 32
include/rfid_access.h . . . 34
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 . . . 48
src/buzzer.cpp . . . 54
src/database_management.cpp
Used to read and write non-volatile memory on ESP8266 . . . 56
src/fridge_state.cpp . . . 59
src/graph_data.cpp . . . 60
src/init_users_and_sale.cpp . . . 61
src/inventory.cpp
Functions responsible for keeping track of the fridge inventory . . . 62
src/lock_ctrl.cpp . . . 66
src/main.cpp
Combined: Web server + Graph + Scale + RFID access + Lock control . . . 68
src/rfid_access.cpp . . . 71
src/sale_html.cpp . . . 75
src/weight_scale.cpp . . . 78

Chapter 5

Directory Documentation

5.1 include Directory Reference

Files

- file admin_html.h

  HTML for the admin page.
- file buzzer.h
- 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 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 induvidual users.
- file lock_ctrl.h
- file login_html.h

  HTML file for the login page.
- file rfid_access.h
- 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

  Used to read and write non-volatile memory on ESP8266.

- file fridge_state.cpp
- file graph_data.cpp
- file init_users_and_sale.cpp
- file inventory.cpp

  Functions responsible for keeping track of the fridge inventory.
- file lock_ctrl.cpp
- file main.cpp

  Combined: Web server + Graph + Scale + RFID access + Lock control.
- file rfid_access.cpp
- file sale_html.cpp
- file weight_scale.cpp

Chapter 6

Class Documentation

6.1 inventory Struct Reference

Strukt used for making the inventory.

`#include <inventory.h>`

Public Attributes

- products_stocked producktes_in_inventory [INVENTORY_CAPACITY]
  The beverages being stocked.
- uint8_t number_of_products_stocked
  Number of beverages stocked, prevents overflow.
- uint8_t room_number
  Not used in the current system.

6.1.1 Detailed Description

Strukt used for making the inventory.

6.1.2 Member Data Documentation

6.1.2.1 number_of_products_stocked

`uint8_t inventory::number_of_products_stocked`

Number of beverages stocked, prevents overflow.

6.1.2.2 producktes_in_inventory

`products_stocked inventory::producktes_in_inventory[INVENTORY_CAPACITY]`

The beverages being stocked.

6.1.2.3 room_number

```
uint8_t inventory::room_number
```

Not used in the current system.

Store the FRID belonging to the spesific user, not implemented

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

- include/inventory.h

6.2 product Struct Reference

Struct holding the information for one item.

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

Public Attributes

- char name [20]
  Beverage name.
- beverage_type beverage_variant
  Beverage type.
- uint16_t weight
  Beverage weight.
- uint8_t price
  Beverage price.

6.2.1 Detailed Description

Struct holding the information for one item.

6.2.2 Member Data Documentation

6.2.2.1 beverage_variant

```
beverage_type product::beverage_variant
```

Beverage type.

6.2.2.2 name

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

Beverage name.

6.2.2.3 price

```
uint8_t product::price
```

Beverage price.

6.2.2.4 weight

```
uint16_t product::weight
```

Beverage weight.

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

- include/inventory.h

6.3 products_stocked Struct Reference

Struckt for keeping track of the original and current quantity of a beverage.

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

Public Attributes

- product beverage
  The beverage being tracked.
- uint8_t original_quantity
  The original quantity of the beverage in stock.
- uint8_t current_quantity
  The current quantity of the beverage in stock.

6.3.1 Detailed Description

Struckt for keeping track of the original and current quantity of a beverage.

6.3.2 Member Data Documentation

6.3.2.1 beverage

```
product products_stocked::beverage
```

The beverage being tracked.

6.3.2.2 current_quantity

```
uint8_t products_stocked::current_quantity
```

The current quantity of the beverage in stock.

6.3.2.3 original_quantity

```
uint8_t products_stocked::original_quantity
```

The original quantity of the beverage in stock.

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 Detailed Description

User record stored in the RFID user database.

6.4.2 Member Data Documentation

6.4.2.1 balance

```
int User::balance
```

6.4.2.2 roomNumber

```
int User::roomNumber
```

6.4.2.3 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 for the admin page, is handled by the compiler as a string.

7.1.1 Detailed Description

HTML for the admin page.

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 inportant stuff goes here! </p>
        </body>
    </html>
)rawliteral"
```

HTML for the admin page, is handled by the compiler as a string.

Opening container for the sales graph.

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 inportant stuff goes here! </p>
00033         </body>
00034     </html>
00035 )rawliteral";
00036
00037 #endif
```

7.3 include/buzzer.h File Reference

```
#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 ()

  Play when the door is unlocked.
- void play_lock ()

  Play when the door locks.

7.3.1 Macro Definition Documentation

7.3.1.1 BUZZER_H

```
#define BUZZER_H
```

7.3.2 Function Documentation

7.3.2.1 play_lock()

```
void play_lock ()
```

Play when the door locks.

7.3.2.2 play_unlock()

```
void play_unlock ()
```

Play when the door is unlocked.

7.3.2.3 play_warning()

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

Plays short sound when door is closed and about to be locked.

Parameters

t	

7.4 buzzer.h

Go to the documentation of this file.

```
00001 #include <Arduino.h> //Tone functions don't work without this here
00002
00003 #ifndef BUZZER_H
00004 #define BUZZER_H
00005
00011 void play_warning(unsigned long t);
00012
00016 void play_unlock();
00017
00021 void play_lock();
00022
00023 #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

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

Author

Baldur G. Toftegaard

7.5.2 Variable Documentation

7.5.2.1 fridge

```
inventory fridge  [extern]
```

Used to set up the fridge inventory.

7.6 fridge_state.h

Go to the documentation of this file.

```
00001
00006
00007 #ifndef FRIDGE_STATE_H
00008 #define FRIDGE_STATE_H
00009
00010 #include "inventory.h"
00011
00015 extern inventory fridge;
00016
00017 #endif
```

7.7 include/graph_data.h File Reference

```
#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 Macro Definition Documentation

7.7.1.1 ROOM_COUNT

```
#define ROOM_COUNT 18
```

Number of rooms supported by the sales graph.

7.7.2 Function Documentation

7.7.2.1 graph_add_to_room_clasic()

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

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

Parameters

roomNumber	
delta	

7.7.2.2 graph_add_to_room_green()

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

Adds a value to the green sales bar of a given room.

Parameters

roomNumber	

delta	

7.7.2.3 print_graph_arrays()

```
void print_graph_arrays ()
```

Prints the current graph height arrays.

7.7.3 Variable Documentation

7.7.3.1 classicHeight

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

Height values for classic product sales per room.

7.7.3.2 greenHeight

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

Height values for green product sales per room.

7.8 graph_data.h

Go to the documentation of this file.

```
00001 #ifndef GRAPH_DATA_H
00002 #define GRAPH_DATA_H
00003
00004 #include <stdint.h>
00005
00010 #define ROOM_COUNT 18
00011
00015 extern int greenHeight[ROOM_COUNT];
00016
00020 extern int classicHeight[ROOM_COUNT];
00021
00028 void graph_add_to_room_green(uint8_t roomNumber, int delta);
00029
00036 void graph_add_to_room_clasic(uint8_t roomNumber, int delta);
00037
00041 void print_graph_arrays();
00042
00043 #endif
```

7.9 include/index_html.h File Reference

HTML for the start page.

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

Variables

- const char INDEX_HTML_HEAD[ ] PROGMEM

  Sting used for the HTML header.

7.9.1 Detailed Description

HTML for the start page.

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"
```

Sting used for the HTML header.

Opening container for the sales graph.

String used for the HTML footer.

This is responsible for updating the graphs.

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

  Function to actually performe a sale between a user and the fridge.

7.11.1 Detailed Description

Function used for declearing the system users and products.

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 initualicing users and products, shuld be moved to fridge_state.h.

Old function for initualicing users and products, shuld be moved to fridge_state.h.

7.11.3.2 perform_sale()

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

Function to actually performe a sale between a user and the fridge.

Parameters

weight	
user_id	
fridge_inventory	

Parameters

fridge_inventory	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();
00017
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 induvidual users.

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

Classes

- struct product

  Struct holding the information for one item.
- struct products_stocked

  Struckt for keeping track of the original and current quantity of a beverage.
- struct inventory

  Strukt used for making the inventory.

Macros

- #define INVENTORY_CAPACITY 6

Enumerations

- enum beverage_type {
  beer , cider , soda , limfjords_porter ,
  other }

Functions

- void inventory_init (inventory *inventory)

  Function for initualicing the inventory.
- product inventory_make_product (const char *name, beverage_type type, uint16_t weight, uint8_t price)

  Function for making a new product.
- bool inventory_add_product (inventory *inventory, product product, uint16_t quantity)

  Function for adding product to inventory.
- bool inventory_remove_product (inventory *inventory, product beverage)

  Function for removing product from inventory.
- bool inventory_add_beverage (inventory *inventory, product beverage, uint16_t amount)

  Function for adding to the amount of a beverage in an inventory.
- 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.

7.13.1 Detailed Description

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

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)
```

Function for adding to the amount of a beverage in an inventory.

Parameters

inventory	
beverage_type	
amount	

Returns

true - the beverage was added to the inventory

false - there was an error adding the beverage

Parameters

inventory	The inventory you want to add a beverage to
beverage	The beverage you want to edit the amount of
amount	The amount you want to add to the inventory

7.13.4.2 inventory_add_product()

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

Function for adding product to inventory.

Parameters

inventory	
beverage	
quantity	

Returns

true - the product was added to the inventory

false - there was an error adding the product

Parameters

inventory	Inventory you want to add a product to

product	The product you want to add to the inventory
quantity	The amount of the priduct you want to add to the inventory

7.13.4.3 inventory_init()

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

Function for initualicing the inventory.

Parameters

Inventory	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)
```

Function for making a new product.

Parameters

name	
type	
weight	
price	

Returns

item created

Parameters

name	Display name of product
type	What type of product it is, whuld allow to sort by product type
weight	Weight of the product, used for detecting how much of the product that was removed
price	Price of the product, this whuld make it posible to automaticaly calculate the bill for eatch user

7.13.4.5 inventory_print()

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

Function to print a users inventory.

Parameters

inventory	

Parameters

inventory	Inventory you want to print the content of

7.13.4.6 inventory_remove_beverage()

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

Function for removing from the amount of a beverage in an inventory.

Parameters

inventory	
beverag	
amount	

Returns

true - the beverage was removed from the inventory

false - there was an error removing the beverage

Parameters

inventory	The inventory you want to add a beverage to
beverag	The beverage you want to edit the amount of
amount	The amount you want to remove from the inventory

7.13.4.7 inventory_remove_product()

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

Function for removing product from inventory.

Parameters

inventory	

beverage	
quantity	

Returns

true - the product was removed from the inventory

false - there was an error removing the product

Parameters

inventory	Inventory you want to remove a product from
beverage	Beverage you want to remove

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
00014 #define INVENTORY_CAPACITY 6
00015
00020 typedef enum {
00021     beer,
00022     cider,
00023     soda,
00024     limfjords_porter,
00025     other
00026 } beverage_type;
00027
00031 typedef struct {
00032     char name[20];
00033     beverage_type beverage_variant;
00034     uint16_t weight;
00035     uint8_t price;
00036 } product;
00037
00041 typedef struct {
00042     product beverage;
00043     uint8_t original_quantity;
00044     uint8_t current_quantity;
00045 } products_stocked;
00046
00051 typedef struct {
00052     products_stocked produckts_in_inventory[INVENTORY_CAPACITY];
00053     uint8_t number_of_products_stocked;
00054     uint8_t room_number;
00056 } inventory;
00057
00058
00064 void inventory_init(
00065     inventory *inventory    /*< Inventory you cant to initialize, ensures that the memory space is
      empty */
00066 );
00067
00077 product inventory_make_product(
00078     const char *name,
00079     beverage_type type,
00080     uint16_t weight,
00081     uint8_t price
00082 );
00083
00093 bool inventory_add_product (
00094         inventory *inventory,
00095         product product,
```

```
00096         uint16_t quantity
00097 );
00098
00108 bool inventory_remove_product(
00109     inventory *inventory,
00110     product beverage
00111 );
00112
00122 bool inventory_add_beverage(
00123     inventory *inventory,
00124     product  beverage,
00125     uint16_t amount
00126 );
00127
00137 bool inventory_remove_beverage(
00138     inventory *inventory,
00139     product beverag,
00140     uint8_t amount
00141 );
00142
00148 void inventory_print(
00149     inventory *inventory
00150 );
00151
00152 #endif
```

7.15 include/lock_ctrl.h File Reference

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

Macros

- #define SERVO_PIN 16
- #define LIGHT_PIN A0
- #define CLOSED_THRESHOLD 70
- #define OPEN_THRESHOLD 100

Functions

- void lock_ctrl_init ()
- void lock_door ()
- void unlock_door ()
- bool is_box_closed ()
- void play_warning (unsigned long t)

  Play the warning sound effet.
- void play_open ()

  Play sound effect when door opens.
- void play_close ()

  Play sound effect when the door closes.

7.15.1 Detailed Description

Authors

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 ()
```

7.15.3.2 lock_ctrl_init()

```
void lock_ctrl_init ()
```

7.15.3.3 lock_door()

```
void lock_door ()
```

7.15.3.4 play_close()

```
void play_close ()
```

Play sound effect when the door closes.

7.15.3.5 play_open()

```
void play_open ()
```

Play sound effect when door opens.

7.15.3.6 play_warning()

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

Play the warning sound effet.

Parameters

t	

Play the warning sound effet.

Parameters

t	

7.15.3.7 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
00013 #define SERVO_PIN 16   // D0 = GP16, D1 = GPIO5
00014 #define LIGHT_PIN A0
00015 #define CLOSED_THRESHOLD 70   // darker than this = closed
00016 #define OPEN_THRESHOLD 100    // to avoid issues
00017
00018 // Only call once in setup()
00019 void lock_ctrl_init();
00020
00021 // Actions
00022 void lock_door();
00023 void unlock_door();
00024
00025 // Photosensor state
00026 bool is_box_closed();
00027
00033 void play_warning(unsigned long t);
00034
00038 void play_open();
00039
00043 void play_close();
00044
00045 #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

HTML file for the login page.

Authors

Baldur G. Toftegaard

7.17.2 Variable Documentation

7.17.2.1 PROGMEM

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

HTML for the login page.

Opening container for the sales graph.

Interperated 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

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

Classes

- struct User

  User record stored in the RFID user database.

Macros

- #define SS_PIN 15
- #define RST_PIN 0
- #define MAX_ROOMS 17
- #define UID_LENGTH 4

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

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
```

Supported serial commands for the RFID management interface.

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)
```

Adds a new user by reading room number and scanning an RFID tag.

Parameters

rfid	

Returns

true

false

7.19.4.2 check_command()

```
RFIDcommand check_command (
            void )
```

brief Reads a command from the serial interface and maps it to an RFIDcommand.

Returns

RFIDcommand

7.19.4.3 compare_UID()

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

Compares two RFID UIDs.

Parameters

uid1	
uid2	

Returns

true

false

7.19.4.4 count_rooms()

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

brief Counts the number of occupied user entries in the database.

Parameters

ptr	

Returns

int

7.19.4.5 display_commands()

```
void display_commands (
            void )
```

Prints the available serial commands.

7.19.4.6 display_commands_um()

```
void display_commands_um ()
```

Prints the available serial commands for user-management mode.

7.19.4.7 find_empty_index()

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

brief Finds an empty slot in the user database.

Parameters

ptr	

Returns

int

7.19.4.8 get_users_db()

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

Copies the current user database to a provided buffer.

Parameters

ptr	

7.19.4.9 print_all_users()

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

Prints all users in the database to the serial interface.

Parameters

ptr	

7.19.4.10 print_single_user()

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

brief Prints a single user entry to the serial interface.

Parameters

ptr	

idx	

7.19.4.11 print_uid()

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

Prints a UID buffer to the serial interface.

Parameters

ptr	

7.19.4.12 read_confirmation()

```
bool read_confirmation ()
```

Reads a confirmation input from the serial interface.

Returns

true

false

7.19.4.13 read_integer()

```
int read_integer ()
```

Reads an integer from the serial interface.

Returns

int

7.19.4.14 read_RFID_tag()

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

Reads an RFID tag UID from the MFRC522 reader.

Parameters

rfid	

uidBuffer	

Returns

true

false

7.19.4.15 remove_user()

```
bool remove_user ()
```

Removes a user from the database.

Returns

true

false

7.19.4.16 rfid_get_last_uid()

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

Function for restoring the last used RFID.

Parameters

uidOut	

Returns

true

false

7.19.4.17 rfid_set_last_uid()

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

Function for storing the last used RFID.

Parameters

uidOut	

Returns

true

false

7.19.4.18 setup_RFID_reader()

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

Initializes SPI and the MFRC522 RFID reader.

Parameters

rfid	

7.19.4.19 user_management()

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

Executes user-management actions based on the provided command.

Parameters

cmd	
ptr	
rfid	

7.19.4.20 validate_rfid()

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

Validates an RFID tag against the registered user database.

Parameters

myRFID	

Returns

true

false

7.19.5 Variable Documentation

7.19.5.1 userCount

```
int userCount  [extern]
```

Number of currently registered users.

7.19.5.2 users

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

Global user database array.

7.20 rfid_access.h

Go to the documentation of this file.

```
00001
00007
00008 #ifndef RFID_ACCESS_H
00009 #define RFID_ACCESS_H
00010
00011 #include <SPI.h>
00012 #include <MFRC522.h>
00013
00014 // Pins
00015 #define SS_PIN 15 // Use GPIO pins for HUZZAH instead of D8
00016 #define RST_PIN 0 // Instead of D3
00017
00018 // Constants
00019 #define MAX_ROOMS 17
00020 #define UID_LENGTH 4
00021
00025 enum RFIDcommand {
00026     CMD_NONE,
00027     CMD_ADD_USER,
00028     CMD_OPEN,
00029     CMD_LOCK,
00030     CMD_REMOVE_USER,
00031     CMD_CONFIRM,
00032     CMD_PRINT
00033 };
00034
00038 struct User {
00039     byte uid[UID_LENGTH];
00040     int roomNumber;
00041     int balance;
00042 };
00043
00047 extern User users[MAX_ROOMS];
00048
00052 extern int userCount;
00053
00059 RFIDcommand check_command(
00060     void
00061 );
00062
00068 void setup_RFID_reader(
00069     MFRC522 &rfid
00070 );
00071
00079 bool add_user(
00080     MFRC522 &rfid
00081 );
00082
00089 bool remove_user(
00090 );
00091
00100 bool compare_UID(
00101     byte *uid1,
00102     byte *uid2
00103 );
00104
00113 bool read_RFID_tag(
00114     MFRC522 &rfid,
00115     byte *uidBuffer
00116 );
00117
00121 void display_commands(
00122     void
00123 );
00124
00128 void display_commands_um(
00129 );
00130
00136 void get_users_db(
```

```
00137     User* ptr
00138 );
00139
00147 void user_management(
00148     RFIDcommand cmd,
00149     User* ptr,
00150     MFRC522 &rfid
00151 );
00152
00160 bool validate_rfid(
00161     MFRC522 myRFID
00162 );
00163
00170 void print_single_user(
00171     User* ptr,
00172     int idx
00173 );
00179 void print_all_users(
00180     User* ptr
00181 );
00182
00188 void print_uid(
00189     byte* ptr
00190 );
00191
00197 int read_integer(
00198 );
00199
00206 bool read_confirmation(
00207 );
00208
00215 int find_empty_index(
00216     User* ptr
00217 );
00218
00225 int count_rooms(
00226     User* ptr
00227 );
00228
00235 void rfid_set_last_uid(
00236     const byte *uidIn
00237 );
00238
00245 bool rfid_get_last_uid(
00246     byte *uidOut
00247 );
00248
00249
00250 #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

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

Authors

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)
```

Sends a single sales bar element to the client.

Parameters

server	
room_number	
bar_type	
bar_height	

Parameters

server	The server
room_number	Number of the relevant room
bar_type	The type of the bar graph
bar_height	The height of the bar graph

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)
```

Sends the complete sales graph page.

Parameters

server	

room_count	
greenHeight	
classicHeights	

Parameters

server	The server
room_count	The number of rooms
greenHeight	The hight of the green bar
classicHeights	The hight of the clasic bar (not used in prototype)

7.21.3 Variable Documentation

7.21.3.1 PROGMEM

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

Opening container for the sales graph.

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.

Opening container for the sales graph.

String used for the HTML footer.

This is responsible for updating the graphs.

Opening container for the sales graph.

Interperated 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

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

The css is handled as a string by the compiler.

7.23.2 Variable Documentation

7.23.2.1 PROGMEM

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

String containing the CSS styling of the webserver.

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
00006
00007 #ifndef STYLE_CSS_H
00008 #define STYLE_CSS_H
00009 #include <pgmspace.h>
00010
00014 const char STYLE_CSS[] PROGMEM = R"rawliteral(
00015 /* ---------- Global page styling ---------- */
00016 body {
00017   margin: 0;
00018   padding: 0;
00019   font-family: Arial, Helvetica, sans-serif;
00020   background-color: #f0f0f0;
00021 }
00022
00023 /* Bar on top of page for displaying message/path and log in */
00024 .topbar {
00025   display: flex;
00026   justify-content: space-between;
00027   align-items: center;
00028   background-color: #2c3e50;
00029   color: #fff;
00030   padding: 12px 16px;
00031   box-sizing: border-box;
00032 }
00033
00034 /* message/path display box */
00035 .topbar .right a {
00036   color: #fff;
00037   text-decoration: none;
00038   border: 1px solid rgba(255,255,255,0.5);
00039   padding: 6px 10px;
00040   border-radius: 6px;
00041 }
00042
00043 /* login button */
00044 .topbar .right a:hover {
00045   background: rgba(255,255,255,0.15);
00046 }
00047
00048 /* box for the graph to be placed inside */
00049 .sale_box {
00050   margin: 40px auto;
00051   width: calc(100% - 100px);
00052   height: 500px;
00053
00054   background-color: #ffffff;
00055   border: 2px solid #000000;
00056   box-sizing: border-box;
00057
00058   display: flex;
00059   justify-content: space-evenly;
00060   align-items: flex-end;
00061   padding: 15px;
00062   gap: 0;
00063 }
00064
00065 /* graph box for eatch room */
00066 .sale_room {
```

```
00067   display: flex;
00068   align-items: flex-end;
00069   gap: 4px;
00070   height: 100%;
00071 }
00072
00073 /* Bars */
00074 .sale_pole_green {
00075   width: 22px;
00076   border: 1px solid #000000;
00077   box-sizing: border-box;
00078 }
00079
00080 .sale_pole_clasic {
00081   width: 22px;
00082   border: 1px solid #ffffff;
00083   box-sizing: border-box;
00084 }
00085
00086 .sale_pole_green { background-color: #2e7d32; }
00087 .sale_pole_clasic { background-color: #ffffff; }
00088 )rawliteral";
00089
00090 #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 error of the weight mesurment.
- #define HX711_DOUT 4
- #define HX711_SCK 5
- #define SCALE_DEFAULT_SETTLE_TIME_MS 3000

Enumerations

- enum weight_recall_action { weight_change_store , weight_change_recall }

  set up the action (input) the rfid_user_id_lattest takes

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 cinfirmation 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 tar the scale.

- bool tare_complete (void)

  Function to signal that the scale has been tarted.

- 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
```

Define the weight of a beer.

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
```

Define the error of the weight mesurment.

7.25.3 Enumeration Type Documentation

7.25.3.1 weight_recall_action

```
enum weight_recall_action
```

set up the action (input) the rfid_user_id_lattest takes

Enumerator

weight_change_store	

weight_change_recall	

7.25.4 Function Documentation

7.25.4.1 get_beer_cans_taken()

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

Function to get the number of beer cans taken.

Parameters

referencWeight	
currentWeight	

Returns

int

7.25.4.2 get_weight()

```
float get_weight (
            void )
```

Function to get the scale reading.

Returns

float

7.25.4.3 get_weight_reference()

```
float get_weight_reference (
            void )
```

FUnction to get the weight reference.

Returns

float

7.25.4.4 reset_weight_reference()

```
void reset_weight_reference (
            void )
```

Function to reset the weight reference.

7.25.4.5 set_weight_reference()

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

Function to set the weight reference.

Parameters

value	

7.25.4.6 setup_scale()

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

Function to seting up the scale, is called in the begining of the program.

Parameters

calFactor	

7.25.4.7 tare_complete()

```
bool tare_complete (
            void )
```

Function to signal that the scale has been tarted.

Returns

true

false

7.25.4.8 tare_scale()

```
void tare_scale (
            void )
```

Function to tar the scale.

7.25.4.9 update_scale()

```
bool update_scale (
            void )
```

Function for updating the scale value.

Returns

true

false

7.25.4.10 weight_reference_is_set()

```
bool weight_reference_is_set (
            void )
```

Function to send cinfirmation that the weight reference is set.

Returns

true

false

7.25.5 Variable Documentation

7.25.5.1 scale

```
HX711_ADC scale  [extern]
```

7.26 weight_scale.h

Go to the documentation of this file.

```
00001
00005 #ifndef WEIGHT_SCALE_H
00006 #define WEIGHT_SCALE_H
00007
00008 #include <Arduino.h>
00009 #include <HX711_ADC.h>
00010 #include <math.h>
00011
00017 float get_weight_reference(
00018     void
00019 );
00020
00026 void  set_weight_reference(
00027     float value
00028 );
00029
00033 void  reset_weight_reference(
00034     void
00035 );
00036
00043 bool  weight_reference_is_set(
00044     void
00045 );
```

```
00046
00050 #define BEER_WEIGHT 350
00051
00055 #define SCALE_TOL 25
00056
00057 // Pins
00058 #define HX711_DOUT 4   // GPIO4=D2
00059 #define HX711_SCK  5   // GPIO5=D1
00060
00061 // Config scale
00062 #define SCALE_DEFAULT_SETTLE_TIME_MS 3000
00063
00064 // Globals
00065 extern HX711_ADC scale;
00066
00070 enum weight_recall_action {
00071     weight_change_store,
00072     weight_change_recall
00073 };
00074
00080 void setup_scale(
00081     float calFactor
00082 );
00083
00090 bool update_scale(
00091     void
00092 );
00093
00099 float get_weight(
00100     void
00101 );
00102
00106 void tare_scale(
00107     void
00108 );
00109
00116 bool tare_complete(
00117     void
00118 );
00119
00127 int get_beer_cans_taken(
00128     float referencWeight,
00129     float currentWeight
00130 );
00131
00132 #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 ()

  Play when the door is unlocked.
- void play_lock ()

  Play when the door locks.

Variables

- const int BUZZERPIN = 2
- const double HIGH_TONE = 1000
- const double LOW_TONE = 600
- const unsigned long TONE_LENGTH = 200

7.28.1 Function Documentation

7.28.1.1 play_lock()

```
void play_lock ()
```

Play when the door locks.

7.28.1.2 play_unlock()

```
void play_unlock ()
```

Play when the door is unlocked.

7.28.1.3 play_warning()

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

Plays short sound when door is closed and about to be locked.

Play the warning sound effet.

Parameters

t	

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

Used to read and write 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

Used to read and write non-volatile memory on ESP8266.

Authors

Anssi Sohlman,

Date

16-01-2026

Version

0.1

Revision history

| Version | Date | Description | |---------|----------|---------------------------------| | 0.1 | 16-01-2026 | Initial version | |

Copyright

Copyright (c) 2026

7.29.2 Function Documentation

7.29.2.1 count_rooms()

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

brief Counts the number of occupied user entries in the database.

Parameters

ptr	

Returns

int

7.29.2.2 find_empty_index()

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

brief Finds an empty slot in the user database.

Parameters

ptr	

Returns

int

7.29.2.3 get_users_db()

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

Copies the current user database to a provided buffer.

Parameters

ptr	

7.29.2.4 print_all_users()

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

Prints all users in the database to the serial interface.

Parameters

ptr	

7.29.2.5 print_single_user()

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

brief Prints a single user entry to the serial interface.

Parameters

ptr	
idx	

7.29.2.6 print_uid()

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

Prints a UID buffer to the serial interface.

Parameters

ptr	

7.29.2.7 read_confirmation()

```
bool read_confirmation ()
```

Reads a confirmation input from the serial interface.

Returns

true

false

7.29.2.8 read_integer()

```
int read_integer ()
```

Reads an integer from the serial interface.

Returns

int

7.29.2.9 remove_user()

```
bool remove_user ()
```

Removes a user from the database.

Returns

true

false

7.29.2.10 user_management()

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

Executes user-management actions based on the provided command.

Parameters

cmd	
ptr	
rfid	

7.30 src/fridge_state.cpp File Reference

```
#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
```

Used to set up the fridge inventory.

7.31 src/graph_data.cpp File Reference

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

7.31.1 Function Documentation

7.31.1.1 graph_add_to_room_clasic()

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

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

Parameters

roomNumber	

delta	

7.31.1.2 graph_add_to_room_green()

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

Adds a value to the green sales bar of a given room.

Parameters

roomNumber	
delta	

7.31.2 Variable Documentation

7.31.2.1 classicHeight

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

Height values for classic product sales per room.

7.31.2.2 greenHeight

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

Height values for green product sales per room.

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)

  Function to actually performe 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 ()
```

Not used in prototype, shuld be moved to inventory.cpp.

Old function for initualicing users and products, shuld be moved to fridge_state.h.

7.32.1.2 perform_sale()

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

Function to actually performe a sale between a user and the fridge.

Parameters

weight	
user_id	
fridge_inventory	

Parameters

fridge_inventory	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)

  Function for initualicing the inventory.
- product inventory_make_product (const char *product_name, beverage_type type, uint16_t weight, uint8_t price)

  Function for making a new product.
- bool inventory_add_product (inventory *inventory, product product, uint16_t quantity)

  Function for adding product to inventory.
- bool inventory_remove_product (inventory *inventory, product beverage)

  Function for removing product from 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

Functions responsible for keeping track of the fridge inventory.

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)
```

Function for adding product to inventory.

Parameters

inventory	
beverage	

quantity	

Returns

true - the product was added to the inventory

false - there was an error adding the product

Parameters

inventory	Inventory you want to add a product to
product	The product you want to add to the inventory
quantity	The amount of the priduct you want to add to the inventory

7.33.2.3 inventory_init()

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

Function for initualicing the inventory.

Parameters

Inventory	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)
```

Function for making a new product.

Parameters

name	
type	
weight	
price	

Returns

item created

Parameters

product_name	Display name of product

type	What type of product it is, whuld allow to sort by product type
weight	Weight of the product, used for detecting how much of the product that was removed
price	Price of the product, this whuld make it posible to automaticaly calculate the bill for eatch user

7.33.2.5 inventory_print()

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

Function to print a users inventory.

Parameters

inventory	

Parameters

inventory	Inventory you want to print the content of

7.33.2.6 inventory_remove_beverage()

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

Function for removing from the amount of a beverage in an inventory.

Parameters

inventory	
beverag	
amount	

Returns

true - the beverage was removed from the inventory

false - there was an error removing the beverage

Parameters

inventory	The inventory you want to add a beverage to
beverag	The beverage you want to edit the amount of
amount	The amount you want to remove from the inventory

7.33.2.7 inventory_remove_product()

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

Function for removing product from inventory.

Parameters

inventory	
beverage	
quantity	

Returns

true - the product was removed from the inventory

false - there was an error removing the product

Parameters

inventory	Inventory you want to remove a product from
beverage	Beverage you want to remove

7.34 src/lock_ctrl.cpp File Reference

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

Functions

- void lock_ctrl_init ()
- void lock_door ()
- void unlock_door ()
- bool is_box_closed ()
- void play_open ()

  Play sound effect when door opens.
- void play_close ()

  Play sound effect when the door closes.

Variables

- static Servo lockServo
- static const int UNLOCK_POS = 0
- static const int LOCK_POS = 100
- const int BUZZER = 2
- const double HIGH_TONE = 1000
- const double LOW_TONE = 600
- const unsigned long TONE_LENGTH = 200
- static bool boxClosed = false

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 ()
```

7.34.2.2 lock_ctrl_init()

```
void lock_ctrl_init ()
```

7.34.2.3 lock_door()

```
void lock_door ()
```

7.34.2.4 play_close()

```
void play_close ()
```

Play sound effect when the door closes.

7.34.2.5 play_open()

```
void play_open ()
```

Play sound effect when door opens.

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
```

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

- void print_graph_arrays ()

  Prints the current graph height arrays.
- 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
- static const uint16_t START_BEER_QTY = 20
- const char * WIFI_SSID = "Baldur's A56"
- const char * WIFI_PASS = "MyPasskeyA56"
- int greenHeight [ROOM_COUNT] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}

  Height values for green product sales per room.
- int classicHeight [ROOM_COUNT] = {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.
- product demo_beer
- RFIDcommand activeCommand = CMD_NONE
- bool doorUnlocked = false
- unsigned long doorCloseTimer = 0

7.35.1 Detailed Description

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

Author

Baldur G. Toftegaard

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 print_graph_arrays()

```
void print_graph_arrays ()
```

Prints the current graph height arrays.

7.35.2.4 rfid()

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

7.35.2.5 server()

```
ESP8266WebServer server (
             80 )
```

7.35.2.6 setup()

```
void setup ()
```

7.35.2.7 setup_inventory_and_scale()

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

7.35.2.8 setup_rfid_and_lock()

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

7.35.2.9 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] = {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.

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

Height values for green product sales per room.

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 = "Baldur's A56"
```

7.36 src/rfid_access.cpp File Reference

```
#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)
```

Adds a new user by reading room number and scanning an RFID tag.

Parameters

rfid	

Returns

true

false

7.36.2.2 check_command()

```
RFIDcommand check_command (
            void )
```

brief Reads a command from the serial interface and maps it to an RFIDcommand.

Returns

RFIDcommand

7.36.2.3 compare_UID()

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

Compares two RFID UIDs.

Parameters

uid1	
uid2	

Returns

true

false

7.36.2.4 display_commands()

```
void display_commands (
            void )
```

Prints the available serial commands.

7.36.2.5 display_commands_um()

```
void display_commands_um ()
```

Prints the available serial commands for user-management mode.

7.36.2.6 read_RFID_tag()

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

Reads an RFID tag UID from the MFRC522 reader.

Parameters

rfid	

uidBuffer	

Returns

true

false

7.36.2.7 rfid_get_last_uid()

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

Function for restoring the last used RFID.

Parameters

uidOut	

Returns

true

false

7.36.2.8 rfid_set_last_uid()

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

Function for storing the last used RFID.

Parameters

uidOut	

Returns

true

false

7.36.2.9 setup_RFID_reader()

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

Initializes SPI and the MFRC522 RFID reader.

Parameters

rfid	

7.36.2.10 validate_rfid()

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

Validates an RFID tag against the registered user database.

Parameters

myRFID	

Returns

true

false

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
```

Number of currently registered users.

7.36.3.4 users

```
User users[MAX_ROOMS]
```

Global user database array.

7.37 src/sale_html.cpp File Reference

```
#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( )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)
```

Sends a single sales bar element to the client.

Parameters

server	
room_number	
bar_type	
bar_height	

Parameters

server	The server
room_number	Number of the relevant room
bar_type	The type of the bar graph
bar_height	The height of the bar graph

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)
```

Sends the complete sales graph page.

Parameters

server	
room_count	
greenHeight	
classicHeights	

Parameters

server	The server
room_count	The number of rooms
greenHeight	The hight of the green bar
classicHeights	The hight of the clasic bar (not used in prototype)

7.37.3 Variable Documentation

7.37.3.1 PROGMEM

```
const char SALE_BOX_STOP [] PROGMEM = R"rawliteral( <div class="sale_box">)rawliteral"
```

Opening container for the sales graph.

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.

Opening container for the sales graph.

String used for the HTML footer.

This is responsible for updating the graphs.

Opening container for the sales graph.

Interperated 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

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

Functions

- HX711_ADC scale (HX711_DOUT, HX711_SCK)
- 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 cinfirmation 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 ()
  Function for updating the scale value.
- float get_weight ()
  Function to get the scale reading.
- void tare_scale ()
  Function to tar the scale.
- bool tare_complete ()
  Function to signal that the scale has been tarted.
- 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)
```

Function to get the number of beer cans taken.

Parameters

referencWeight	

currentWeight	

Returns

int

7.38.2.2 get_weight()

```
float get_weight (
            void )
```

Function to get the scale reading.

Returns

float

7.38.2.3 get_weight_reference()

```
float get_weight_reference (
            void )
```

FUnction to get the weight reference.

Returns

float

7.38.2.4 reset_weight_reference()

```
void reset_weight_reference (
            void )
```

Function to reset the weight reference.

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)
```

Function to set the weight reference.

Parameters

value	

7.38.2.7 setup_scale()

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

Function to seting up the scale, is called in the begining of the program.

Parameters

calFactor	

7.38.2.8 tare_complete()

```
bool tare_complete (
            void )
```

Function to signal that the scale has been tarted.

Returns

true

false

7.38.2.9 tare_scale()

```
void tare_scale (
            void )
```

Function to tar the scale.

7.38.2.10 update_scale()

```
bool update_scale (
            void )
```

Function for updating the scale value.

Returns

true

false

7.38.2.11 weight_reference_is_set()

```
bool weight_reference_is_set (
            void )
```

Function to send cinfirmation that the weight reference is set.

Returns

true

false

7.38.3 Variable Documentation

7.38.3.1 g_referenceWeight

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

Index

activeCommand
    main.cpp, 70
add_user
    rfid_access.cpp, 72
    rfid_access.h, 36
admin_html.h
    PROGMEM, 15

balance
    User, 14
beer
    inventory.h, 25
BEER_WEIGHT
    weight_scale.h, 49
beverage
    products_stocked, 13
beverage_type
    inventory.h, 25
beverage_variant
    product, 12
BFIT, 1
boxClosed
    lock_ctrl.cpp, 67
BUZZER
    lock_ctrl.cpp, 67
buzzer.cpp
    BUZZERPIN, 55
    HIGH_TONE, 55
    LOW_TONE, 55
    play_lock, 55
    play_unlock, 55
    play_warning, 55
    TONE_LENGTH, 55
buzzer.h
    BUZZER_H, 17
    play_lock, 17
    play_unlock, 17
    play_warning, 17
BUZZER_H
    buzzer.h, 17
BUZZERPIN
    buzzer.cpp, 55

CAL_FACTOR
    main.cpp, 70
check_command
    rfid_access.cpp, 72
    rfid_access.h, 36
cider
    inventory.h, 25
classicHeight
    graph_data.cpp, 61
    graph_data.h, 20
    main.cpp, 70
CLOSED_THRESHOLD
    lock_ctrl.h, 31
CMD_ADD_USER
    rfid_access.h, 36
CMD_CONFIRM
    rfid_access.h, 36
CMD_LOCK
    rfid_access.h, 36
CMD_NONE
    rfid_access.h, 36
CMD_OPEN
    rfid_access.h, 36
CMD_PRINT
    rfid_access.h, 36
CMD_REMOVE_USER
    rfid_access.h, 36
compare_UID
    rfid_access.cpp, 73
    rfid_access.h, 36
connect_wifi_and_start_mdns
    main.cpp, 69
count_rooms
    database_management.cpp, 57
    rfid_access.h, 37
current_quantity
    products_stocked, 13

database_management.cpp
    count_rooms, 57
    find_empty_index, 57
    get_users_db, 57
    print_all_users, 57
    print_single_user, 58
    print_uid, 58
    read_confirmation, 58
    read_integer, 58
    remove_user, 59
    user_management, 59
demo_beer
    main.cpp, 71
display_commands
    rfid_access.cpp, 73
    rfid_access.h, 37
display_commands_um
    rfid_access.cpp, 73
    rfid_access.h, 37

doorCloseTimer
main.cpp, 71
doorUnlocked
main.cpp, 71

find_empty_index
database_management.cpp, 57
rfid_access.h, 37
fridge
fridge_state.cpp, 60
fridge_state.h, 18
fridge_state.cpp
fridge, 60
fridge_state.h
fridge, 18

g_referenceWeight
weight_scale.cpp, 81
get_beer_cans_taken
weight_scale.cpp, 78
weight_scale.h, 51
get_users_db
database_management.cpp, 57
rfid_access.h, 38
get_weight
weight_scale.cpp, 79
weight_scale.h, 51
get_weight_reference
weight_scale.cpp, 79
weight_scale.h, 51
graph_add_to_room_clasic
graph_data.cpp, 60
graph_data.h, 19
graph_add_to_room_green
graph_data.cpp, 61
graph_data.h, 19
graph_data.cpp
classicHeight, 61
graph_add_to_room_clasic, 60
graph_add_to_room_green, 61
greenHeight, 61
graph_data.h
classicHeight, 20
graph_add_to_room_clasic, 19
graph_add_to_room_green, 19
greenHeight, 20
print_graph_arrays, 20
ROOM_COUNT, 19
greenHeight
graph_data.cpp, 61
graph_data.h, 20
main.cpp, 71

hasUID
rfid_access.cpp, 75
HIGH_TONE
buzzer.cpp, 55
lock_ctrl.cpp, 68
HX711_DOUT
weight_scale.h, 49
HX711_SCK
weight_scale.h, 49

include Directory Reference, 9
include/admin_html.h, 15, 16
include/buzzer.h, 16, 17
include/fridge_state.h, 18
include/graph_data.h, 18, 20
include/index_html.h, 20, 22
include/init_users_and_sale.h, 22, 24
include/inventory.h, 24, 29
include/lock_ctrl.h, 30, 32
include/login_html.h, 32, 33
include/rfid_access.h, 34, 42
include/sale_html.h, 43, 46
include/style_css.h, 46, 47
include/weight_scale.h, 48, 53
index_html.h
PROGMEM, 21
init_users_and_products
init_users_and_sale.cpp, 62
init_users_and_sale.h, 23
init_users_and_sale.cpp
init_users_and_products, 62
perform_sale, 62
read_current_weight_blocking, 62
init_users_and_sale.h
init_users_and_products, 23
number_of_users, 23
perform_sale, 23
inventory, 11
number_of_products_stocked, 11
produckts_in_inventory, 11
room_number, 11
inventory.cpp
inventory_add_beverage, 63
inventory_add_product, 63
inventory_init, 64
inventory_make_product, 64
inventory_print, 65
inventory_remove_beverage, 65
inventory_remove_product, 65
inventory.h
beer, 25
beverage_type, 25
cider, 25
inventory_add_beverage, 26
inventory_add_product, 26
INVENTORY_CAPACITY, 25
inventory_init, 27
inventory_make_product, 27
inventory_print, 27
inventory_remove_beverage, 28
inventory_remove_product, 28
limfjords_porter, 25
other, 25
soda, 25
inventory_add_beverage

inventory.cpp, 63
inventory.h, 26
inventory_add_product
inventory.cpp, 63
inventory.h, 26
INVENTORY_CAPACITY
inventory.h, 25
inventory_init
inventory.cpp, 64
inventory.h, 27
inventory_make_product
inventory.cpp, 64
inventory.h, 27
inventory_print
inventory.cpp, 65
inventory.h, 27
inventory_remove_beverage
inventory.cpp, 65
inventory.h, 28
inventory_remove_product
inventory.cpp, 65
inventory.h, 28
is_box_closed
lock_ctrl.cpp, 67
lock_ctrl.h, 31

lastUID
rfid_access.cpp, 75
LIGHT_PIN
lock_ctrl.h, 31
limfjords_porter
inventory.h, 25
lock_ctrl.cpp
boxClosed, 67
BUZZER, 67
HIGH_TONE, 68
is_box_closed, 67
lock_ctrl_init, 67
lock_door, 67
LOCK_POS, 68
lockServo, 68
LOW_TONE, 68
play_close, 67
play_open, 67
TONE_LENGTH, 68
unlock_door, 67
UNLOCK_POS, 68
lock_ctrl.h
CLOSED_THRESHOLD, 31
is_box_closed, 31
LIGHT_PIN, 31
lock_ctrl_init, 31
lock_door, 31
OPEN_THRESHOLD, 31
play_close, 31
play_open, 31
play_warning, 31
SERVO_PIN, 31
unlock_door, 32
lock_ctrl_init
lock_ctrl.cpp, 67
lock_ctrl.h, 31
lock_door
lock_ctrl.cpp, 67
lock_ctrl.h, 31
LOCK_POS
lock_ctrl.cpp, 68
lockServo
lock_ctrl.cpp, 68
login_html.h
PROGMEM, 33
loop
main.cpp, 69
LOW_TONE
buzzer.cpp, 55
lock_ctrl.cpp, 68

main.cpp
activeCommand, 70
CAL_FACTOR, 70
classicHeight, 70
connect_wifi_and_start_mdns, 69
demo_beer, 71
doorCloseTimer, 71
doorUnlocked, 71
greenHeight, 71
loop, 69
print_graph_arrays, 69
rfid, 70
server, 70
setup, 70
setup_inventory_and_scale, 70
setup_rfid_and_lock, 70
setup_web_routes, 70
START_BEER_QTY, 71
WIFI_PASS, 71
WIFI_SSID, 71
MAX_ROOMS
rfid_access.h, 35

name
product, 12
number_of_products_stocked
inventory, 11
number_of_users
init_users_and_sale.h, 23

OPEN_THRESHOLD
lock_ctrl.h, 31
original_quantity
products_stocked, 14
other
inventory.h, 25

perform_sale
init_users_and_sale.cpp, 62
init_users_and_sale.h, 23
play_close

lock_ctrl.cpp, 67
lock_ctrl.h, 31
play_lock
buzzer.cpp, 55
buzzer.h, 17
play_open
lock_ctrl.cpp, 67
lock_ctrl.h, 31
play_unlock
buzzer.cpp, 55
buzzer.h, 17
play_warning
buzzer.cpp, 55
buzzer.h, 17
lock_ctrl.h, 31
price
product, 12
print_all_users
database_management.cpp, 57
rfid_access.h, 38
print_graph_arrays
graph_data.h, 20
main.cpp, 69
print_single_user
database_management.cpp, 58
rfid_access.h, 38
print_uid
database_management.cpp, 58
rfid_access.h, 39
produckts_in_inventory
inventory, 11
product, 12
beverage_variant, 12
name, 12
price, 12
weight, 13
products_stocked, 13
beverage, 13
current_quantity, 13
original_quantity, 14
PROGMEM
admin_html.h, 15
index_html.h, 21
login_html.h, 33
sale_html.cpp, 77
sale_html.h, 45
style_css.h, 47

read_confirmation
database_management.cpp, 58
rfid_access.h, 39
read_current_weight_blocking
init_users_and_sale.cpp, 62
read_integer
database_management.cpp, 58
rfid_access.h, 39
read_RFID_tag
rfid_access.cpp, 73
rfid_access.h, 39
README.md, 54
remove_user
database_management.cpp, 59
rfid_access.h, 40
reset_weight_reference
weight_scale.cpp, 79
weight_scale.h, 51
rfid
main.cpp, 70
rfid_access.cpp
add_user, 72
check_command, 72
compare_UID, 73
display_commands, 73
display_commands_um, 73
hasUID, 75
lastUID, 75
read_RFID_tag, 73
rfid_get_last_uid, 74
rfid_set_last_uid, 74
setup_RFID_reader, 74
userCount, 75
users, 75
validate_rfid, 74
rfid_access.h
add_user, 36
check_command, 36
CMD_ADD_USER, 36
CMD_CONFIRM, 36
CMD_LOCK, 36
CMD_NONE, 36
CMD_OPEN, 36
CMD_PRINT, 36
CMD_REMOVE_USER, 36
compare_UID, 36
count_rooms, 37
display_commands, 37
display_commands_um, 37
find_empty_index, 37
get_users_db, 38
MAX_ROOMS, 35
print_all_users, 38
print_single_user, 38
print_uid, 39
read_confirmation, 39
read_integer, 39
read_RFID_tag, 39
remove_user, 40
rfid_get_last_uid, 40
rfid_set_last_uid, 40
RFIDcommand, 36
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, 74
rfid_access.h, 40
rfid_set_last_uid
rfid_access.cpp, 74
rfid_access.h, 40
RFIDcommand
rfid_access.h, 36
ROOM_COUNT
graph_data.h, 19
room_number
inventory, 11
roomNumber
User, 14
RST_PIN
rfid_access.h, 35

sale_html.cpp
PROGMEM, 77
send_sale_html_graph, 76
send_sale_html_page, 76
sale_html.h
PROGMEM, 45
send_sale_html_graph, 44
send_sale_html_page, 44
scale
weight_scale.cpp, 79
weight_scale.h, 53
SCALE_DEFAULT_SETTLE_TIME_MS
weight_scale.h, 49
SCALE_TOL
weight_scale.h, 49
send_sale_html_graph
sale_html.cpp, 76
sale_html.h, 44
send_sale_html_page
sale_html.cpp, 76
sale_html.h, 44
server
main.cpp, 70
SERVO_PIN
lock_ctrl.h, 31
set_weight_reference
weight_scale.cpp, 79
weight_scale.h, 52
setup
main.cpp, 70
setup_inventory_and_scale
main.cpp, 70
setup_rfid_and_lock
main.cpp, 70
setup_RFID_reader
rfid_access.cpp, 74
rfid_access.h, 40
setup_scale
weight_scale.cpp, 80
weight_scale.h, 52
setup_web_routes
main.cpp, 70
soda
inventory.h, 25
src Directory Reference, 9
src/buzzer.cpp, 54
src/database_management.cpp, 56
src/fridge_state.cpp, 59
src/graph_data.cpp, 60
src/init_users_and_sale.cpp, 61
src/inventory.cpp, 62
src/lock_ctrl.cpp, 66
src/main.cpp, 68
src/rfid_access.cpp, 71
src/sale_html.cpp, 75
src/weight_scale.cpp, 78
SS_PIN
rfid_access.h, 35
START_BEER_QTY
main.cpp, 71
style_css.h
PROGMEM, 47

tare_complete
weight_scale.cpp, 80
weight_scale.h, 52
tare_scale
weight_scale.cpp, 80
weight_scale.h, 52
TONE_LENGTH
buzzer.cpp, 55
lock_ctrl.cpp, 68

uid
User, 14
UID_LENGTH
rfid_access.h, 35
unlock_door
lock_ctrl.cpp, 67
lock_ctrl.h, 32
UNLOCK_POS
lock_ctrl.cpp, 68
update_scale
weight_scale.cpp, 80
weight_scale.h, 52
User, 14
balance, 14
roomNumber, 14
uid, 14
user_management
database_management.cpp, 59
rfid_access.h, 41
userCount
rfid_access.cpp, 75
rfid_access.h, 41
users
rfid_access.cpp, 75
rfid_access.h, 41

validate_rfid

rfid_access.cpp, 74
rfid_access.h, 41

weight
product, 13
weight_change_recall
weight_scale.h, 51
weight_change_store
weight_scale.h, 50
weight_recall_action
weight_scale.h, 50
weight_reference_is_set
weight_scale.cpp, 80
weight_scale.h, 53
weight_scale.cpp
g_referenceWeight, 81
get_beer_cans_taken, 78
get_weight, 79
get_weight_reference, 79
reset_weight_reference, 79
scale, 79
set_weight_reference, 79
setup_scale, 80
tare_complete, 80
tare_scale, 80
update_scale, 80
weight_reference_is_set, 80
weight_scale.h
BEER_WEIGHT, 49
get_beer_cans_taken, 51
get_weight, 51
get_weight_reference, 51
HX711_DOUT, 49
HX711_SCK, 49
reset_weight_reference, 51
scale, 53
SCALE_DEFAULT_SETTLE_TIME_MS, 49
SCALE_TOL, 49
set_weight_reference, 52
setup_scale, 52
tare_complete, 52
tare_scale, 52
update_scale, 52
weight_change_recall, 51
weight_change_store, 50
weight_recall_action, 50
weight_reference_is_set, 53
WIFI_PASS
main.cpp, 71
WIFI_SSID
main.cpp, 71