BFIT

Generated by Doxygen 1.16.1

Chapter 1

BFIT

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

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

1.1 Features

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

1.2 Repository structure

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

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

1.3 Documentation

The Doxygen generated document is `final_doxygen_doc.pdf`.

Chapter 2

Directory Hierarchy

2.1 Directories

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Directory Documentation

5.1 include Directory Reference

Files

- file admin_html.h

  HTML for the admin page.
- file buzzer.h

  Declaring functions to play sounds using a buzzer.
- file fridge_state.h

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

  File used for setting up the graph section of the web server.
- file index_html.h

  HTML for the start page.
- file init_users_and_sale.h

  Function used for declearing the system users and products.
- file inventory.h

  Inventory system for tracking the inventory of both the fridge and the induvidual users.
- file lock_ctrl.h

  Door lock control interface using a servo and light sensor.
- file login_html.h

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

  Declare functions related to RFID reading and user database.
- file sale_html.h

  Headerfile for displaying the graph of sales on the main page.
- file style_css.h

  CSS served at /style.css used for setting the style for the web server.
- file weight_scale.h

5.2 src Directory Reference

Files

- file buzzer.cpp
- file database_management.cpp

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

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

  File used to set up the graph section in the web server.
- file init_users_and_sale.cpp
- file inventory.cpp

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

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

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

  File used to setup the RFID.
- file sale_html.cpp

  File for setting up the graph section of the web server.
- file weight_scale.cpp

  File used for setting up the scale.

Chapter 6

Class Documentation

6.1 inventory Struct Reference

Represents a user's beverage inventory.

`#include <inventory.h>`

Public Attributes

- products_stocked products_in_inventory [INVENTORY_CAPACITY]
  Products currently in inventory.
- uint8_t number_of_products_stocked
  Number of valid entries in products_in_inventory.
- uint8_t room_number
  Room number associated with the inventory (currently unused).

6.1.1 Member Data Documentation

6.1.1.1 number_of_products_stocked

```
uint8_t inventory::number_of_products_stocked
```

6.1.1.2 products_in_inventory

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

6.1.1.3 room_number

```
uint8_t inventory::room_number
```

FRID of the user owning the inventory (not implemented)

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

- include/inventory.h

6.2 product Struct Reference

Describes a single beverage product.

`#include <inventory.h>`

Public Attributes

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

6.2.1 Member Data Documentation

6.2.1.1 beverage_variant

`beverage_type product::beverage_variant`

6.2.1.2 name

`char product::name[20]`

6.2.1.3 price

`uint8_t product::price`

6.2.1.4 weight

`uint16_t product::weight`

grams)

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

- include/inventory.h

6.3 products_stocked Struct Reference

Tracks inventory quantities for a single product.

`#include <inventory.h>`

Public Attributes

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

6.3.1 Member Data Documentation

6.3.1.1 beverage

```
product products_stocked::beverage
```

6.3.1.2 current_quantity

```
uint8_t products_stocked::current_quantity
```

6.3.1.3 original_quantity

```
uint8_t products_stocked::original_quantity
```

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

- include/inventory.h

6.4 User Struct Reference

User record stored in the RFID user database.

`#include <rfid_access.h>`

Public Attributes

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

6.4.1 Member Data Documentation

6.4.1.1 balance

```
int User::balance
```

6.4.1.2 roomNumber

```
int User::roomNumber
```

6.4.1.3 uid

```
byte User::uid[UID_LENGTH]
```

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

- include/rfid_access.h

Chapter 7

File Documentation

7.1 include/admin_html.h File Reference

HTML for the admin page.

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

Variables

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

7.1.1 Detailed Description

Authors

Baldur G. Toftegaard

7.1.2 Variable Documentation

7.1.2.1 PROGMEM

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

Initial value:

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

Opening container for the sales graph.

Interperated by compiler as a string.

7.2 admin_html.h

Go to the documentation of this file.

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

7.3 include/buzzer.h File Reference

Declaring functions to play sounds using a buzzer.

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

Macros

- #define BUZZER_H

Functions

- void play_warning (unsigned long t)

  Plays short sound when door is closed and about to be locked.
- void play_unlock ()

  Plays an ascending tone, when the door unlocks.
- void play_lock ()

  Play a descending tone, when the door locks.

7.3.1 Detailed Description

Implementation of buzzer sound effects.

Author

Anssi Sohlman

7.3.2 Macro Definition Documentation

7.3.2.1 BUZZER_H

```
#define BUZZER_H
```

7.3.3 Function Documentation

7.3.3.1 play_lock()

```
void play_lock ()
```

7.3.3.2 play_unlock()

```
void play_unlock ()
```

7.3.3.3 play_warning()

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

Parameters

t	Time passed since door has been closed

7.4 buzzer.h

Go to the documentation of this file.

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

7.5 include/fridge_state.h File Reference

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

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

Variables

- inventory fridge

  Used to set up the fridge inventory.

7.5.1 Detailed Description

Author

Baldur G. Toftegaard

7.5.2 Variable Documentation

7.5.2.1 fridge

```
inventory fridge  [extern]
```

7.6 fridge_state.h

Go to the documentation of this file.

```
00001
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

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

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

Macros

- #define ROOM_COUNT 18

  Number of rooms supported by the sales graph.

Functions

- void graph_add_to_room_green (uint8_t roomNumber, int delta)

  Adds a value to the green sales bar of a given room.
- void graph_add_to_room_clasic (uint8_t roomNumber, int delta)

  Adds a value to the classic sales bar of a given room.
- void print_graph_arrays ()

  Prints the current graph height arrays.

Variables

- int greenHeight [ROOM_COUNT]

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

  Height values for classic product sales per room.

7.7.1 Detailed Description

Author

Baldur G. Toftegaard

7.7.2 Macro Definition Documentation

7.7.2.1 ROOM_COUNT

```
#define ROOM_COUNT 18
```

7.7.3 Function Documentation

7.7.3.1 graph_add_to_room_clasic()

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

Parameters

roomNumber	Room operated on.
delta	Incremental value used to update the bar height.

7.7.3.2 graph_add_to_room_green()

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

Parameters

roomNumber	Room operated on.
delta	Incremental value used to update the bar height.

7.7.3.3 print_graph_arrays()

```
void print_graph_arrays ()
```

7.7.4 Variable Documentation

7.7.4.1 classicHeight

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

Height values for classic product sales per room.

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

7.7.4.2 greenHeight

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

Height values for green product sales per room.

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

7.8 graph_data.h

Go to the documentation of this file.

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

7.9 include/index_html.h File Reference

HTML for the start page.

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

Variables

- const char INDEX_HTML_HEAD[ ] PROGMEM

  Sting used for the HTML header, read by the compiler as a string.

7.9.1 Detailed Description

Author

Baldur G. Toftegaard

7.9.2 Variable Documentation

7.9.2.1 PROGMEM

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

Initial value:

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

Opening container for the sales graph.

String used for the HTML footer.

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

7.10 index_html.h

Go to the documentation of this file.

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

```
00028 )rawliteral";
00029
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 initializing users and products.
- void perform_sale (inventory *fridge_inventory)

  Performs a sale between a user and the fridge.

7.11.1 Detailed Description

Author

Baldur G. Toftegaard

7.11.2 Macro Definition Documentation

7.11.2.1 number_of_users

```
#define number_of_users 18
```

7.11.3 Function Documentation

7.11.3.1 init_users_and_products()

```
void init_users_and_products ()
```

Old function for initializing users and products.

7.11.3.2 perform_sale()

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

Parameters

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

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

  Describes a single beverage product.
- struct products_stocked

  Tracks inventory quantities for a single product.
- struct inventory

  Represents a user's beverage inventory.

Macros

- #define INVENTORY_CAPACITY 6

  Maximum number of products the inventory can hold.

Enumerations

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

  Types of beverages supported by the system.

Functions

- void inventory_init (inventory *inventory)

  Initializes an inventory structure.
- product inventory_make_product (const char *name, beverage_type type, uint16_t weight, uint8_t price)

  Creates a new product instance.
- bool inventory_add_product (inventory *inventory, product product, uint16_t quantity)

  Adds a new product to the inventory.
- bool inventory_remove_product (inventory *inventory, product beverage)

  Removes a product entirely from the inventory.
- bool inventory_add_beverage (inventory *inventory, product beverage, uint16_t amount)

  Increases the quantity of a beverage in the inventory.
- bool inventory_remove_beverage (inventory *inventory, product beverage, uint8_t amount)

  Function for removing from the amount of a beverage in an inventory.
- void inventory_print (inventory *inventory)

  Function to print a users inventory.

7.13.1 Detailed Description

Author

Baldur G. Toftegaard

7.13.2 Macro Definition Documentation

7.13.2.1 INVENTORY_CAPACITY

```
#define INVENTORY_CAPACITY 6
```

7.13.3 Enumeration Type Documentation

7.13.3.1 beverage_type

```
enum beverage_type
```

Enumerator

beer	Beer.

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

7.13.4 Function Documentation

7.13.4.1 inventory_add_beverage()

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

Parameters

inventory	Inventory to modify.
beverage	Beverage to update.
amount	Quantity to add.

Returns

true - Quantity updated successfully.

false - Product not found or overflow.

7.13.4.2 inventory_add_product()

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

Parameters

inventory	Inventory to add the product to.
product	Product to add.
quantity	Initial quantity of the product.

Returns

true - Product successfully added.

false - Error occurred (inventory full or duplicate product).

7.13.4.3 inventory_init()

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

Clears internal state and prepares the inventory for use.

Parameters

inventory	Pointer to inventory instance to initialize.

7.13.4.4 inventory_make_product()

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

Parameters

name	Display name of the product.
type	Beverage type.
weight	Weight of the product.
price	Price of the product.

Returns

Initialized product instance.

7.13.4.5 inventory_print()

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

Parameters

inventory	Inventory to print.

7.13.4.6 inventory_remove_beverage()

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

Parameters

inventory	Inventory to modify.

beverage	Beverage to update.
amount	Quantity to remove.

Returns

true - Quantity updated successfully.

false - Product not found or insufficient stock.

7.13.4.7 inventory_remove_product()

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

Parameters

inventory	Inventory to remove the product from.
beverage	Product to remove.

Returns

true - Product removed successfully.

false - Product not found.

7.14 inventory.h

Go to the documentation of this file.

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

```
00052 } products_stocked;
00053
00054
00059 typedef struct {
00060     products_stocked products_in_inventory[INVENTORY_CAPACITY];
00061     uint8_t number_of_products_stocked;
00062     uint8_t room_number;
00064 } inventory;
00065
00066
00074 void inventory_init(
00075     inventory *inventory
00076 );
00077
00088 product inventory_make_product(
00089     const char *name,
00090     beverage_type type,
00091     uint16_t weight,
00092     uint8_t price
00093 );
00094
00105 bool inventory_add_product(
00106     inventory *inventory,
00107     product product,
00108     uint16_t quantity
00109 );
00110
00111
00121 bool inventory_remove_product(
00122     inventory *inventory,
00123     product beverage
00124 );
00125
00126
00137 bool inventory_add_beverage(
00138     inventory *inventory,
00139     product beverage,
00140     uint16_t amount
00141 );
00142
00153 bool inventory_remove_beverage(
00154     inventory *inventory,
00155     product beverage,
00156     uint8_t amount
00157 );
00158
00164 void inventory_print(
00165     inventory *inventory
00166 );
00167
00168 #endif
```

7.15 include/lock_ctrl.h File Reference

Door lock control interface using a servo and light sensor.

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

Macros

- #define SERVO_PIN 16

  GPIO pin connected to the servo motor.
- #define CLOSED_THRESHOLD 70

  Light threshold indicating a closed box.
- #define LIGHT_PIN A0

  GPIO pin connected to light detector.
- #define OPEN_THRESHOLD 100

  Light threshold indicating an open box.

Functions

- void lock_ctrl_init ()

  Initializes the lock control module.
- void lock_door ()

  Locks the door using the servo.
- void unlock_door ()

  Unlocks the door using the servo.
- bool is_box_closed ()

  Checks whether the box is closed.
- void play_open ()

  Plays the sound effect for door opening.
- void play_close ()

  Play sound effect when the door closes.

7.15.1 Detailed Description

Author

Amal Araweelo Almis

7.15.2 Macro Definition Documentation

7.15.2.1 CLOSED_THRESHOLD

```
#define CLOSED_THRESHOLD 70
```

7.15.2.2 LIGHT_PIN

```
#define LIGHT_PIN A0
```

7.15.2.3 OPEN_THRESHOLD

```
#define OPEN_THRESHOLD 100
```

7.15.2.4 SERVO_PIN

```
#define SERVO_PIN 16
```

7.15.3 Function Documentation

7.15.3.1 is_box_closed()

```
bool is_box_closed ()
```

Uses the light sensor to determine door state.

Returns

true Box is closed.

false Box is open.

7.15.3.2 lock_ctrl_init()

```
void lock_ctrl_init ()
```

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

7.15.3.3 lock_door()

```
void lock_door ()
```

7.15.3.4 play_close()

```
void play_close ()
```

7.15.3.5 play_open()

```
void play_open ()
```

7.15.3.6 unlock_door()

```
void unlock_door ()
```

7.16 lock_ctrl.h

Go to the documentation of this file.

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

7.17 include/login_html.h File Reference

HTML file for the login page.

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

Variables

- const char LOGIN_HTML[ ] PROGMEM
  HTML for the login page.

7.17.1 Detailed Description

Authors

Baldur G. Toftegaard

7.17.2 Variable Documentation

7.17.2.1 PROGMEM

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

Opening container for the sales graph.

Interpreted as a string by the compiler.

7.18 login_html.h

Go to the documentation of this file.

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

7.19 include/rfid_access.h File Reference

Declare functions related to RFID reading and user database.

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

Classes

- struct User

  User record stored in the RFID user database.

Macros

- #define SS_PIN 15

  GPIO pin connected to SS.
- #define RST_PIN 0

  GPIO pin connected to RST.
- #define MAX_ROOMS 17

  Maximum number of rooms supported by the system.
- #define UID_LENGTH 4

  Length of an RFID UID in bytes.

Enumerations

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

  Supported serial commands for the RFID management interface.

Functions

- RFIDcommand check_command (void)

  brief Reads a command from the serial interface and maps it to an RFIDcommand.
- void setup_RFID_reader (MFRC522 &rfid)

  Initializes SPI and the MFRC522 RFID reader.
- bool add_user (MFRC522 &rfid)

  Adds a new user by reading room number and scanning an RFID tag.
- bool remove_user ()

  Removes a user from the database.
- bool compare_UID (byte ∗uid1, byte ∗uid2)

  Compares two RFID UIDs.
- bool read_RFID_tag (MFRC522 &rfid, byte ∗uidBuffer)

  Reads an RFID tag UID from the MFRC522 reader.
- void display_commands (void)

  Prints the available serial commands.
- void display_commands_um ()

  Prints the available serial commands for user-management mode.
- void get_users_db (User ∗ptr)

  Copies the current user database to a provided buffer.
- void user_management (RFIDcommand cmd, User ∗ptr, MFRC522 &rfid)

  Executes user-management actions based on the provided command.
- bool validate_rfid (MFRC522 myRFID)

  Validates an RFID tag against the registered user database.
- void print_single_user (User ∗ptr, int idx)

  brief Prints a single user entry to the serial interface.
- void print_all_users (User ∗ptr)

  Prints all users in the database to the serial interface.
- void print_uid (byte ∗ptr)

  Prints a UID buffer to the serial interface.
- int read_integer ()

  Reads an integer from the serial interface.
- bool read_confirmation ()

  Reads a confirmation input from the serial interface.
- int find_empty_index (User ∗ptr)

  brief Finds an empty slot in the user database.
- int count_rooms (User ∗ptr)

  brief Counts the number of occupied user entries in the database.
- void rfid_set_last_uid (const byte ∗uidIn)

  Function for storing the last used RFID.
- bool rfid_get_last_uid (byte ∗uidOut)

  Function for restoring the last used RFID.

Variables

- User users [MAX_ROOMS]

  Global user database array.
- int userCount

  Number of currently registered users.

7.19.1 Detailed Description

Author

Amal Araweelo Almis

Anssi Sohlman

Baldur G. Toftegaard

7.19.2 Macro Definition Documentation

7.19.2.1 MAX_ROOMS

```
#define MAX_ROOMS 17
```

7.19.2.2 RST_PIN

```
#define RST_PIN 0
```

7.19.2.3 SS_PIN

```
#define SS_PIN 15
```

7.19.2.4 UID_LENGTH

```
#define UID_LENGTH 4
```

7.19.3 Enumeration Type Documentation

7.19.3.1 RFIDcommand

```
enum RFIDcommand
```

Enumerator

CMD_NONE	

CMD_ADD_USER	
CMD_OPEN	
CMD_LOCK	
CMD_REMOVE_USER	
CMD_CONFIRM	
CMD_PRINT	

7.19.4 Function Documentation

7.19.4.1 add_user()

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

Parameters

rfid	An instance of a MFRC522 object.

Returns

true User was added.

false User was not added.

7.19.4.2 check_command()

```
RFIDcommand check_command (
            void )
```

Returns

RFIDcommand

7.19.4.3 compare_UID()

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

Parameters

uid1	The first uid number.
uid2	The uid number we want to dompare it to.

Returns

true Numbers match.

false NUmbers do not match.

7.19.4.4 count_rooms()

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

Parameters

ptr	A pointer to the users[] array

Returns

int Number of rooms

7.19.4.5 display_commands()

```
void display_commands (
            void )
```

7.19.4.6 display_commands_um()

```
void display_commands_um ()
```

7.19.4.7 find_empty_index()

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

Parameters

ptr	A pointer to the users[] array

Returns

int The empty index.

7.19.4.8 get_users_db()

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

Parameters

ptr	A pointer to the users[] array

7.19.4.9 print_all_users()

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

Parameters

ptr	A pointer to the users[] array

7.19.4.10 print_single_user()

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

Parameters

ptr	A pointer to the users[] array
idx	The index to be printed

7.19.4.11 print_uid()

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

Parameters

ptr	A pointer to member of the users[] array

7.19.4.12 read_confirmation()

```
bool read_confirmation ()
```

Returns

true Confirmed.

false Not confirmed.

7.19.4.13 read_integer()

```
int read_integer ()
```

Returns

int Intiger read from serial interface.

7.19.4.14 read_RFID_tag()

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

Parameters

rfid	An instance of a MFRC522 object

uidBuffer	An array where the UID is saved to

Returns

true RFID tag was read.

false No RFID tag was read.

7.19.4.15 remove_user()

```
bool remove_user ()
```

Returns

true User was removed.

false User was not removed.

7.19.4.16 rfid_get_last_uid()

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

Parameters

uidOut	Pointer to user it.

Returns

true There was an last uid to retrive.

false No last uid was retrived.

7.19.4.17 rfid_set_last_uid()

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

Parameters

uidOut	Pointer to user it.

Returns

true Stored last used uid.

false Last uid was not stored

7.19.4.18 setup_RFID_reader()

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

Parameters

rfid	An instance of a MFRC522 object

7.19.4.19 user_management()

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

Parameters

cmd	The current RFIDcommand (add or remove user)
ptr	A pointer to the users[] array
rfid	An instance of a MFRC522 object

7.19.4.20 validate_rfid()

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

Parameters

myRFID	An instance of a MFRC522 object

Returns

true The RFID was a match.

false There was no matching RFID in the database.

7.19.5 Variable Documentation

7.19.5.1 userCount

```
int userCount  [extern]
```

7.19.5.2 users

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

7.20 rfid_access.h

Go to the documentation of this file.

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

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

7.21 include/sale_html.h File Reference

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

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

Functions

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

  Sends a single sales bar element to the client.
- void send_sale_html_page (ESP8266WebServer &server, uint8_t room_count, const int *greenHeight, const int *classicHeights)

  Sends the complete sales graph page.

Variables

- const char SALE_BOX_START[ ] PROGMEM

  Opening container for the sales graph.

7.21.1 Detailed Description

Author

Baldur G. Toftegaard

7.21.2 Function Documentation

7.21.2.1 send_sale_html_graph()

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

Parameters

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

7.21.2.2 send_sale_html_page()

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

Parameters

server	The server instance.
room_count	Number of registered rooms.
greenHeight	The height of green bars.
classicHeights	The height of clasic bars.

7.21.3 Variable Documentation

7.21.3.1 PROGMEM

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

Closing container for the complete sales graph.

Closing container for a single room graph.

Closing fragment for a single sales bar.

HTML fragment defining the height style of a bar.

HTML fragment defining the CSS class type for a bar.

HTML fragment for the room identifier.

Opening container for a single room graph.

Opening container for the sales graph.

Interperated by compiler as a string.

Opening container for the sales graph.

String used for the HTML footer.

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

Opening container for the sales graph.

Interpreted as a string by the compiler.

Opening container for the sales graph.

is handled like a string by the compiler

7.22 sale_html.h

Go to the documentation of this file.

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

7.23 include/style_css.h File Reference

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

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

Variables

- const char STYLE_CSS[ ] PROGMEM
  String containing the CSS styling of the webserver.

7.23.1 Detailed Description

The css is handled as a string by the compiler.

Author

Baldur G. Toftegaard

7.23.2 Variable Documentation

7.23.2.1 PROGMEM

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

Opening container for the sales graph.

is handled like a string by the compiler

7.24 style_css.h

Go to the documentation of this file.

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

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

7.25 include/weight_scale.h File Reference

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

Macros

- #define BEER_WEIGHT 350

  Define the weight of a beer.

- #define SCALE_TOL 25

  Define the uncertainty of the weight mesurment.

- #define HX711_DOUT 4
- #define HX711_SCK 5
- #define SCALE_DEFAULT_SETTLE_TIME_MS 3000

Functions

- float get_weight_reference (void)

  Function to get the weight reference.
- void set_weight_reference (float value)

  Function to set the weight reference.
- void reset_weight_reference (void)

  Function to reset the weight reference.
- bool weight_reference_is_set (void)

  Function to send confirmation that the weight reference is set.
- void setup_scale (float calFactor)

  Function to seting up the scale, is called in the begining of the program.
- bool update_scale (void)

  Function for updating the scale value.
- float get_weight (void)

  Function to get the scale reading.
- void tare_scale (void)

  Function to tare the scale.
- bool tare_complete (void)

  Function to signal that the scale has been tared.
- int get_beer_cans_taken (float referencWeight, float currentWeight)

  Function to get the number of beer cans taken.

Variables

- HX711_ADC scale

7.25.1 Detailed Description

Author

Amal Araweelo Almis

7.25.2 Macro Definition Documentation

7.25.2.1 BEER_WEIGHT

```
#define BEER_WEIGHT 350
```

7.25.2.2 HX711_DOUT

```
#define HX711_DOUT 4
```

7.25.2.3 HX711_SCK

```
#define HX711_SCK 5
```

7.25.2.4 SCALE_DEFAULT_SETTLE_TIME_MS

```
#define SCALE_DEFAULT_SETTLE_TIME_MS 3000
```

7.25.2.5 SCALE_TOL

```
#define SCALE_TOL 25
```

7.25.3 Function Documentation

7.25.3.1 get_beer_cans_taken()

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

Parameters

referencWeight	The set reference weight [g]
currentWeight	The current mesured weight [g]

Returns

int number of cans taken

7.25.3.2 get_weight()

```
float get_weight (
            void )
```

Returns

float Value read from weight.

7.25.3.3 get_weight_reference()

```
float get_weight_reference (
            void )
```

Returns

float The weight reference.

7.25.3.4 reset_weight_reference()

```
void reset_weight_reference (
            void )
```

7.25.3.5 set_weight_reference()

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

Parameters

value	Reference value.

7.25.3.6 setup_scale()

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

Parameters

calFactor	The calibration factor.

7.25.3.7 tare_complete()

```
bool tare_complete (
            void )
```

Returns

true The scale was tared.

false The scale was not tared.

7.25.3.8 tare_scale()

```
void tare_scale (
            void )
```

7.25.3.9 update_scale()

```
bool update_scale (
            void )
```

Returns

true Scale was updated.

false Scale was not updated.

7.25.3.10 weight_reference_is_set()

```
bool weight_reference_is_set (
            void )
```

Returns

true Weight reference is set.

false Weight reference is not set.

7.25.4 Variable Documentation

7.25.4.1 scale

```
HX711_ADC scale  [extern]
```

7.26 weight_scale.h

Go to the documentation of this file.

```
00001
00005
00006 #ifndef WEIGHT_SCALE_H
00007 #define WEIGHT_SCALE_H
00008
00009 #include <Arduino.h>
00010 #include <HX711_ADC.h>
00011 #include <math.h>
00012
00018 float get_weight_reference(
00019     void
00020 );
00021
00027 void  set_weight_reference(
00028     float value
00029 );
00030
00034 void  reset_weight_reference(
00035     void
00036 );
00037
00044 bool  weight_reference_is_set(
00045     void
00046 );
00047
00051 #define BEER_WEIGHT 350
00052
00056 #define SCALE_TOL 25
00057
00058 // Pins
00059 #define HX711_DOUT 4   // GPIO4=D2
00060 #define HX711_SCK  5   // GPIO5=D1
00061
00062 // Config scale
00063 #define SCALE_DEFAULT_SETTLE_TIME_MS 3000
00064
00065 // Globals
00066 extern HX711_ADC scale;
00067
00068
00074 void setup_scale(
00075     float calFactor
00076 );
00077
00084 bool update_scale(
00085     void
00086 );
00087
00093 float get_weight(
```

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

7.27 README.md File Reference

7.28 src/buzzer.cpp File Reference

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

Functions

- void play_warning (unsigned long t)

  Plays short sound when door is closed and about to be locked.
- void play_unlock ()

  Plays an ascending tone, when the door unlocks.
- void play_lock ()

  Play a descending tone, when the door locks.

Variables

- const int BUZZERPIN = 2

  GPIO pin connected to the buzzer.
- const double HIGH_TONE = 1000

  Frequency (Hz) used for high-tone buzzer sound.
- const double LOW_TONE = 600

  Frequency (Hz) used for low-tone buzzer sound.
- const unsigned long TONE_LENGTH = 200

  Duration of buzzer tone in milliseconds.

7.28.1 Function Documentation

7.28.1.1 play_lock()

```
void play_lock ()
```

7.28.1.2 play_unlock()

```
void play_unlock ()
```

7.28.1.3 play_warning()

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

Parameters

t	Time passed since door has been closed

7.28.2 Variable Documentation

7.28.2.1 BUZZERPIN

```
const int BUZZERPIN = 2
```

7.28.2.2 HIGH_TONE

```
const double HIGH_TONE = 1000
```

7.28.2.3 LOW_TONE

```
const double LOW_TONE = 600
```

7.28.2.4 TONE_LENGTH

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

7.29 src/database_management.cpp File Reference

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

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

Functions

- void user_management (RFIDcommand incomingCommand, User *ptr, MFRC522 &rfid)

  Executes user-management actions based on the provided command.
- bool remove_user ()

  Removes a user from the database.
- void get_users_db (User *ptr)

  Copies the current user database to a provided buffer.
- void print_all_users (User *ptr)

  Prints all users in the database to the serial interface.
- void print_single_user (User *ptr, int idx)

  brief Prints a single user entry to the serial interface.
- void print_uid (byte *ptr)

  Prints a UID buffer to the serial interface.
- int read_integer ()

  Reads an integer from the serial interface.
- bool read_confirmation ()

  Reads a confirmation input from the serial interface.
- int find_empty_index (User *ptr)

  brief Finds an empty slot in the user database.
- int count_rooms (User *ptr)

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

7.29.1 Detailed Description

Authors

Anssi Sohlman,

7.29.2 Function Documentation

7.29.2.1 count_rooms()

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

Parameters

ptr	A pointer to the users[] array

Returns

int Number of rooms

7.29.2.2 find_empty_index()

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

Parameters

ptr	A pointer to the users[] array

Returns

int The empty index.

7.29.2.3 get_users_db()

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

Parameters

ptr	A pointer to the users[] array

7.29.2.4 print_all_users()

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

Parameters

ptr	A pointer to the users[] array

7.29.2.5 print_single_user()

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

Parameters

ptr	A pointer to the users[] array
idx	The index to be printed

7.29.2.6 print_uid()

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

Parameters

ptr	A pointer to member of the users[] array

7.29.2.7 read_confirmation()

```
bool read_confirmation ()
```

Returns

true Confirmed.

false Not confirmed.

7.29.2.8 read_integer()

```
int read_integer ()
```

Returns

int Intiger read from serial interface.

7.29.2.9 remove_user()

```
bool remove_user ()
```

Returns

true User was removed.

false User was not removed.

7.29.2.10 user_management()

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

Parameters

cmd	The current RFIDcommand (add or remove user)

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

7.30 src/fridge_state.cpp File Reference

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

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

Variables

- inventory fridge

  Used to set up the fridge inventory.

7.30.1 Detailed Description

Author

Baldur G. Toftegaard

7.30.2 Variable Documentation

7.30.2.1 fridge

```
inventory fridge
```

7.31 src/graph_data.cpp File Reference

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

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

Functions

- void graph_add_to_room_green (uint8_t roomNumber, int delta)

  Adds a value to the green sales bar of a given room.
- void graph_add_to_room_clasic (uint8_t roomNumber, int delta)

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

Variables

- int greenHeight [ROOM_COUNT]

  Sales graph height data for green beer.
- int classicHeight [ROOM_COUNT]

  Sales graph height data for classic beer.

7.31.1 Detailed Description

Author

Baldur G. Toftegaard

7.31.2 Function Documentation

7.31.2.1 graph_add_to_room_clasic()

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

Parameters

roomNumber	Room operated on.
delta	Incremental value used to update the bar height.

7.31.2.2 graph_add_to_room_green()

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

Parameters

roomNumber	Room operated on.
delta	Incremental value used to update the bar height.

7.31.3 Variable Documentation

7.31.3.1 classicHeight

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

Height values for classic product sales per room.

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

7.31.3.2 greenHeight

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

Height values for green product sales per room.

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

7.32 src/init_users_and_sale.cpp File Reference

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

Functions

- void init_users_and_products ()

  Not used in prototype, shuld be moved to inventory.cpp.
- static float read_current_weight_blocking (uint32_t timeoutMs=1200)
- void perform_sale (inventory *fridge_inventory)

  Performs a sale between a user and the fridge.

7.32.1 Function Documentation

7.32.1.1 init_users_and_products()

```
void init_users_and_products ()
```

Old function for initializing users and products.

7.32.1.2 perform_sale()

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

Parameters

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

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)

  Initializes an inventory structure.
- product inventory_make_product (const char *product_name, beverage_type type, uint16_t weight, uint8_t price)

  Creates a new product instance.
- bool inventory_add_product (inventory *inventory, product product, uint16_t quantity)

  Adds a new product to the inventory.
- bool inventory_remove_product (inventory *inventory, product beverage)

  Removes a product entirely from the inventory.
- bool inventory_add_beverage (inventory *inventory, product beverag, uint8_t amount)
- bool inventory_remove_beverage (inventory *inventory, product beverag, uint8_t amount)

  Function for removing from the amount of a beverage in an inventory.
- void inventory_print (inventory *inventory)

  Function to print a users inventory.

7.33.1 Detailed Description

Author

Baldur G. Toftegaard

7.33.2 Function Documentation

7.33.2.1 inventory_add_beverage()

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

7.33.2.2 inventory_add_product()

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

Parameters

inventory	Inventory to add the product to.

product	Product to add.
quantity	Initial quantity of the product.

Returns

true - Product successfully added.

false - Error occurred (inventory full or duplicate product).

7.33.2.3 inventory_init()

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

Clears internal state and prepares the inventory for use.

Parameters

inventory	Pointer to inventory instance to initialize.

7.33.2.4 inventory_make_product()

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

Parameters

name	Display name of the product.
type	Beverage type.
weight	Weight of the product.
price	Price of the product.

Returns

Initialized product instance.

7.33.2.5 inventory_print()

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

Parameters

inventory	Inventory to print.

7.33.2.6 inventory_remove_beverage()

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

Parameters

inventory	Inventory to modify.
beverage	Beverage to update.
amount	Quantity to remove.

Returns

true - Quantity updated successfully.

false - Product not found or insufficient stock.

7.33.2.7 inventory_remove_product()

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

Parameters

inventory	Inventory to remove the product from.
beverage	Product to remove.

Returns

true - Product removed successfully.

false - Product not found.

7.34 src/lock_ctrl.cpp File Reference

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

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

Functions

- void lock_ctrl_init ()

  Initializes the lock control module.
- void lock_door ()

  Locks the door using the servo.
- void unlock_door ()

  Unlocks the door using the servo.
- bool is_box_closed ()

  Checks whether the box is closed.
- void play_open ()

  Plays the sound effect for door opening.
- void play_close ()

  Play sound effect when the door closes.

Variables

- static Servo lockServo
- static const int UNLOCK_POS = 0

  Servo position corresponding to the unlocked state (mechanically fixed).
- static const int LOCK_POS = 100

  Servo position corresponding to the locked state.
- const int BUZZER = 2

  GPIO pin connected to the buzzer.
- const double HIGH_TONE = 1000

  Frequency (Hz) used for the high-tone buzzer sound.
- const double LOW_TONE = 600

  Frequency (Hz) used for the low-tone buzzer sound.
- const unsigned long TONE_LENGTH = 200

  Duration of buzzer tone in milliseconds.
- static bool boxClosed = false

  Internal latched state indicating whether the box is closed.

7.34.1 Detailed Description

Authors

Amal Araweelo Almis

7.34.2 Function Documentation

7.34.2.1 is_box_closed()

```
bool is_box_closed ()
```

Uses the light sensor to determine door state.

Returns

true Box is closed.

false Box is open.

7.34.2.2 lock_ctrl_init()

```
void lock_ctrl_init ()
```

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

7.34.2.3 lock_door()

```
void lock_door ()
```

7.34.2.4 play_close()

```
void play_close ()
```

7.34.2.5 play_open()

```
void play_open ()
```

7.34.2.6 unlock_door()

```
void unlock_door ()
```

7.34.3 Variable Documentation

7.34.3.1 boxClosed

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

7.34.3.2 BUZZER

```
const int BUZZER = 2
```

Note

This pin choice may cause reset issues on some boards.

7.34.3.3 HIGH_TONE

```
const double HIGH_TONE = 1000
```

7.34.3.4 LOCK_POS

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

7.34.3.5 lockServo

```
Servo lockServo  [static]
```

7.34.3.6 LOW_TONE

```
const double LOW_TONE = 600
```

7.34.3.7 TONE_LENGTH

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

7.34.3.8 UNLOCK_POS

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

7.35 src/main.cpp File Reference

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

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

Functions

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

Variables

- static const float CAL_FACTOR = 22.9f

  Calibration factor used for weight or sensor calculations.
- static const uint16_t START_BEER_QTY = 20

  Initial quantity of beer available at system startup.
- const char * WIFI_SSID = "Baldur's A56"

  Wi-Fi network SSID used by the device.
- const char * WIFI_PASS = "MyPasskeyA56"

  Wi-Fi network password used by the device.
- int greenHeight [ROOM_COUNT]

  Sales graph height data for green beer.
- int classicHeight [ROOM_COUNT]

  Sales graph height data for classic beer.
- product demo_beer
- RFIDcommand activeCommand = CMD_NONE
- bool doorUnlocked = false
- unsigned long doorCloseTimer = 0

7.35.1 Detailed Description

Author

Baldur G. Toftegaard

Anssi Sohlman

7.35.2 Function Documentation

7.35.2.1 connect_wifi_and_start_mdns()

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

7.35.2.2 loop()

```
void loop ()
```

7.35.2.3 rfid()

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

7.35.2.4 server()

```
ESP8266WebServer server (
            80 )
```

7.35.2.5 setup()

```
void setup ()
```

7.35.2.6 setup_inventory_and_scale()

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

7.35.2.7 setup_rfid_and_lock()

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

7.35.2.8 setup_web_routes()

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

7.35.3 Variable Documentation

7.35.3.1 activeCommand

```
RFIDcommand activeCommand = CMD_NONE
```

7.35.3.2 CAL_FACTOR

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

7.35.3.3 classicHeight

```
int classicHeight[ROOM_COUNT]
```

Initial value:

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

Height values for classic product sales per room.

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

7.35.3.4 demo_beer

```
product demo_beer
```

7.35.3.5 doorCloseTimer

```
unsigned long doorCloseTimer = 0
```

7.35.3.6 doorUnlocked

```
bool doorUnlocked = false
```

7.35.3.7 greenHeight

```
int greenHeight[ROOM_COUNT]
```

Initial value:

```
                = {
    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.

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

7.35.3.8 START_BEER_QTY

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

7.35.3.9 WIFI_PASS

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

7.35.3.10 WIFI_SSID

```
const char* WIFI_SSID = "Baldur's A56"
```

7.36 src/rfid_access.cpp File Reference

File used to setup the RFID.

```
#include "rfid_access.h"
```

Functions

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

Variables

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

7.36.1 Detailed Description

Author

Amal Araweelo Almis

Baldur G. Toftegaard

7.36.2 Function Documentation

7.36.2.1 add_user()

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

Parameters

rfid	An instance of a MFRC522 object.

Returns

true User was added.

false User was not added.

7.36.2.2 check_command()

```
RFIDcommand check_command (
            void )
```

Returns

RFIDcommand

7.36.2.3 compare_UID()

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

Parameters

uid1	The first uid number.
uid2	The uid number we want to dompare it to.

Returns

true Numbers match.

false NUmbers do not match.

7.36.2.4 display_commands()

```
void display_commands (
            void )
```

7.36.2.5 display_commands_um()

```
void display_commands_um ()
```

7.36.2.6 read_RFID_tag()

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

Parameters

rfid	An instance of a MFRC522 object
uidBuffer	An array where the UID is saved to

Returns

true RFID tag was read.

false No RFID tag was read.

7.36.2.7 rfid_get_last_uid()

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

Parameters

uidOut	Pointer to user it.

Returns

true There was an last uid to retrive.

false No last uid was retrived.

7.36.2.8 rfid_set_last_uid()

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

Parameters

uidOut	Pointer to user it.

Returns

true Stored last used uid.

false Last uid was not stored

7.36.2.9 setup_RFID_reader()

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

Parameters

rfid	An instance of a MFRC522 object

7.36.2.10 validate_rfid()

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

Parameters

myRFID	An instance of a MFRC522 object

Returns

true The RFID was a match.

false There was no matching RFID in the database.

7.36.3 Variable Documentation

7.36.3.1 hasUID

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

7.36.3.2 lastUID

```
byte lastUID[UID_LENGTH]  [static]
```

7.36.3.3 userCount

```
int userCount = 0
```

7.36.3.4 users

```
User users[MAX_ROOMS]
```

7.37 src/sale_html.cpp File Reference

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

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

Functions

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

  Sends a single sales bar element to the client.
- void send_sale_html_page (ESP8266WebServer &server, uint8_t room_count, const int *greenHeight, const int *classicHeights)

  Sends the complete sales graph page.

Variables

- const char SALE_BOX_START[ ] PROGMEM = R"rawliteral( )rawliteral"

  Opening container for the sales graph.

7.37.1 Detailed Description

Author

Baldur G. Toftegaard

7.37.2 Function Documentation

7.37.2.1 send_sale_html_graph()

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

Parameters

server	The server instance.

room_number	The room number operated on.
bar_type	Type of the bar operated on.
bar_height	Height of the bre operated on.

7.37.2.2 send_sale_html_page()

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

Parameters

server	The server instance.
room_count	Number of registered rooms.
greenHeight	The height of green bars.
classicHeights	The height of clasic bars.

7.37.3 Variable Documentation

7.37.3.1 PROGMEM

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

Closing container for the complete sales graph.

Closing container for a single room graph.

Closing fragment for a single sales bar.

HTML fragment defining the height style of a bar.

HTML fragment defining the CSS class type for a bar.

HTML fragment for the room identifier.

Opening container for a single room graph.

Opening container for the sales graph.

Interperated by compiler as a string.

Opening container for the sales graph.

String used for the HTML footer.

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

Opening container for the sales graph.

Interpreted as a string by the compiler.

Opening container for the sales graph.

is handled like a string by the compiler

7.38 src/weight_scale.cpp File Reference

File used for setting up the scale.

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

Functions

- HX711_ADC scale (HX711_DOUT, HX711_SCK)
- float get_weight_reference (void)

  Function to get the weight reference.
- void set_weight_reference (float value)

  Function to set the weight reference.
- void reset_weight_reference (void)

  Function to reset the weight reference.
- bool weight_reference_is_set (void)

  Function to send confirmation that the weight reference is set.
- void setup_scale (float calFactor)

  Function to seting up the scale, is called in the begining of the program.
- bool update_scale ()

  Function for updating the scale value.
- float get_weight ()

  Function to get the scale reading.
- void tare_scale ()

  Function to tare the scale.
- bool tare_complete ()

  Function to signal that the scale has been tared.
- int get_beer_cans_taken (float referenceWeight, float currentWeight)

  Function to get the number of beer cans taken.

Variables

- static float g_referenceWeight = NAN

7.38.1 Detailed Description

Author

Amal Araweelo Almis

7.38.2 Function Documentation

7.38.2.1 get_beer_cans_taken()

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

Parameters

referencWeight	The set reference weight [g]

currentWeight	The current mesured weight [g]

Returns

int number of cans taken

7.38.2.2 get_weight()

```
float get_weight (
            void )
```

Returns

float Value read from weight.

7.38.2.3 get_weight_reference()

```
float get_weight_reference (
            void )
```

Returns

float The weight reference.

7.38.2.4 reset_weight_reference()

```
void reset_weight_reference (
            void )
```

7.38.2.5 scale()

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

7.38.2.6 set_weight_reference()

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

Parameters

value	Reference value.

7.38.2.7 setup_scale()

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

Parameters

calFactor	The calibration factor.

7.38.2.8 tare_complete()

```
bool tare_complete (
            void )
```

Returns

true The scale was tared.

false The scale was not tared.

7.38.2.9 tare_scale()

```
void tare_scale (
            void )
```

7.38.2.10 update_scale()

```
bool update_scale (
            void )
```

Returns

true Scale was updated.

false Scale was not updated.

7.38.2.11 weight_reference_is_set()

```
bool weight_reference_is_set (
            void )
```

Returns

true Weight reference is set.

false Weight reference is not set.

7.38.3 Variable Documentation

7.38.3.1 g_referenceWeight

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

Index