

Desk Greenhouse

v1.0

Generated by Doxygen 1.9.1

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 fan.c File Reference	3
2.1.1 Detailed Description	3
2.1.2 Function Documentation	4
2.1.2.1 fan_init()	4
2.1.2.2 fan_run()	4
2.1.2.3 PA15_Init()	4
2.2 fan.h File Reference	5
2.2.1 Detailed Description	5
2.2.2 Function Documentation	5
2.2.2.1 fan_init()	5
2.2.2.2 fan_run()	6
2.2.2.3 PA15_Init()	6
2.3 info.c File Reference	6
2.3.1 Detailed Description	7
2.3.2 Function Documentation	7
2.3.2.1 draw_info()	7
2.3.2.2 info()	8
2.4 info.h File Reference	8
2.4.1 Detailed Description	8
2.4.2 Function Documentation	8
2.4.2.1 info()	8
2.5 moisture.c File Reference	9
2.5.1 Detailed Description	10
2.5.2 Function Documentation	10
2.5.2.1 moisture()	10
2.5.3 Variable Documentation	10
2.5.3.1 AirValue	10
2.5.3.2 intervals	11
2.5.3.3 WaterValue	11
2.6 moisture.h File Reference	11
2.6.1 Detailed Description	11
2.6.2 Function Documentation	11
2.6.2.1 moisture()	11
2.6.3 Variable Documentation	12
2.6.3.1 AirValue	12
2.6.3.2 intervals	12
2.6.3.3 WaterValue	12
2.7 pump.c File Reference	12

2.7.1 Detailed Description	13
2.7.2 Function Documentation	13
2.7.2.1 pump_init()	13
2.7.2.2 pump_start()	13
2.8 pump.h File Reference	14
2.8.1 Detailed Description	14
2.8.2 Function Documentation	14
2.8.2.1 pump_init()	14
2.8.2.2 pump_start()	15
2.9 sensors.c File Reference	15
2.9.1 Detailed Description	17
2.9.2 Function Documentation	17
2.9.2.1 draw_sensor_box()	17
2.9.2.2 draw_sensors()	18
2.9.2.3 handler()	18
2.9.2.4 sensors()	18
2.9.2.5 test()	19
2.10 sensors.h File Reference	19
2.10.1 Detailed Description	19
2.10.2 Function Documentation	19
2.10.2.1 sensors()	19
2.11 Serial.c File Reference	20
2.11.1 Detailed Description	21
2.11.2 Function Documentation	21
2.11.2.1 EXTI9_5_IRQHandler()	21
2.11.2.2 SER_Busy()	21
2.11.2.3 SER_Init()	22
2.11.2.4 SER_Ready()	22
2.11.2.5 SER_Write()	23
2.11.3 Variable Documentation	23
2.11.3.1 input_buffer	23
2.11.3.2 output_buffer	23
2.11.3.3 sensor_num	23
2.12 Serial.h File Reference	23
2.12.1 Detailed Description	24
2.12.2 Function Documentation	24
2.12.2.1 SER_Busy()	24
2.12.2.2 SER_Init()	25
2.12.2.3 SER_Read()	25
2.12.2.4 SER_Ready()	26
2.12.2.5 SER_Write()	26
2.12.3 Variable Documentation	27

2.12.3.1 input_buffer	27
2.12.3.2 output_buffer	27
2.12.3.3 sensor_num	27
2.13 settings.h File Reference	27
2.13.1 Detailed Description	28
2.13.2 Function Documentation	28
2.13.2.1 settings()	28
2.13.3 Variable Documentation	29
2.13.3.1 max_humidity	29
2.13.3.2 max_temp	29
2.13.3.3 min_moisture_level	29
2.14 SystemClock.c File Reference	29
2.14.1 Detailed Description	29
2.14.2 Function Documentation	30
2.14.2.1 SystemClock_Config()	30
2.15 SystemClock.h File Reference	30
2.15.1 Detailed Description	30
2.15.2 Function Documentation	30
2.15.2.1 SystemClock_Config()	30
2.16 ui_elements.c File Reference	31
2.16.1 Detailed Description	32
2.16.2 Function Documentation	32
2.16.2.1 draw_background()	32
2.16.2.2 draw_check_option()	32
2.16.2.3 draw_flat_background()	33
2.16.2.4 draw_string_box()	33
2.16.2.5 draw_top_bar()	33
2.17 ui_elements.h File Reference	35
2.17.1 Detailed Description	35
2.17.2 Macro Definition Documentation	35
2.17.2.1 BACKGROUND_BLUE	36
2.17.2.2 HOME	36
2.17.3 Function Documentation	36
2.17.3.1 draw_check_option()	36
2.17.3.2 draw_string_box()	36
2.17.3.3 draw_top_bar()	37

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

fan.c	Fan control functions	3
fan.h	Fan initialisation and control	5
info.c	Information screen for the Desk Green House	6
info.h	Information screen for the Desk Green House	8
moisture.c	Moisture sensor related settings	9
moisture.h	Moisture sensor and values handler	11
pump.c	Pump control functions	12
pump.h	Pump initialisation and control	14
sensors.c	Sensor screen displayed to the user and output sensor handler	15
sensors.h	Sensors screen and outputs sensors handler	19
Serial.c	Moisture sensor related settings	20
Serial.h	Serial communication between the STM32 and the Arduino	23
settings.h	User defined settings used in sensors.c 's handler	27
SystemClock.c	Clock configuration for the stm32f746g-disco	29
SystemClock.h	Clock configuration for the stm32f746g-disco	30
ui_elements.c	User interface elements for creating the various user screens	31
ui_elements.h	Sensors screen and outputs sensors handler	35

Chapter 2

File Documentation

2.1 fan.c File Reference

Fan control functions.

```
#include "stm32f7xx_hal.h"
#include "SystemClock.h"
#include "ui_elements.h"
#include "fan.h"
```

Functions

- void [fan_init](#) (void)
Runs the necessary init functions and starts the PWM on the pin.
- void [fan_run](#) (char speed)
Runs the fan at the specified speed by setting CCR1.
- void [PA15_Init](#) (void)

Variables

- TIM_HandleTypeDef **htim2**
- TIM_OC_InitTypeDef **sConfigOC**

2.1.1 Detailed Description

Fan control functions.

Author

T. Buckingham

```
The file contains ::
+ Function to initialise the fan's PWM pin
+ Basic function to set the fans speed
```

2.1.2 Function Documentation

2.1.2.1 fan_init()

```
void fan_init (
    void )
```

Runs the necessary init functions and starts the PWM on the pin.

Parameters

Void.	
-------	--

Returns

Void.

2.1.2.2 fan_run()

```
void fan_run (
    char speed )
```

Runs the fan at the specified speed by setting CCR1.

Parameters

<i>speed</i>	The speed the fan is to run at, speeds are defined in fan.h
--------------	---

Returns

Void.

2.1.2.3 PA15_Init()

```
void PA15_Init (
    void )
```

Pin PA15 configuration (TIM2_CH1) and initialization. The only difference between the traditional GPIO initialization is the use of alternate function: 1) We need to specify the mode (gpio.Mode) as the alternate; 2) We need to map the right timer (or other peripheral) to the pin (gpio.Alternate).

2.2 fan.h File Reference

Fan initialisation and control.

Macros

- `#define SLOW 20` /* slow fan speed */
- `#define MED 15` /* medium fan speed */
- `#define FAST 10` /* fast fan speed */
- `#define F_FAST 5` /* very fast speed */
- `#define OFF 200` /* turns fan off :: -1 may also work */

Functions

- void `PA15_Init` (void)
- void `fan_init` (void)
Runs the necessary init functions and starts the PWM on the pin.
- void `fan_run` (char speed)
Runs the fan at the specified speed by setting CCR1.

2.2.1 Detailed Description

Fan initialisation and control.

Author

T. Buckingham

```
The file contains ::  
+ Fan speed variables
```

2.2.2 Function Documentation

2.2.2.1 fan_init()

```
void fan_init (  
    void )
```

Runs the necessary init functions and starts the PWM on the pin.

Parameters

Void.

Returns

Void.

2.2.2.2 fan_run()

```
void fan_run (
    char speed )
```

Runs the fan at the specified speed by setting CCR1.

Parameters

<i>speed</i>	The speed the fan is to run at, speeds are defined in fan.h
--------------	---

Returns

Void.

2.2.2.3 PA15_Init()

```
void PA15_Init (
    void )
```

Pin PA15 configuration (TIM2_CH1) and initialization. The only difference between the traditional GPIO initialization is the use of alternate function: 1) We need to specify the mode (gpio.Mode) as the alternate; 2) We need to map the right timer (or other peripheral) to the pin (gpio.Alternate).

2.3 info.c File Reference

Information screen for the Desk Green House.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "stm32f7xx_hal.h"
#include "GLCD_Config.h"
#include "Board_GLCD.h"
#include "cmsis_os.h"
#include "SystemClock.h"
#include "Board_Touch.h"
#include "ui_elements.h"
#include "settings.h"
#include "sensors.h"
#include "info.h"
```

Macros

- #define **wait_delay** HAL_Delay

Functions

- void **draw_info** (void)
Draws the 'help' screen to the user.
- int **info** (void)
The control function of the help screen.

Variables

- GLCD_FONT **GLCD_Font_6x8**
- GLCD_FONT **GLCD_Font_16x24**

2.3.1 Detailed Description

Information screen for the Desk Green House.

Author

T. Buckingham

```
The file contains ::  
+ Information text box displayed to the user on the 'help' screen
```

2.3.2 Function Documentation

2.3.2.1 draw_info()

```
void draw_info (  
    void )
```

Draws the 'help' screen to the user.

Parameters

Void.	
-------	--

Returns

Void.

2.3.2.2 info()

```
int info (
    void )
```

The control function of the help screen.

Displays an info screen on the GLCD screen with some basic information about the device.

Parameters

Void.	
-------	--

Returns

Void.

2.4 info.h File Reference

Information screen for the Desk Green House.

Functions

- int [info](#) (void)

Displays an info screen on the GLCD screen with some basic information about the device.

2.4.1 Detailed Description

Information screen for the Desk Green House.

Author

T. Buckingham

```
The file contains ::
+ Function prototypes
```

2.4.2 Function Documentation

2.4.2.1 info()

```
int info (
    void )
```

Displays an info screen on the GLCD screen with some basic information about the device.

Parameters

Void.	
-------	--

Returns

int - next page for [sensors.c](#) to open.

Displays an info screen on the GLCD screen with some basic information about the device.

Parameters

Void.	
-------	--

Returns

Void.

2.5 moisture.c File Reference

Moisture sensor related settings.

```
#include "stm32f7xx_hal.h"
#include "GLCD_Config.h"
#include "Board_GLCD.h"
#include "cmsis_os.h"
#include "SystemClock.h"
#include "Board_Touch.h"
#include "stm32f7xx_hal_gpio.h"
#include "moisture.h"
#include "ui_elements.h"
#include <stdio.h>
#include <stdlib.h>
#include "string.h"
#include "Serial.h"
#include "pump.h"
#include "fan.h"
```

Macros

- #define **wait_delay** HAL_Delay

Functions

- void [moisture](#) (void)
Test function for the moisture sensor - [unused].

Variables

- GLCD_FONT **GLCD_Font_6x8**
- GLCD_FONT **GLCD_Font_16x24**
- const int **AirValue** = 650
- const int **WaterValue** = 300
- int **intervals** = (**AirValue** - **WaterValue**)/3

2.5.1 Detailed Description

Moisture sensor related settings.

Author

T. Buckingham

```
The file contains ::
+ The air and water variables for determining the current moisture level
+ A basic test function for the moisture sensor
```

2.5.2 Function Documentation

2.5.2.1 moisture()

```
void moisture (
    void )
```

Test function for the moisture sensor - [unused].

Currently used as a testing function for the moisture sensor.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

2.5.3 Variable Documentation

2.5.3.1 AirValue

```
const int AirValue = 650
```

Defines air value reading of the sensor

2.5.3.2 intervals

```
int intervals = (AirValue - WaterValue)/3
```

Intervals used in determining if the reading is considered 'dry' or 'wet'

2.5.3.3 WaterValue

```
const int WaterValue = 300
```

Defines water value reading of the sensor

2.6 moisture.h File Reference

Moisture sensor and values handler.

Functions

- void `moisture` (void)
Currently used as a testing function for the moisture sensor.

Variables

- const int `AirValue`
- const int `WaterValue`
- int `intervals`

2.6.1 Detailed Description

Moisture sensor and values handler.

Author

T. Buckingham

```
The file contains ::  
+ Variables for calculating the current moisture level
```

2.6.2 Function Documentation

2.6.2.1 moisture()

```
void moisture (  
    void )
```

Currently used as a testing function for the moisture sensor.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

Currently used as a testing function for the moisture sensor.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

2.6.3 Variable Documentation

2.6.3.1 AirValue

```
const int AirValue [extern]
```

Defines air value reading of the sensor

2.6.3.2 intervals

```
int intervals [extern]
```

Intervals used indetermining if the reading is considered 'dry' or 'wet'

2.6.3.3 WaterValue

```
const int WaterValue [extern]
```

Defines water value reading of the sensor

2.7 pump.c File Reference

Pump control functions.

```
#include <stm32f7xx_hal_gpio.h>
#include "stm32f7xx_hal.h"
#include "ui_elements.h"
#include "pump.h"
```

Functions

- void `pump_start` (int time)
Runs the pump for a given amount of time (in milliseconds).
- void `pump_init` (void)
Initialises the pin used to activate the relay and thus the pump - pin D7.

Variables

- GPIO_InitTypeDef `pump_pin`

2.7.1 Detailed Description

Pump control functions.

Author

T. Buckingham

```
The file contains ::  
+ Basic function to run the pump
```

2.7.2 Function Documentation

2.7.2.1 `pump_init()`

```
void pump_init (  
    void )
```

Initialises the pin used to activate the relay and thus the pump - pin D7.

Initialises the necessary pins for activating the relay.

Parameters

Void.	
-------	--

Returns

Void.

2.7.2.2 `pump_start()`

```
void pump_start (  
    int time )
```

Runs the pump for a given amount of time (in milliseconds).

Activates the relay that in turn turns on the pump for x amount of time.

Parameters

Void.	
-------	--

Returns

Void.

2.8 pump.h File Reference

Pump initialisation and control.

Functions

- void `pump_start` (int time)
Activates the relay that in turn turns on the pump for x amount of time.
- void `pump_init` (void)
Initialises the necessary pins for activating the relay.

2.8.1 Detailed Description

Pump initialisation and control.

Author

T. Buckingham

```
The file contains ::  
+ Function prototypes
```

2.8.2 Function Documentation

2.8.2.1 pump_init()

```
void pump_init (  
    void )
```

Initialises the necessary pins for activating the relay.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

Initialises the necessary pins for activating the relay.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

2.8.2.2 pump_start()

```
void pump_start (
    int time )
```

Activates the relay that in turn turns on the pump for x amount of time.

Parameters

<i>time</i>	The amount of time the pump will be on for.
-------------	---

Returns

Void.

Activates the relay that in turn turns on the pump for x amount of time.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

2.9 sensors.c File Reference

Sensor screen displayed to the user and output sensor handler.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "stm32f7xx_hal.h"
#include "GLCD_Config.h"
#include "Board_GLCD.h"
#include "cmsis_os.h"
#include "SystemClock.h"
#include "Board_Touch.h"
#include "ui_elements.h"
#include "info.h"
#include "settings.h"
#include "serial.h"
#include "sensors.h"
#include "fan.h"
#include "moisture.h"
#include "pump.h"
```

Macros

- #define **wait_delay** HAL_Delay
- #define **TEMP** 3 /* button option for the temperature box */
- #define **HUMID** 4 /* button option for the humidity box */
- #define **MOIST** 5 /* button option for the [moisture](#) box */
- #define **RESET** 6 /* button option for the reset option */
- #define **TEST** 7 /* button option for the [test](#) option */
- #define **DRY** 0b00000000 /* if the Arduino states the [moisture](#) sensor is dry */
- #define **WET** 0b00000001 /* if the Arduino states the [moisture](#) sensor is wet */
- #define **V_WET** 0b00000010 /* if the Arduino states the [moisture](#) sensor is very wet, used the same as wet */
- #define **PUMP_DELAY** 10800 /* pump timer based on the regular transmissions of the Arduino */

Functions

- void [handler](#) (void)
Handles whether the fan and/or pump should turn on based on whether the readings are greater than the max values or if the moisture flag is 'wet' or 'dry'. Fan speed is determined by the difference between the max value and current reading.
- void [test](#) (void)
Basic function used to test if the peripherals are working - used by the user.
- void [draw_sensor_box](#) (char *name, int amount)
Draws a box with a background, containing the sensor name and current value.
- void [draw_sensors](#) (void)
Draws the sensors screen with boxes, buttons and top bar.
- void [sensors](#) (void)
The control function of the sensors screen. Handles return values from other screens.
- void [draw_sensors_box](#) (void)

Variables

- GLCD_FONT **GLCD_Font_6x8**
- GLCD_FONT **GLCD_Font_16x24**
- int **fan_speed**
- int **pump_tick**
- unsigned char **pump_time** = 0
- unsigned char **temp_value** = 0
- unsigned char **humidity_value** = 0
- unsigned char **soilMoistureValue** = 0
- unsigned char * **sensor_values** [3]
- char * **sensor_names** [] = {"Temperature\0", "Humidity\0", "Moisture\0"}

2.9.1 Detailed Description

Sensor screen displayed to the user and output sensor handler.

Author

T. Buckingham

```
The file contains ::  
+ Functions for displaying the current input readigns to the user  
  + A handler function to handle turning the pump and fans on or off  
+ Local definitions  
  - Button options  
  - Pump setting flags
```

2.9.2 Function Documentation

2.9.2.1 draw_sensor_box()

```
void draw_sensor_box (  
    char * name,  
    int amount )
```

Draws a box with a background, containing the sensor name and current value.

Parameters

Void.	
-------	--

Returns

Void.

2.9.2.2 draw_sensors()

```
void draw_sensors (
    void )
```

Draws the sensors screen with boxes, buttons and top bar.

Parameters

Void.	
-------	--

Returns

Void.

2.9.2.3 handler()

```
void handler (
    void )
```

Handles whether the fan and/or pump should turn on based on whether the readings are greater than the max values or if the moisture flag is 'wet' or 'dry'. Fan speed is determined by the difference between the max value and current reading.

Parameters

Void.	
-------	--

Returns

Void.

2.9.2.4 sensors()

```
void sensors (
    void )
```

The control function of the sensors screen. Handles return values from other screens.

Displays the sensors screen to the user. Handles return values from other screens' button presses. Handles the output peripherals.

Parameters

Void.	
-------	--

Returns

Void.

2.9.2.5 test()

```
void test (
    void )
```

Basic function used to test if the peripherals are working - used by the user.

Parameters

Void.	
-------	--

Returns

Void.

2.10 sensors.h File Reference

Sensors screen and outputs sensors handler.

Functions

- void [sensors](#) (void)

Displays the sensors screen to the user. Handles return values from other screens' button presses. Handles the output peripherals.

2.10.1 Detailed Description

Sensors screen and outputs sensors handler.

Author

T. Buckingham

```
The file contains ::
+ Function prototypes
```

2.10.2 Function Documentation**2.10.2.1 sensors()**

```
void sensors (
    void )
```

Displays the sensors screen to the user. Handles return values from other screens' button presses. Handles the output peripherals.

Parameters

Void.	
-------	--

Returns

Void.

Displays the sensors screen to the user. Handles return values from other screens' button presses. Handles the output peripherals.

Parameters

Void.	
-------	--

Returns

Void.

2.11 Serial.c File Reference

Moisture sensor related settings.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include "stm32f7xx.h"
#include "stm32f7xx_hal.h"
#include <stm32f7xx_hal_gpio.h>
#include "GLCD_Config.h"
#include "Board_GLCD.h"
#include "Serial.h"
#include "dwt_delay.h"
```

Functions

- uint32_t **DWT_Delay_Init** (void)
- void **EXTI9_5_IRQHandler** (void)

Serial interrupt handler. Reads in a buffer from the Arduino.
- void **SER_Init** (void)

Determines if a string box, or button, has been pressed by the user.
- void **SER_Write** (unsigned char buffer)

Function to transmit an 8 bit buffer to the arduino.
- void **SER_Busy** (void)

Function to tell the arduino to stop and reset as the STM32 is busy - [unused].
- void **SER_Ready** (void)

Function to tell the arduino begin transmitting as the STM32 is ready - [unused].

Variables

- int `i` = 0
- GPIO_InitTypeDef `read_pin`
- GPIO_InitTypeDef `write_pin`
- unsigned char `sensor_num`
- unsigned char `input_buffer` [3]
- unsigned char `output_buffer` [3]

2.11.1 Detailed Description

Moisture sensor related settings.

Author

T. Buckingham

```
The file contains ::
+ Functions for transmitting and recieving bytes
+ Input and output buffers
+ Counters to state what sensor is currently being transmitted recieved
```

2.11.2 Function Documentation

2.11.2.1 EXTI9_5_IRQHandler()

```
void EXTI9_5_IRQHandler (
    void )
```

Serial interrupt handler. Reads in a buffer from the Arduino.

Parameters

Void.	
-------	--

Returns

Void.

2.11.2.2 SER_Busy()

```
void SER_Busy (
    void )
```

Function to tell the arduino to stop and reset as the STM32 is busy - [unused].

Tells the Arduino that the STM32 is busy and cannot recieve data. The Arduino will reset its transmission if any are partially sent.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

2.11.2.3 SER_Init()

```
void SER_Init (
                void )
```

Determines is a string box, or button, has been pressed by the user.

Initialises the necessary pins for input and output, and sets interrupts.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

2.11.2.4 SER_Ready()

```
void SER_Ready (
                void )
```

Function to tell the arduino begin transmitting as the STM32 is ready - [unused].

Tells the Arduino that the STM32 is ready to receive data.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

2.11.2.5 SER_Write()

```
void SER_Write (
    unsigned char buffer )
```

Function to transmit an 8 bit buffer to the arduino.

Sends a buffer to the Arduino using the transmit pin. [not currently used].

Parameters

<i>Void.</i>

Returns

Void.

2.11.3 Variable Documentation

2.11.3.1 input_buffer

```
unsigned char input_buffer[3]
```

Input buffer used to store recieved sensor data

2.11.3.2 output_buffer

```
unsigned char output_buffer[3]
```

Output buffer used to store outgoing sensor data

2.11.3.3 sensor_num

```
unsigned char sensor_num
```

Counter to track of what sensor data is being recieved/transmitted

2.12 Serial.h File Reference

Serial communication between the STM32 and the Arduino.

Functions

- void `SER_Init` (void)
Initialises the necessary pins for input and output, and sets interrupts.
- void `SER_Read` (unsigned char buffer)
Reads incoming data from the Arduino - [deprecated] as an interrupt handler is used.
- void `SER_Write` (unsigned char buffer)
Sends a buffer to the Arduino using the transmit pin. [not currently used].
- void `SER_Busy` (void)
Tells the Arduino that the STM32 is busy and cannot receive data. The Arduino will reset its transmission if any are partially sent.
- void `SER_Ready` (void)
Tells the Arduino that the STM32 is ready to receive data.

Variables

- unsigned char `sensor_num`
- unsigned char `input_buffer` [3]
- unsigned char `output_buffer` [3]

2.12.1 Detailed Description

Serial communication between the STM32 and the Arduino.

Author

T. Buckingham

```
The file contains ::
+ Buffers for inputting and outputting data using bit banging
```

2.12.2 Function Documentation

2.12.2.1 `SER_Busy()`

```
void SER_Busy (
    void )
```

Tells the Arduino that the STM32 is busy and cannot receive data. The Arduino will reset its transmission if any are partially sent.

Parameters

Void.	
-------	--

Returns

Void.

Tells the Arduino that the STM32 is busy and cannot receive data. The Arduino will reset its transmission if any are partially sent.

Parameters

Void.	
-------	--

Returns

Void.

2.12.2.2 SER_Init()

```
void SER_Init (
    void )
```

Initialises the necessary pins for input and output, and sets interrupts.

Parameters

Void.	
-------	--

Returns

Void.

Initialises the necessary pins for input and output, and sets interrupts.

Parameters

Void.	
-------	--

Returns

Void.

2.12.2.3 SER_Read()

```
void SER_Read (
    unsigned char buffer )
```

Reads incoming data from the Arduino - [deprecated] as an interrupt handler is used.

Parameters

<i>buffer</i>	The 8 bits buffer used to store the recieved data.
---------------	--

Returns

Void.

2.12.2.4 SER_Ready()

```
void SER_Ready (
    void )
```

Tells the Arduino that the STM32 is ready to recieve data.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

Tells the Arduino that the STM32 is ready to recieve data.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

2.12.2.5 SER_Write()

```
void SER_Write (
    unsigned char buffer )
```

Sends a buffer to the Arduino using the transmit pin. [not currently used].

Parameters

<i>buffer</i>	The 8 bit buffer being transmitted.
---------------	-------------------------------------

Returns

Void.

Sends a buffer to the Arduino using the transmit pin. [not currently used].

Parameters

Void.	
-------	--

Returns

Void.

2.12.3 Variable Documentation**2.12.3.1 input_buffer**

```
unsigned char input_buffer[3] [extern]
```

Input buffer used to store recieved sensor data

2.12.3.2 output_buffer

```
unsigned char output_buffer[3] [extern]
```

Output buffer used to store outgoing sensor data

2.12.3.3 sensor_num

```
unsigned char sensor_num [extern]
```

Counter to track of what sensor data is being recieved/transmitted

2.13 settings.h File Reference

User defined settings used in [sensors.c](#)'s handler.

Functions

- int [settings](#) (void)

Displays the settings screen to the user by drawing the UI elements.

Variables

- unsigned char [max_temp](#)
- unsigned char [min_moisture_level](#)
- unsigned char [max_humidity](#)

2.13.1 Detailed Description

User defined settings used in [sensors.c](#)'s handler.

Author

T. Buckingham

```
The file contains ::  
+ External variables defined by the user for controller the devices'  
  temperature, humidity and moisture level
```

2.13.2 Function Documentation

2.13.2.1 settings()

```
int settings (  
    void )
```

Displays the settings screen to the user by drawing the UI elements.

Parameters

Void.	
-------	--

Returns

int that tells the [sensors.c](#) what screen to display next

Displays the settings screen to the user by drawing the UI elements.

Parameters

Void.	
-------	--

Returns

Void.

2.13.3 Variable Documentation

2.13.3.1 max_humidity

```
unsigned char max_humidity [extern]
```

Defines the maximum humidity of the device - set by the user

2.13.3.2 max_temp

```
unsigned char max_temp [extern]
```

Defines the maximum temperature of the device - set by the user

2.13.3.3 min_moisture_level

```
unsigned char min_moisture_level [extern]
```

Defines the minimum moisture level of the soil - set by the user

2.14 SystemClock.c File Reference

Clock configuration for the stm32f746g-disco.

```
#include "stm32f7xx_hal.h"
#include "SystemClock.h"
```

Functions

- void [SystemClock_Config](#) (void)
- void **Error_Handler** (void)

2.14.1 Detailed Description

Clock configuration for the stm32f746g-disco.

Author

Generated by the STM32CubeMX

```
The file contains ::
+ Information text box displayed to the user on the 'help' screen
```

2.14.2 Function Documentation

2.14.2.1 SystemClock_Config()

```
void SystemClock_Config (  
    void )
```

System Clock Configuration

2.15 SystemClock.h File Reference

Clock configuration for the stm32f746g-disco.

Functions

- void [SystemClock_Config](#) (void)
- void **Error_Handler** (void)

2.15.1 Detailed Description

Clock configuration for the stm32f746g-disco.

Author

Generated by the STM32CubeMX

```
The file contains ::  
+ Function protoypes
```

2.15.2 Function Documentation

2.15.2.1 SystemClock_Config()

```
void SystemClock_Config (  
    void )
```

System Clock Configuration

2.16 ui_elements.c File Reference

User interface elements for creating the various user screens.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "stm32f7xx_hal.h"
#include "GLCD_Config.h"
#include "Board_GLCD.h"
#include "cmsis_os.h"
#include "SystemClock.h"
#include "Board_Touch.h"
#include "ui_elements.h"
#include "info.h"
#include "settings.h"
#include "serial.h"
#include "sensors.h"
#include "moisture.h"
```

Macros

- #define **wait_delay** HAL_Delay
- #define **HOME** 0 /* home option for the user */
- #define **SETTINGS** 1 /* [settings](#) option for the user */
- #define **HELP** 2 /* help option for the user*/

Functions

- void [draw_flat_background](#) (void)
Function to draw the solid colour flat background.
- void [draw_background](#) (void)
Function to draw the checkered background.
- void [draw_check_option](#) (int x, int y, char *str, int boxes)
Draws a string with checkboxes to the right of it.
- void [draw_string_box](#) (int x, int y, char *str)
Draws a pseudo button on the screen by drawing a box then drawing a string on top. The padding uniform and adjusts to the strings length.
- void [draw_top_bar](#) (void)
Draws the top menu bar with the options 'Home', 'Settings' and 'Help'. These options [sensors.c](#) , [settings.c](#) and [info.c](#) respectively.
- int **main** (void)

Variables

- GLCD_FONT **GLCD_Font_6x8**
- GLCD_FONT **GLCD_Font_16x24**

2.16.1 Detailed Description

User interface elements for creating the various user screens.

Author

T. Buckingham

```
The file contains ::
+ Two functions to draw different backgrounds
    - A checkered background
    - A flat solid colour background
+ A function to draw a top menu bar with options
+ A 'string box' function to draw pseudo buttons
```

2.16.2 Function Documentation

2.16.2.1 draw_background()

```
void draw_background (
    void )
```

Function to draw the checkered background.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

2.16.2.2 draw_check_option()

```
void draw_check_option (
    int x,
    int y,
    char * str,
    int boxes )
```

Draws a string with checkboxes to the right of it.

Parameters

<i>x</i>	The x position of the start of the string.
<i>y</i>	The y position of the string.
<i>str</i>	The string to be drawn before the boxes.
<i>boxes</i>	The number of boxes drawn after the string.

Returns

Void.

2.16.2.3 draw_flat_background()

```
void draw_flat_background (
    void )
```

Function to draw the solid colour flat background.

Parameters

Void.	
-------	--

Returns

Void.

2.16.2.4 draw_string_box()

```
void draw_string_box (
    int x,
    int y,
    char * str )
```

Draws a pseudo button on the screen by drawing a box then drawing a string on top. The padding uniform and adjusts to the strings length.

Parameters

<i>x</i>	The x position of the start button.
<i>y</i>	The y position of the start button.
<i>str</i>	The string that will be drawn in the button.

Returns

Void.

2.16.2.5 draw_top_bar()

```
void draw_top_bar (
    void )
```

Draws the top menu bar with the options 'Home', 'Settings' and 'Help'. These options [sensors.c](#) , [settings.c](#) and [info.c](#) respectively.

Parameters

Void.

Returns

Void.

2.17 ui_elements.h File Reference

Sensors screen and outputs sensors handler.

Macros

- #define [HOME](#) 0
- #define **SETTINGS** 1
- #define **HELP** 2
- #define [BACKGROUND_BLUE](#) 0x02D9
- #define **FOREGROUND_BLUE** 0x52DF
- #define **TEXT_BLUE** 0x5F5F
- #define **TEXT_WHITE** 0x739C
- #define **LIGHT_GREY** 0x4A52

Functions

- void [draw_string_box](#) (int x, int y, char *str)
Draws a pseudo button on the screen by drawing a box then drawing a string on top. The padding uniform and adjusts to the strings length.
- void [draw_top_bar](#) (void)
Draws the top menu bar with the options 'Home', 'Settings' and 'Help'. These options [sensors.c](#) , [settings.c](#) and [info.c](#) respectively.
- void [draw_check_option](#) (int x, int y, char *str, int boxes)
Draws a string with checkboxes to the right of it.

2.17.1 Detailed Description

Sensors screen and outputs sensors handler.

Author

T. Buckingham

```
The file contains ::
+ Option definitions for the top menu bar
+ Colours used throughout the project for UI elements
```

2.17.2 Macro Definition Documentation

2.17.2.1 BACKGROUND_BLUE

```
#define BACKGROUND_BLUE 0x02D9
```

Defines the colour values used throughout the UI elements.

2.17.2.2 HOME

```
#define HOME 0
```

Global button options - mostly used in the top menu bar

2.17.3 Function Documentation

2.17.3.1 draw_check_option()

```
void draw_check_option (
    int x,
    int y,
    char * str,
    int boxes )
```

Draws a string with checkboxes to the right of it.

Parameters

<i>x</i>	The x position of the start of the string.
<i>y</i>	The y position of the string.
<i>str</i>	The string to be drawn before the boxes.
<i>boxes</i>	The number of boxes drawn after the string.

Returns

Void.

2.17.3.2 draw_string_box()

```
void draw_string_box (
    int x,
    int y,
    char * str )
```

Draws a pseudo button on the screen by drawing a box then drawing a string on top. The padding uniform and adjusts to the strings length.

Parameters

<i>x</i>	The x position of the start button.
<i>y</i>	The y position of the start button.
<i>str</i>	The string that will be drawn in the button.

Returns

Void.

2.17.3.3 draw_top_bar()

```
void draw_top_bar (
    void )
```

Draws the top menu bar with the options 'Home', 'Settings' and 'Help'. These options [sensors.c](#) , [settings.c](#) and [info.c](#) respectively.

Parameters

<i>Void.</i>	
--------------	--

Returns

Void.

Index

AirValue
 moisture.c, 10
 moisture.h, 12

BACKGROUND_BLUE
 ui_elements.h, 35

draw_background
 ui_elements.c, 32

draw_check_option
 ui_elements.c, 32
 ui_elements.h, 36

draw_flat_background
 ui_elements.c, 33

draw_info
 info.c, 7

draw_sensor_box
 sensors.c, 17

draw_sensors
 sensors.c, 17

draw_string_box
 ui_elements.c, 33
 ui_elements.h, 36

draw_top_bar
 ui_elements.c, 33
 ui_elements.h, 37

EXTI9_5_IRQHandler
 Serial.c, 21

fan.c, 3
 fan_init, 4
 fan_run, 4
 PA15_Init, 4

fan.h, 5
 fan_init, 5
 fan_run, 6
 PA15_Init, 6

fan_init
 fan.c, 4
 fan.h, 5

fan_run
 fan.c, 4
 fan.h, 6

handler
 sensors.c, 18

HOME
 ui_elements.h, 36

info
 info.c, 7
 info.h, 8

info.c, 6
 draw_info, 7
 info, 7

info.h, 8
 info, 8

input_buffer
 Serial.c, 23
 Serial.h, 27

intervals
 moisture.c, 10
 moisture.h, 12

max_humidity
 settings.h, 29

max_temp
 settings.h, 29

min_moisture_level
 settings.h, 29

moisture
 moisture.c, 10
 moisture.h, 11

moisture.c, 9
 AirValue, 10
 intervals, 10
 moisture, 10
 WaterValue, 11

moisture.h, 11
 AirValue, 12
 intervals, 12
 moisture, 11
 WaterValue, 12

output_buffer
 Serial.c, 23
 Serial.h, 27

PA15_Init
 fan.c, 4
 fan.h, 6

pump.c, 12
 pump_init, 13
 pump_start, 13

pump.h, 14
 pump_init, 14
 pump_start, 15

pump_init
 pump.c, 13
 pump.h, 14

- pump_start
 - pump.c, [13](#)
 - pump.h, [15](#)
- sensor_num
 - Serial.c, [23](#)
 - Serial.h, [27](#)
- sensors
 - sensors.c, [18](#)
 - sensors.h, [19](#)
- sensors.c, [15](#)
 - draw_sensor_box, [17](#)
 - draw_sensors, [17](#)
 - handler, [18](#)
 - sensors, [18](#)
 - test, [19](#)
- sensors.h, [19](#)
 - sensors, [19](#)
- SER_Busy
 - Serial.c, [21](#)
 - Serial.h, [24](#)
- SER_Init
 - Serial.c, [22](#)
 - Serial.h, [25](#)
- SER_Read
 - Serial.h, [25](#)
- SER_Ready
 - Serial.c, [22](#)
 - Serial.h, [26](#)
- SER_Write
 - Serial.c, [22](#)
 - Serial.h, [26](#)
- Serial.c, [20](#)
 - EXTI9_5_IRQHandler, [21](#)
 - input_buffer, [23](#)
 - output_buffer, [23](#)
 - sensor_num, [23](#)
 - SER_Busy, [21](#)
 - SER_Init, [22](#)
 - SER_Ready, [22](#)
 - SER_Write, [22](#)
- Serial.h, [23](#)
 - input_buffer, [27](#)
 - output_buffer, [27](#)
 - sensor_num, [27](#)
 - SER_Busy, [24](#)
 - SER_Init, [25](#)
 - SER_Read, [25](#)
 - SER_Ready, [26](#)
 - SER_Write, [26](#)
- settings
 - settings.h, [28](#)
- settings.h, [27](#)
 - max_humidity, [29](#)
 - max_temp, [29](#)
 - min_moisture_level, [29](#)
 - settings, [28](#)
- SystemClock.c, [29](#)
 - SystemClock_Config, [30](#)
- SystemClock.h, [30](#)
 - SystemClock_Config, [30](#)
- SystemClock_Config
 - SystemClock.c, [30](#)
 - SystemClock.h, [30](#)
- test
 - sensors.c, [19](#)
- ui_elements.c, [31](#)
 - draw_background, [32](#)
 - draw_check_option, [32](#)
 - draw_flat_background, [33](#)
 - draw_string_box, [33](#)
 - draw_top_bar, [33](#)
- ui_elements.h, [35](#)
 - BACKGROUND_BLUE, [35](#)
 - draw_check_option, [36](#)
 - draw_string_box, [36](#)
 - draw_top_bar, [37](#)
 - HOME, [36](#)
- WaterValue
 - moisture.c, [11](#)
 - moisture.h, [12](#)