

Microclimate montior

Generated by Doxygen 1.10.0



<b>1 Microclimate manager</b>	<b>1</b>
<b>2 Bug List</b>	<b>3</b>
<b>3 Data Structure Index</b>	<b>5</b>
3.1 Data Structures . . . . .	5
<b>4 File Index</b>	<b>7</b>
4.1 File List . . . . .	7
<b>5 Data Structure Documentation</b>	<b>9</b>
5.1 DHT_DataTypedef Struct Reference . . . . .	9
5.1.1 Detailed Description . . . . .	9
5.1.2 Field Documentation . . . . .	9
5.1.2.1 Humidity . . . . .	9
5.1.2.2 Temperature . . . . .	9
5.2 LDR_DataTypedef Struct Reference . . . . .	10
5.2.1 Detailed Description . . . . .	10
5.2.2 Field Documentation . . . . .	10
5.2.2.1 LDR_Val . . . . .	10
5.3 Moisture_DataTypedef Struct Reference . . . . .	10
5.3.1 Detailed Description . . . . .	10
5.3.2 Field Documentation . . . . .	11
5.3.2.1 MoisturePercent . . . . .	11
5.4 MQ135_DataTypedef Struct Reference . . . . .	11
5.4.1 Detailed Description . . . . .	11
5.4.2 Field Documentation . . . . .	11
5.4.2.1 PPM . . . . .	11
<b>6 File Documentation</b>	<b>13</b>
6.1 C:/temp/humidtemp/DHT.c File Reference . . . . .	13
6.1.1 Detailed Description . . . . .	14
6.1.2 Function Documentation . . . . .	14
6.1.2.1 DHT_Check_Response() . . . . .	14
6.1.2.2 DHT_GetData() . . . . .	14
6.1.2.3 DHT_Read() . . . . .	15
6.1.2.4 DHT_Start() . . . . .	15
6.1.2.5 DWT_Delay_Init() . . . . .	15
6.1.2.6 DWT_Delay_us() . . . . .	15
6.1.2.7 Set_Pin_Input() . . . . .	16
6.1.2.8 Set_Pin_Output() . . . . .	16
6.2 C:/temp/humidtemp/DHT.h File Reference . . . . .	16
6.2.1 Detailed Description . . . . .	16
6.2.2 Function Documentation . . . . .	17

6.2.2.1 DHT_GetData()	17
6.3 C:/temp/humidtemp/DHT.h	17
6.4 C:/temp/humidtemp/GLCD.c File Reference	17
6.4.1 Detailed Description	19
6.4.2 Function Documentation	19
6.4.2.1 checkAndDisplayStatus()	19
6.4.2.2 displayStatus()	19
6.4.2.3 GLCDLogic()	20
6.4.3 Variable Documentation	20
6.4.3.1 dataTransition	20
6.4.3.2 drawBars	20
6.4.3.3 GLCD_Font_16x24	20
6.4.3.4 GLCD_Font_6x8	20
6.4.3.5 Humidity	20
6.4.3.6 initialPage	21
6.4.3.7 k	21
6.4.3.8 ldr_val	21
6.4.3.9 preset	21
6.4.3.10 readyTransition	21
6.4.3.11 settingsTransition	21
6.4.3.12 showData	21
6.4.3.13 showDataPage	21
6.4.3.14 showSettingsPage	22
6.4.3.15 soilMoisturePercent	22
6.4.3.16 Temperature	22
6.4.3.17 tsc_state	22
6.4.3.18 userTrigger	22
6.5 C:/temp/humidtemp/GLCD.h File Reference	22
6.5.1 Detailed Description	22
6.5.2 Function Documentation	23
6.5.2.1 GLCDLogic()	23
6.6 C:/temp/humidtemp/GLCD.h	23
6.7 C:/temp/humidtemp/LDR.c File Reference	23
6.7.1 Detailed Description	23
6.7.2 Function Documentation	23
6.7.2.1 LDR_GetData()	23
6.7.2.2 readLDR()	24
6.8 C:/temp/humidtemp/LDR.h File Reference	24
6.8.1 Detailed Description	24
6.8.2 Function Documentation	24
6.8.2.1 LDR_GetData()	24
6.9 C:/temp/humidtemp/LDR.h	25

6.10 C:/temp/humidtemp/main.c File Reference	25
6.10.1 Detailed Description	26
6.10.2 Function Documentation	26
6.10.2.1 HAL_GPIO_EXTI_Callback()	26
6.10.3 Variable Documentation	26
6.10.3.1 buffer	26
6.10.3.2 GLCD_Font_16x24	26
6.10.3.3 GPIO_InitStruct	27
6.10.3.4 hadc1	27
6.10.3.5 hadc2	27
6.10.3.6 hadc3	27
6.10.3.7 userTrigger	27
6.11 C:/temp/humidtemp/main.h File Reference	27
6.11.1 Detailed Description	28
6.11.2 Variable Documentation	28
6.11.2.1 buffer	28
6.11.2.2 hadc1	28
6.11.2.3 hadc2	28
6.11.2.4 hadc3	28
6.11.2.5 userTrigger	28
6.12 C:/temp/humidtemp/main.h	29
6.13 C:/temp/humidtemp/Moisture.c File Reference	29
6.13.1 Detailed Description	30
6.13.2 Function Documentation	30
6.13.2.1 map()	30
6.13.2.2 Moisture_GetData()	30
6.13.2.3 readMoisture()	31
6.13.2.4 readMoisturePercent()	31
6.14 C:/temp/humidtemp/Moisture.h File Reference	31
6.14.1 Detailed Description	32
6.14.2 Function Documentation	32
6.14.2.1 Moisture_GetData()	32
6.15 C:/temp/humidtemp/Moisture.h	32
6.16 C:/temp/humidtemp/mq135.c File Reference	32
6.16.1 Detailed Description	33
6.16.2 Function Documentation	33
6.16.2.1 getMQ135PPM()	33
6.16.2.2 getMQ135Resistance()	34
6.16.2.3 MQ_GetData()	34
6.16.2.4 my_pow()	34
6.16.2.5 readMQ135()	34
6.17 C:/temp/humidtemp/mq135.h File Reference	35

6.17.1 Detailed Description . . . . .	35
6.17.2 Function Documentation . . . . .	35
6.17.2.1 MQ_GetData() . . . . .	35
6.18 C:/temp/humidtemp/mq135.h . . . . .	36
<b>Index</b>	<b>37</b>

# Chapter 1

## Microclimate manager

### Author

Joseph Sier ( [pxw22tju@uea.ac.uk](mailto:pxw22tju@uea.ac.uk) )

The aim of this project is to provide a device that can read necessary values within a closed terrarium.

Peripherals used: MQ135, DHT11, Light Dependent Resistor (photocell), Capacitive Moisture Sensor, GLCD and touchscreen.

The initial warmup page presents a ready button to enter the data page, or alternatively you can skip the warmup and go straight to settings (via bottom right button) to choose a preset.

I chose the presets "tropical" "temperate" and "desert" as i thought that those would be the most likely options within a terrarium in the UK e.g succulents, ferns etc.

Upon choosing a preset and returning to the data page, the user can get some guidance on what needs to be adjusted within their terrarium.

Header files: for the sensors, i used typedef to easily assign the data and make it more readable and manageable. It also allowed me to encapsulate each sensor within its own file, for easy and modular use.

In most of the .h files, i opted to use uint8-16 or int8, as most of these values were percentages, they would have a max of 100 or stay within four digits, and some ([moisture.h](#)) could enter negatives, which is why it has to be signed.





## Chapter 2

# Bug List

### File [DHT.c](#)

There is an issue with the timer in DHT11, and the code will only run from flash load. when the device is disconnected and plugged in, it will hang on the [DHT\\_Start\(\)](#).



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">DHT_DataTypeDef</a>	
Struct for storing DHT data . . . . .	9
<a href="#">LDR_DataTypeDef</a>	
Struct for storing LDR data . . . . .	10
<a href="#">Moisture_DataTypeDef</a>	
Struct for storing moisture data . . . . .	10
<a href="#">MQ135_DataTypeDef</a>	
Struct for storing MQ135 data . . . . .	11



# Chapter 4

## File Index

### 4.1 File List

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

C:/temp/humidtemp/ <a href="#">DHT.c</a>	
This file contains functions to read data from the DHT11	13
C:/temp/humidtemp/ <a href="#">DHT.h</a>	
Header File Containing typedef definition	16
C:/temp/humidtemp/ <a href="#">GLCD.c</a>	
Implementation of GLCD functions	17
C:/temp/humidtemp/ <a href="#">GLCD.h</a>	
Header file containing the declaration of the function GLCDLogic	22
C:/temp/humidtemp/ <a href="#">LDR.c</a>	
This file contains functions to read data from the Light Dependent Resistor (LDR)	23
C:/temp/humidtemp/ <a href="#">LDR.h</a>	
Header File Containing typedef definition	24
C:/temp/humidtemp/ <a href="#">main.c</a>	
This file contains functions to read sensor data and control the display	25
C:/temp/humidtemp/ <a href="#">main.h</a>	
File Containing definitions for hadc, buffer and userTrigger	27
C:/temp/humidtemp/ <a href="#">Moisture.c</a>	
This file contains functions to read moisture sensor data and calculate moisture percentage	29
C:/temp/humidtemp/ <a href="#">Moisture.h</a>	
Header File Containing typedef definition	31
C:/temp/humidtemp/ <a href="#">mq135.c</a>	
This file contains functions to read data from the MQ135 sensor and calculate the PPM (Parts Per Million) of CO2	32
C:/temp/humidtemp/ <a href="#">mq135.h</a>	
Header File Containing typedef definition	35



## Chapter 5

# Data Structure Documentation

### 5.1 DHT\_DataTypedef Struct Reference

Struct for storing DHT data.

```
#include <DHT.h>
```

#### Data Fields

- uint8\_t [Temperature](#)
- uint8\_t [Humidity](#)

#### 5.1.1 Detailed Description

Struct for storing DHT data.

This struct contains two members, `Temperature` and `Humidity`, which represents the temperature and humidity from the sensor.

#### 5.1.2 Field Documentation

##### 5.1.2.1 Humidity

```
uint8_t Humidity
```

Humidity

##### 5.1.2.2 Temperature

```
uint8_t Temperature
```

Temperature

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

- `C:/temp/humidtemp/DHT.h`

## 5.2 LDR\_DataTypedef Struct Reference

Struct for storing LDR data.

```
#include <LDR.h>
```

### Data Fields

- int [LDR\\_Val](#)

### 5.2.1 Detailed Description

Struct for storing LDR data.

This struct contains a single member, `LDR_Val`, which represents the analog reading from LDR.

### 5.2.2 Field Documentation

#### 5.2.2.1 LDR\_Val

```
int LDR_Val
```

Analog LDR value

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

- `C:/temp/humidtemp/LDR.h`

## 5.3 Moisture\_DataTypedef Struct Reference

Struct for storing moisture data.

```
#include <Moisture.h>
```

### Data Fields

- int8\_t [MoisturePercent](#)

### 5.3.1 Detailed Description

Struct for storing moisture data.

This struct contains a single member, `MoisturePercent`, which represents the percentage of moisture.



### 5.3.2 Field Documentation

#### 5.3.2.1 MoisturePercent

```
int8_t MoisturePercent
```

Percentage of moisture

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

- C:/temp/humidtemp/[Moisture.h](#)

## 5.4 MQ135\_DataTypedef Struct Reference

Struct for storing MQ135 data.

```
#include <mq135.h>
```

### Data Fields

- uint16\_t [PPM](#)

### 5.4.1 Detailed Description

Struct for storing MQ135 data.

This struct contains a single member, `PPM`, which represents the CO2 in PPM (Parts per million).

### 5.4.2 Field Documentation

#### 5.4.2.1 PPM

```
uint16_t PPM
```

PPM of CO2

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

- C:/temp/humidtemp/[mq135.h](#)



# Chapter 6

## File Documentation

### 6.1 C:/temp/humidtemp/DHT.c File Reference

This file contains functions to read data from the DHT11.

```
#include "stm32f7xx_hal.h"
#include "main.h"
#include "DHT.h"
```

#### Macros

- #define **DHT\_PORT** GPIOA
- #define **DHT\_PIN** GPIO\_PIN\_0

#### Functions

- uint32\_t **DWT\_Delay\_Init** (void)  
*Initializes the DWT (Data Watchpoint and Trace) unit for microsecond delay. used from <https://deepblueembedded.com/stm32-delay-microsecond-millisecond-utility-dwt-delay-timer-delay/> as HAL works in ms only.*
- \_\_STATIC\_INLINE void **DWT\_Delay\_us** (volatile uint32\_t au32\_microseconds)  
*Provides delay in microseconds using DWT.*
- void **Set\_Pin\_Output** (GPIO\_TypeDef \*GPIOx, uint16\_t GPIO\_Pin)  
*Sets a GPIO pin as output.*
- void **Set\_Pin\_Input** (GPIO\_TypeDef \*GPIOx, uint16\_t GPIO\_Pin)  
*Sets a GPIO pin as input.*
- void **DHT\_Start** (void)  
*Initiates communication with DHT sensor.*
- uint8\_t **DHT\_Check\_Response** (void)  
*Checks response from DHT sensor.*
- uint8\_t **DHT\_Read** (void)  
*Reads data from DHT sensor.*
- void **DHT\_GetData** (**DHT\_DataTypedef** \*DHT\_Data)  
*Get DHT data.*

## Variables

- `uint8_t Rh_byte1`
- `uint8_t Rh_byte2`
- `uint8_t Temp_byte1`
- `uint8_t Temp_byte2`
- `uint16_t SUM`
- `uint8_t Presence = 0`

### 6.1.1 Detailed Description

This file contains functions to read data from the DHT11.

#### Author

Joseph Sier

DHT logic based on <https://controllerstech.com/using-dht11-sensor-with-stm32/> The DHT11 uses serial communication, which is very susceptible to errors, as most of the delays need to be in microseconds, which HAL does not support. I used a DWT delay to get a us (microsecond) delay.

The DHT sensor operates by initiating communication with a start pulse to request data, then it awaits and reads the response pulse sent back by the DHT sensor. This response contains 40 bits of data which includes information about temperature and humidity. Upon reception, the data is processed and verified for integrity before being parsed and assigned to their respective temperature and humidity values.

**Bug** There is an issue with the timer in DHT11, and the code will only run from flash load. when the device is disconnected and plugged in, it will hang on the `DHT_Start()`.

### 6.1.2 Function Documentation

#### 6.1.2.1 DHT\_Check\_Response()

```
uint8_t DHT_Check_Response (
    void )
```

Checks response from DHT sensor.

#### Returns

1 if response received, -1 if response timeout.

#### 6.1.2.2 DHT\_GetData()

```
void DHT_GetData (
    DHT_DataTypedef * DHT_Data )
```

Get DHT data.

This function retrieves DHT data and stores it in the provided `DHT_DataTypedef` structure.

**Parameters**

<i>DHT_Data</i>	Pointer to the structure <a href="#">DHT_DataTypeDef</a> .
-----------------	--

**6.1.2.3 DHT\_Read()**

```
uint8_t DHT_Read (
    void )
```

Reads data from DHT sensor.

**Returns**

Data read from sensor.

**6.1.2.4 DHT\_Start()**

```
void DHT_Start (
    void )
```

Initiates communication with DHT sensor.

This function initializes the communication sequence with the DHT sensor to request data.

**6.1.2.5 DWT\_Delay\_Init()**

```
uint32_t DWT_Delay_Init (
    void )
```

Initializes the DWT (Data Watchpoint and Trace) unit for microsecond delay. used from <https://deepbluembedded.com/stm32-delay-microsecond-millisecond-utility-dwt-delay-timer-delay/> as HAL works in ms only.

**Returns**

0 if initialization successful, 1 otherwise.

**6.1.2.6 DWT\_Delay\_us()**

```
__STATIC_INLINE void DWT_Delay_us (
    volatile uint32_t au32_microseconds )
```

Provides delay in microseconds using DWT.

**Parameters**

<i>au32_microseconds</i>	Delay duration in microseconds.
--------------------------	---------------------------------

### 6.1.2.7 Set\_Pin\_Input()

```
void Set_Pin_Input (
    GPIO_TypeDef * GPIOx,
    uint16_t GPIO_Pin )
```

Sets a GPIO pin as input.

#### Parameters

<i>GPIOx</i>	Pointer to GPIO port.
<i>GPIO_Pin</i>	GPIO pin to set as input.

### 6.1.2.8 Set\_Pin\_Output()

```
void Set_Pin_Output (
    GPIO_TypeDef * GPIOx,
    uint16_t GPIO_Pin )
```

Sets a GPIO pin as output.

#### Parameters

<i>GPIOx</i>	Pointer to GPIO port.
<i>GPIO_Pin</i>	GPIO pin to set as output.

## 6.2 C:/temp/humidtemp/DHT.h File Reference

Header File Containing typedef definition.

### Data Structures

- struct [DHT\\_DataTypeDef](#)  
*Struct for storing DHT data.*

### Functions

- void [DHT\\_GetData](#) ([DHT\\_DataTypeDef](#) \*DHT\_Data)  
*Get DHT data.*

### 6.2.1 Detailed Description

Header File Containing typedef definition.

#### Author

Joseph Sier

This file defines a typedef struct for managing moisture data.

## 6.2.2 Function Documentation

### 6.2.2.1 DHT\_GetData()

```
void DHT_GetData (
    DHT_DataTypedef * DHT_Data )
```

Get DHT data.

This function retrieves DHT data and stores it in the provided [DHT\\_DataTypedef](#) structure.

#### Parameters

<i>DHT_Data</i>	Pointer to the structure <a href="#">DHT_DataTypedef</a> .
-----------------	--

## 6.3 C:/temp/humidtemp/DHT.h

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

```
00001
00010 #ifndef DHT_H_
00011 #define DHT_H_
00012
00019 typedef struct
00020 {
00021     uint8_t Temperature;
00022     uint8_t Humidity;
00023 }DHT_DataTypedef;
00024
00033 void DHT_GetData (DHT_DataTypedef *DHT_Data);
00034
00035 #endif /* DHT_H_ */
00036
00037
00038
00039
00040
00041
00042
```

## 6.4 C:/temp/humidtemp/GLCD.c File Reference

Implementation of GLCD functions.

```
#include "Board_GLCD.h"
#include "Board_Touch.h"
#include "GLCD_Config.h"
#include "main.h"
#include <string.h>
#include "Moisture.h"
#include "mq135.h"
#include "DHT.h"
#include "LDR.h"
```

## Functions

- void **displayMoisture** (void)  
*Display soil moisture data on the GLCD.*
- void **getTimeActive** (void)  
*Display system uptime in hours, minutes, and seconds on the GLCD.*
- void **displayMQ135** ()  
*Display MQ135 data (CO2 concentration) on the GLCD.*
- void **displayDHT** ()  
*Display temperature and humidity data on the GLCD.*
- void **displayLDR** (void)  
*When the LDR detects light, it starts a counter which is displayed on the GLCD.*
- void **settingsPage** ()  
*Display settings page on the GLCD.*
- void **displayStatus** (int x, int y, const char \*status)  
*Display status message at specified location.*
- void **checkAndDisplayStatus** (int highthreshold, int lowThreshold, int value, int x, int y, const char \*highMsg, const char \*lowMsg, const char \*goodMsg)  
*Check value against thresholds and display corresponding status message.*
- void **dataPage** ()  
*Display data page on the GLCD.*
- void **readyPage** (void)  
*Display ready page on the GLCD.*
- void **allPage** (void)  
*Display all page elements on the GLCD.*
- void **warmUp** (void)  
*Display warm up page on the GLCD.*
- void **GLCDLogic** ()  
*GLCDLogic function.*

## Variables

- TOUCH\_STATE **tsc\_state**
- GLCD\_FONT **GLCD\_Font\_6x8**
- GLCD\_FONT **GLCD\_Font\_16x24**
- bool **userTrigger**
- bool **dataTransition** =true
- bool **drawBars** =true
- bool **initialPage** =true
- bool **readyTransition** =true
- bool **settingsTransition** =true
- bool **showData** =false
- bool **showDataPage** =false
- bool **showSettingsPage** =false
- char **preset** [9]
- uint8\_t **Temperature**
- uint8\_t **Humidity**
- int8\_t **soilMoisturePercent**
- uint16\_t **ldr\_val**
- uint32\_t **k**
- **LDR\_DataTypedef** **LDR\_Data**
- **DHT\_DataTypedef** **DHT11\_Data**
- **Moisture\_DataTypedef** **Moisture\_Data**
- **MQ135\_DataTypedef** **MQ135\_Data**



### 6.4.1 Detailed Description

Implementation of GLCD functions.

#### Author

Joseph Sier

I opted to keep everything GLCD related inside [GLCD.c](#).

This displays every page, as well as the data from sensors. Preset parameteres are loosely based off the Koppen climate classification.

### 6.4.2 Function Documentation

#### 6.4.2.1 checkAndDisplayStatus()

```
void checkAndDisplayStatus (
    int highThreshold,
    int lowThreshold,
    int value,
    int x,
    int y,
    const char * highMsg,
    const char * lowMsg,
    const char * goodMsg )
```

Check value against thresholds and display corresponding status message.

#### Parameters

<i>highThreshold</i>	High threshold value
<i>lowThreshold</i>	Low threshold value
<i>value</i>	Value to compare against thresholds
<i>x</i>	X-coordinate of the status message
<i>y</i>	Y-coordinate of the status message
<i>highMsg</i>	Message to display if value exceeds high threshold
<i>lowMsg</i>	Message to display if value falls below low threshold
<i>goodMsg</i>	Message to display if value is within thresholds

#### 6.4.2.2 displayStatus()

```
void displayStatus (
    int x,
    int y,
    const char * status )
```

Display status message at specified location.

**Parameters**

<i>x</i>	X-coordinate of the message
<i>y</i>	Y-coordinate of the message
<i>status</i>	Status message to display

**6.4.2.3 GLCDLogic()**

```
void GLCDLogic (
    void )
```

GLCDLogic function.

This function is responsible for the logic to control the GLCD. It provides the necessary functionality to interact with the GLCD hardware.

**6.4.3 Variable Documentation****6.4.3.1 dataTransition**

```
bool dataTransition =true
```

Flag indicating transition in data display.

**6.4.3.2 drawBars**

```
bool drawBars =true
```

Flag indicating whether to draw bars on the display.

**6.4.3.3 GLCD\_Font\_16x24**

```
GLCD_FONT GLCD_Font_16x24 [extern]
```

External declaration of GLCD font

**6.4.3.4 GLCD\_Font\_6x8**

```
GLCD_FONT GLCD_Font_6x8 [extern]
```

External declaration of GLCD font

**6.4.3.5 Humidity**

```
uint8_t Humidity
```

Current humidity value.

#### 6.4.3.6 initialPage

```
bool initialPage =true
```

FFlag indicating whether to show initial page on the display.

#### 6.4.3.7 k

```
uint32_t k
```

Count value for LDR.

#### 6.4.3.8 ldr\_val

```
uint16_t ldr_val
```

Current LDR (Light Dependent Resistor) value.

#### 6.4.3.9 preset

```
char preset[9]
```

Array storing preset values.

#### 6.4.3.10 readyTransition

```
bool readyTransition =true
```

Flag indicating transition in ready display.

#### 6.4.3.11 settingsTransition

```
bool settingsTransition =true
```

Flag indicating transition in settings display.

#### 6.4.3.12 showData

```
bool showData =false
```

Flag indicating whether to show data on the display.

#### 6.4.3.13 showDataPage

```
bool showDataPage =false
```

Flag indicating whether to show data page on the display.

#### 6.4.3.14 showSettingsPage

```
bool showSettingsPage =false
```

Flag indicating whether to show settings page on the display.

#### 6.4.3.15 soilMoisturePercent

```
int8_t soilMoisturePercent
```

Current soil moisture percentage.

#### 6.4.3.16 Temperature

```
uint8_t Temperature
```

Current temperature value.

#### 6.4.3.17 tsc\_state

```
TOUCH_STATE tsc_state
```

State of touch input, containing coordinates and if pressed

#### 6.4.3.18 userTrigger

```
bool userTrigger [extern]
```

Flag for user trigger

## 6.5 C:/temp/humidtemp/GLCD.h File Reference

Header file containing the declaration of the function GLCDLogic.

### Functions

- void [GLCDLogic](#) (void)  
*GLCDLogic function.*

### 6.5.1 Detailed Description

Header file containing the declaration of the function GLCDLogic.

#### Author

Joseph Sier

This header file provides the declaration for the function GLCDLogic, which is used to control the GLCD (Graphical LCD) in the system.

## 6.5.2 Function Documentation

### 6.5.2.1 GLCDLogic()

```
void GLCDLogic (
    void )
```

GLCDLogic function.

This function is responsible for the logic to control the GLCD. It provides the necessary functionality to interact with the GLCD hardware.

## 6.6 C:/temp/humidtemp/GLCD.h

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

```
00001
00012 #ifndef GLCD_H_
00013 #define GLCD_H_
00014
00021 void GLCDLogic(void);
00022 #endif /* GLCD_H_ */
00023
00024
```

## 6.7 C:/temp/humidtemp/LDR.c File Reference

This file contains functions to read data from the Light Dependent Resistor (LDR).

```
#include "main.h"
#include "LDR.h"
```

### Functions

- uint32\_t [readLDR](#) ()  
*Reads the analog value from the LDR.*
- void [LDR\\_GetData](#) ([LDR\\_DataTypedef](#) \*LDR\_Data)  
*Get LDR data.*

### 6.7.1 Detailed Description

This file contains functions to read data from the Light Dependent Resistor (LDR).

#### Author

Joseph Sier

## 6.7.2 Function Documentation

### 6.7.2.1 LDR\_GetData()

```
void LDR_GetData (
    LDR\_DataTypedef * LDR_Data )
```

Get LDR data.

This function retrieves moisture data and stores it in the provided [LDR\\_DataTypedef](#) structure.

#### Parameters

<code>LDR_Data</code>	Pointer to the structure <a href="#">LDR_DataTypedef</a> .
-----------------------	--

#### 6.7.2.2 readLDR()

```
uint32_t readLDR ( )
```

Reads the analog value from the LDR.

#### Note

The sensor is connected to channel 6 (A3) of ADC3.

#### Returns

The value read from the LDR.

## 6.8 C:/temp/humidtemp/LDR.h File Reference

Header File Containing typedef definition.

### Data Structures

- struct [LDR\\_DataTypedef](#)  
*Struct for storing LDR data.*

### Functions

- void [LDR\\_GetData](#) ([LDR\\_DataTypedef](#) \*LDR\_Data)  
*Get LDR data.*

#### 6.8.1 Detailed Description

Header File Containing typedef definition.

#### Author

Joseph Sier

This file defines a typedef struct for managing LDR data.

#### 6.8.2 Function Documentation

##### 6.8.2.1 LDR\_GetData()

```
void LDR_GetData (
    LDR\_DataTypedef * LDR_Data )
```

Get LDR data.

This function retrieves moisture data and stores it in the provided [LDR\\_DataTypedef](#) structure.

## Parameters

<code>LDR_Data</code>	Pointer to the structure <a href="#">LDR_DataTypeDef</a> .
-----------------------	--

## 6.9 C:/temp/humidtemp/LDR.h

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

```

00001
00010 #ifndef LDR_H_
00011 #define LDR_H_
00012
00019 typedef struct
00020 {
00021     int LDR_Val;
00022 }LDR_DataTypeDef;
00023
00032 void LDR_GetData (LDR_DataTypeDef *LDR_Data);
00033
00034 #endif /* LDR_H_ */
00035
00036
00037
00038
00039

```

## 6.10 C:/temp/humidtemp/main.c File Reference

This file contains functions to read sensor data and control the display.

```

#include "Board_GLCD.h"
#include "GLCD_Config.h"
#include "stm32f7xx_hal.h"
#include "Board_Touch.h"
#include "GLCD.h"
#include "main.h"

```

### Functions

- void **SystemClock\_Config** (void)  
*Configure the system clock.*
- void **init\_config** (void)  
*All configurations to be run on start.*
- void **EXTI15\_10\_IRQHandler** (void)  
*Interrupt handler for EXTI15\_10.*
- void [HAL\\_GPIO\\_EXTI\\_Callback](#) (uint16\_t GPIO\_Pin)  
*Callback function for EXTI interrupt.*
- int **main** (void)  
*Main function.*

## Variables

- GLCD\_FONT [GLCD\\_Font\\_16x24](#)
- GPIO\_InitTypeDef [GPIO\\_InitStruct](#)
- ADC\_HandleTypeDef [hadc1](#)
- ADC\_HandleTypeDef [hadc2](#)
- ADC\_HandleTypeDef [hadc3](#)
- char [buffer](#) [128]
- bool [userTrigger](#) =true

### 6.10.1 Detailed Description

This file contains functions to read sensor data and control the display.

This file contains the main functionality for reading sensor data and controlling the display on the STM32F746G-↔ Discovery microcontroller.

Utilizes an interrupt with the user button on the back of the board to pause sensor reading, the user can pause and resume the reading instead of it being active all the time.

#### Author

Joseph Sier

### 6.10.2 Function Documentation

#### 6.10.2.1 HAL\_GPIO\_EXTI\_Callback()

```
void HAL_GPIO_EXTI_Callback (
    uint16_t GPIO_Pin )
```

Callback function for EXTI interrupt.

#### Parameters

<i>GPIO_Pin</i>	The pin number triggering the interrupt.
-----------------	--

### 6.10.3 Variable Documentation

#### 6.10.3.1 buffer

```
char buffer[128]
```

Buffer for storing data

#### 6.10.3.2 GLCD\_Font\_16x24

```
GLCD_FONT GLCD_Font_16x24 [extern]
```

External declaration of GLCD font



### 6.10.3.3 GPIO\_InitStruct

```
GPIO_InitTypeDef GPIO_InitStruct
```

GPIO initialization structure

### 6.10.3.4 hadc1

```
ADC_HandleTypeDef hadc1
```

ADC handle 1 Moisture

### 6.10.3.5 hadc2

```
ADC_HandleTypeDef hadc2
```

ADC handle 2 MQ135

### 6.10.3.6 hadc3

```
ADC_HandleTypeDef hadc3
```

ADC handle 3 LDR

### 6.10.3.7 userTrigger

```
bool userTrigger =true
```

Flag for user trigger

## 6.11 C:/temp/humidtemp/main.h File Reference

File Containing definitions for hadc, buffer and userTrigger.

```
#include <stdbool.h>
#include "stm32f7xx_hal.h"
```

### Variables

- char [buffer](#) [128]
- bool [userTrigger](#)
- ADC\_HandleTypeDef [hadc1](#)
- ADC\_HandleTypeDef [hadc2](#)
- ADC\_HandleTypeDef [hadc3](#)

### 6.11.1 Detailed Description

File Containing definitions for hadc, buffer and userTrigger.

Author

Joseph Sier

Defines string buffer and the usertrigger toggled by interrupt, as well as the ADC\_HandleTypeDef for the sensors.

### 6.11.2 Variable Documentation

#### 6.11.2.1 buffer

```
char buffer[128] [extern]
```

Buffer for storing data

#### 6.11.2.2 hadc1

```
ADC_HandleTypeDef hadc1 [extern]
```

ADC handle 1 Moisture

#### 6.11.2.3 hadc2

```
ADC_HandleTypeDef hadc2 [extern]
```

ADC handle 2 MQ135

#### 6.11.2.4 hadc3

```
ADC_HandleTypeDef hadc3 [extern]
```

ADC handle 3 LDR

#### 6.11.2.5 userTrigger

```
bool userTrigger [extern]
```

Flag for user trigger

## 6.12 C:/temp/humidtemp/main.h

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

```
00001
00011 #ifndef MAIN_H
00012 #define MAIN_H
00013
00014 #include <stdbool.h>
00015
00016 extern char buffer[128];
00017 extern bool userTrigger;
00018
00019 #endif
00020
00021 #ifndef ADC_H_
00022 #define ADC_H_
00023
00024 #include "stm32f7xx_hal.h"
00025
00026 extern ADC_HandleTypeDef hadc1;
00027 extern ADC_HandleTypeDef hadc2;
00028 extern ADC_HandleTypeDef hadc3;
00029
00030 #endif
00031
00032
```

## 6.13 C:/temp/humidtemp/Moisture.c File Reference

This file contains functions to read moisture sensor data and calculate moisture percentage.

```
#include "main.h"
#include "Moisture.h"
```

### Macros

- **#define AIR\_VALUE 698**  
*Value when the sensor is exposed to air.*
- **#define WATER\_VALUE 285**  
*Value when the sensor is submerged in water.*

### Functions

- int **map** (int x, int in\_min, int in\_max, int out\_min, int out\_max)  
*Maps a value from one range to another.*
- uint16\_t **readMoisture** ()  
*Reads the analog value from the moisture sensor.*
- int **readMoisturePercent** (void)  
*Calculates the moisture percentage based on the analog value.*
- void **Moisture\_GetData** (Moisture\_DataTypeDef \*Moisture\_Data)  
*Get moisture data.*

### 6.13.1 Detailed Description

This file contains functions to read moisture sensor data and calculate moisture percentage.

#### Author

Joseph Sier

Calibrated for the value in air (0%), and the value in water (100%). The ADC reading is then converted to a percentage within this.

I used a Capacitive sensor instead of resistive as the resistive can corrode.

### 6.13.2 Function Documentation

#### 6.13.2.1 map()

```
int map (
    int x,
    int in_min,
    int in_max,
    int out_min,
    int out_max )
```

Maps a value from one range to another.

#### Parameters

<i>x</i>	The value to map.
<i>in_min</i>	The minimum value of the input range.
<i>in_max</i>	The maximum value of the input range.
<i>out_min</i>	The minimum value of the output range.
<i>out_max</i>	The maximum value of the output range.

#### Returns

The mapped value.

#### 6.13.2.2 Moisture\_GetData()

```
void Moisture_GetData (
    Moisture\_DataTypeDef * Moisture_Data )
```

Get moisture data.

This function retrieves moisture data and stores it in the provided [Moisture\\_DataTypeDef](#) structure.

#### Parameters

<i>Moisture_Data</i>	Pointer to the structure <a href="#">Moisture_DataTypedef</a> .
----------------------	---

#### 6.13.2.3 readMoisture()

```
uint16_t readMoisture ( )
```

Reads the analog value from the moisture sensor.

#### Note

The sensor is connected to channel 7 (A2) of ADC3.

#### Returns

The analog value.

#### 6.13.2.4 readMoisturePercent()

```
int readMoisturePercent (
    void )
```

Calculates the moisture percentage based on the analog value.

#### Returns

The moisture percentage.

## 6.14 C:/temp/humidtemp/Moisture.h File Reference

Header File Containing typedef definition.

#### Data Structures

- struct [Moisture\\_DataTypedef](#)  
*Struct for storing moisture data.*

#### Functions

- void [Moisture\\_GetData](#) ([Moisture\\_DataTypedef](#) \*Moisture\_Data)  
*Get moisture data.*

### 6.14.1 Detailed Description

Header File Containing typedef definition.

Author

Joseph Sier

This file defines a typedef struct for managing moisture data.

### 6.14.2 Function Documentation

#### 6.14.2.1 Moisture\_GetData()

```
void Moisture_GetData (
    Moisture_DataTypedef * Moisture_Data )
```

Get moisture data.

This function retrieves moisture data and stores it in the provided [Moisture\\_DataTypedef](#) structure.

Parameters

<i>Moisture_Data</i>	Pointer to the structure <a href="#">Moisture_DataTypedef</a> .
----------------------	---

## 6.15 C:/temp/humidtemp/Moisture.h

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

```
00001
00011 #ifndef Moisture_H_
00012 #define Moisture_H_
00013
00020 typedef struct
00021 {
00022     int8_t MoisturePercent;
00023 } Moisture_DataTypedef;
00024
00033 void Moisture_GetData(Moisture_DataTypedef *Moisture_Data);
00034
00035 #endif /* Moisture_H_ */
00036
00037
00038
00039
```

## 6.16 C:/temp/humidtemp/mq135.c File Reference

This file contains functions to read data from the MQ135 sensor and calculate the PPM (Parts Per Million) of CO2.

```
#include "main.h"
#include "mq135.h"
```

## Macros

- `#define _rload 21.0f`  
*Load on board in kOhm.*
- `#define _rzero 1766.01f`  
*resistance at atmospheric CO2 level*
- `#define PARA 116.6020682`  
*Param for getting CO2 PPM from sensor resistance.*
- `#define PARB 2.769034857`  
*Param for getting CO2 PPM from sensor resistance.*

## Functions

- `uint32_t readMQ135 ()`  
*Reads the analog value from the MQ135 sensor.*
- `double my_pow (double base, int exponent)`  
*Custom power function.*
- `float getMQ135Resistance ()`  
*Calculates the resistance of the MQ135 sensor from the analog reading.*
- `float getMQ135PPM ()`  
*Calculates the PPM of CO2 using calibrated values.*
- `void MQ_GetData (MQ135_DataTypeDef *MQ135_Data)`  
*Get MQ135 data.*

### 6.16.1 Detailed Description

This file contains functions to read data from the MQ135 sensor and calculate the PPM (Parts Per Million) of CO2.

#### Author

Joseph Sier

The implementation is based on the MQ135 Arduino library.

The MQ135 sensor is a multi purpose sensor, but i have it configured for CO2, using 21K total resistance (\_Rload). It requires a preheat before use, which is why there is a warmup page at the beginning. It should be around 20 seconds to 2 minutes, but decreased for useability. The analog output increases when the tin oxide (SnO2) pad inside the MQ135 comes into contact with CO2, which increases its conductivity. This gives a value proportional to the gas concentration, which is converted to volts. It is calibrated using the resistance at atmospheric co2 level, but is not adjusted for temperature and humidity.

### 6.16.2 Function Documentation

#### 6.16.2.1 getMQ135PPM()

```
float getMQ135PPM ( )
```

Calculates the PPM of CO2 using calibrated values.

#### Returns

PPM of CO2.

### 6.16.2.2 getMQ135Resistance()

```
float getMQ135Resistance ( )
```

Calculates the resistance of the MQ135 sensor from the analog reading.

#### Returns

Resistance of the MQ135 sensor in volts.

### 6.16.2.3 MQ\_GetData()

```
void MQ_GetData (
    MQ135_DataTypeDef * MQ135_Data )
```

Get MQ135 data.

This function retrieves MQ135 data and stores it in the provided [MQ135\\_DataTypeDef](#) structure.

#### Parameters

<i>MQ135_Data</i>	Pointer to the structure <a href="#">MQ135_DataTypeDef</a> .
-------------------	--

### 6.16.2.4 my\_pow()

```
double my_pow (
    double base,
    int exponent )
```

Custom power function.

#### Parameters

<i>base</i>	Base value.
<i>exponent</i>	Exponent value.

#### Returns

Result of base raised to the power of exponent.

### 6.16.2.5 readMQ135()

```
uint32_t readMQ135 ( )
```

Reads the analog value from the MQ135 sensor.



**Note**

The sensor is connected to channel 8 (A1) of ADC3.

**Returns**

The analog value read from the sensor.

## 6.17 C:/temp/humidtemp/mq135.h File Reference

Header File Containing typedef definition.

**Data Structures**

- struct [MQ135\\_DataTypeDef](#)  
*Struct for storing MQ135 data.*

**Functions**

- void [MQ\\_GetData](#) ([MQ135\\_DataTypeDef](#) \*MQ135\_Data)  
*Get MQ135 data.*

### 6.17.1 Detailed Description

Header File Containing typedef definition.

**Author**

Joseph Sier

This file defines a typedef struct for managing MQ135 data.

### 6.17.2 Function Documentation

#### 6.17.2.1 MQ\_GetData()

```
void MQ_GetData (
    MQ135\_DataTypeDef * MQ135_Data )
```

Get MQ135 data.

This function retrieves MQ135 data and stores it in the provided [MQ135\\_DataTypeDef](#) structure.

**Parameters**

<i>MQ135_Data</i>	Pointer to the structure <a href="#">MQ135_DataTypeDef</a> .
-------------------	--

## 6.18 C:/temp/humidtemp/mq135.h

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

```
00001
00010 #ifndef MQ135_H_
00011 #define MQ135_H_
00012
00020 typedef struct
00021 {
00022     uint16_t PPM;
00023 }MQ135_DataTypeDef;
00024
00034 void MQ_GetData (MQ135_DataTypeDef *MQ135_Data);
00035
00036 #endif /* MQ135_H_ */
00037
00038
00039
00040
00041
```

# Index

buffer  
    main.c, [26](#)  
    main.h, [28](#)  
Bug List, [3](#)

C:/temp/humidtemp/DHT.c, [13](#)  
C:/temp/humidtemp/DHT.h, [16](#)  
C:/temp/humidtemp/GLCD.c, [17](#)  
C:/temp/humidtemp/GLCD.h, [22](#)  
C:/temp/humidtemp/LDR.c, [23](#)  
C:/temp/humidtemp/LDR.h, [24](#)  
C:/temp/humidtemp/main.c, [25](#)  
C:/temp/humidtemp/main.h, [27](#)  
C:/temp/humidtemp/Moisture.c, [29](#)  
C:/temp/humidtemp/Moisture.h, [31](#)  
C:/temp/humidtemp/mq135.c, [32](#)  
C:/temp/humidtemp/mq135.h, [35](#)  
checkAndDisplayStatus  
    GLCD.c, [19](#)

dataTransition  
    GLCD.c, [20](#)

DHT.c  
    DHT\_Check\_Response, [14](#)  
    DHT\_GetData, [14](#)  
    DHT\_Read, [15](#)  
    DHT\_Start, [15](#)  
    DWT\_Delay\_Init, [15](#)  
    DWT\_Delay\_us, [15](#)  
    Set\_Pin\_Input, [16](#)  
    Set\_Pin\_Output, [16](#)

DHT.h  
    DHT\_GetData, [17](#)

DHT\_Check\_Response  
    DHT.c, [14](#)

DHT\_DataTypedef, [9](#)  
    Humidity, [9](#)  
    Temperature, [9](#)

DHT\_GetData  
    DHT.c, [14](#)  
    DHT.h, [17](#)

DHT\_Read  
    DHT.c, [15](#)

DHT\_Start  
    DHT.c, [15](#)

displayStatus  
    GLCD.c, [19](#)

drawBars  
    GLCD.c, [20](#)

DWT\_Delay\_Init  
    DHT.c, [15](#)  
    DWT\_Delay\_us  
        DHT.c, [15](#)

getMQ135PPM  
    mq135.c, [33](#)

getMQ135Resistance  
    mq135.c, [33](#)

GLCD.c  
    checkAndDisplayStatus, [19](#)  
    dataTransition, [20](#)  
    displayStatus, [19](#)  
    drawBars, [20](#)  
    GLCD\_Font\_16x24, [20](#)  
    GLCD\_Font\_6x8, [20](#)  
    GLCDLogic, [20](#)  
    Humidity, [20](#)  
    initialPage, [20](#)  
    k, [21](#)  
    ldr\_val, [21](#)  
    preset, [21](#)  
    readyTransition, [21](#)  
    settingsTransition, [21](#)  
    showData, [21](#)  
    showDataPage, [21](#)  
    showSettingsPage, [21](#)  
    soilMoisturePercent, [22](#)  
    Temperature, [22](#)  
    tsc\_state, [22](#)  
    userTrigger, [22](#)

GLCD.h  
    GLCDLogic, [23](#)

GLCD\_Font\_16x24  
    GLCD.c, [20](#)  
    main.c, [26](#)

GLCD\_Font\_6x8  
    GLCD.c, [20](#)

GLCDLogic  
    GLCD.c, [20](#)  
    GLCD.h, [23](#)

GPIO\_InitStruct  
    main.c, [26](#)

hadc1  
    main.c, [27](#)  
    main.h, [28](#)

hadc2  
    main.c, [27](#)  
    main.h, [28](#)

hadc3

- main.c, [27](#)
- main.h, [28](#)
- HAL\_GPIO\_EXTI\_Callback
  - main.c, [26](#)
- Humidity
  - DHT\_DataTypeDef, [9](#)
  - GLCD.c, [20](#)
- initialPage
  - GLCD.c, [20](#)
- k
  - GLCD.c, [21](#)
- LDR.c
  - LDR\_GetData, [23](#)
  - readLDR, [24](#)
- LDR.h
  - LDR\_GetData, [24](#)
- LDR\_DataTypeDef, [10](#)
  - LDR\_Val, [10](#)
- LDR\_GetData
  - LDR.c, [23](#)
  - LDR.h, [24](#)
- LDR\_Val
  - LDR\_DataTypeDef, [10](#)
- ldr\_val
  - GLCD.c, [21](#)
- main.c
  - buffer, [26](#)
  - GLCD\_Font\_16x24, [26](#)
  - GPIO\_InitStruct, [26](#)
  - hadc1, [27](#)
  - hadc2, [27](#)
  - hadc3, [27](#)
  - HAL\_GPIO\_EXTI\_Callback, [26](#)
  - userTrigger, [27](#)
- main.h
  - buffer, [28](#)
  - hadc1, [28](#)
  - hadc2, [28](#)
  - hadc3, [28](#)
  - userTrigger, [28](#)
- map
  - Moisture.c, [30](#)
- Microclimate manager, [1](#)
- Moisture.c
  - map, [30](#)
  - Moisture\_GetData, [30](#)
  - readMoisture, [31](#)
  - readMoisturePercent, [31](#)
- Moisture.h
  - Moisture\_GetData, [32](#)
- Moisture\_DataTypeDef, [10](#)
  - MoisturePercent, [11](#)
- Moisture\_GetData
  - Moisture.c, [30](#)
  - Moisture.h, [32](#)
- MoisturePercent
  - Moisture\_DataTypeDef, [11](#)
- mq135.c
  - getMQ135PPM, [33](#)
  - getMQ135Resistance, [33](#)
  - MQ\_GetData, [34](#)
  - my\_pow, [34](#)
  - readMQ135, [34](#)
- mq135.h
  - MQ\_GetData, [35](#)
- MQ135\_DataTypeDef, [11](#)
  - PPM, [11](#)
- MQ\_GetData
  - mq135.c, [34](#)
  - mq135.h, [35](#)
- my\_pow
  - mq135.c, [34](#)
- PPM
  - MQ135\_DataTypeDef, [11](#)
- preset
  - GLCD.c, [21](#)
- readLDR
  - LDR.c, [24](#)
- readMoisture
  - Moisture.c, [31](#)
- readMoisturePercent
  - Moisture.c, [31](#)
- readMQ135
  - mq135.c, [34](#)
- readyTransition
  - GLCD.c, [21](#)
- Set\_Pin\_Input
  - DHT.c, [16](#)
- Set\_Pin\_Output
  - DHT.c, [16](#)
- settingsTransition
  - GLCD.c, [21](#)
- showData
  - GLCD.c, [21](#)
- showDataPage
  - GLCD.c, [21](#)
- showSettingsPage
  - GLCD.c, [21](#)
- soilMoisturePercent
  - GLCD.c, [22](#)
- Temperature
  - DHT\_DataTypeDef, [9](#)
  - GLCD.c, [22](#)
- tsc\_state
  - GLCD.c, [22](#)
- userTrigger
  - GLCD.c, [22](#)
  - main.c, [27](#)
  - main.h, [28](#)