LED-Clock

Release 2.0.0

Florian Laschober

Jun 25, 2023

CONTENTS

This is the software of a variation of the LED clock from DIY Machines A configuration which works on the original version of the clock is also part of the source code.

I decided to code it completely from scratch at I wanted to use a ESP32 instead of the Arduino nano and RTC that is used in the original project. This enables some cool features like smartphone app control, OTA updates and fetching of the time using the internet.

Additionally this has full support for animations. By default I provide all needed animations for a 12h clock to morph one digit into another soothly. Further animations can be easily added and existing animation can be adjusted to your liking.

The whole codebase is highly modular and configurable and can be tweaked exactly to your preferences.

Detailed documentation is available on read the docs.

If you are interested in my variation of the design which uses a lot of wood instead of the 3D prints and is a bit bigger than the original you can find it on thingiverse here

CHAPTER

ONE

DEVELOPMENT ENVIRONMENT

I am using VScode with PlatformIO. VSCode can be downloaded from here. And PlatformIO is an extension that can easily be installed from inside of VSCode.

Via the PlatformIO home the following libraries have to be installed:

- “Blynk” by Volodymyr Shymanskyy
- “FastLED” by Daniel Garcia
- “LinkedList” by Ivan Seidel

CHAPTER

TWO

GETTING STARTED

A quick list of things that have to be done to get this project up and running:

1. Make sure you have VS Code and the PlatformIO extension installed
2. Download the source either via the releases tab or by downloading or cloning this git repository
3. Go to lib/LED*clock and choose one of the library**.json files and rename it to library.json
4. Check that you have all the above mentioned Platform IO libraries installed either system wide or in the project itself.
5. If you want to modify any of the configuration options you can do so by editing the files in `lib/LED_clock/Config/Setup/<chosen_version>`
6. Build and upload

If you want a more detailed walk through of the installation process take a look at the setup wiki page.

CHAPTER

THREE

SUPPORT

First of all thanks to DIY machines for the awesome idea!

Also thank you to the great people who developed the libraries that I am using for this project:

- Volodymyr Shymanskyy for the Blynk library
- Daniel Garcia for the FastLED library
- Ivan Seidel for his LinkedList implementation
- Andy Brown for the Easings library that I modified a bit to suit my needs better

If you would like to say thank you:

3.1 Where to go from here

3.1.1 Wiring setup

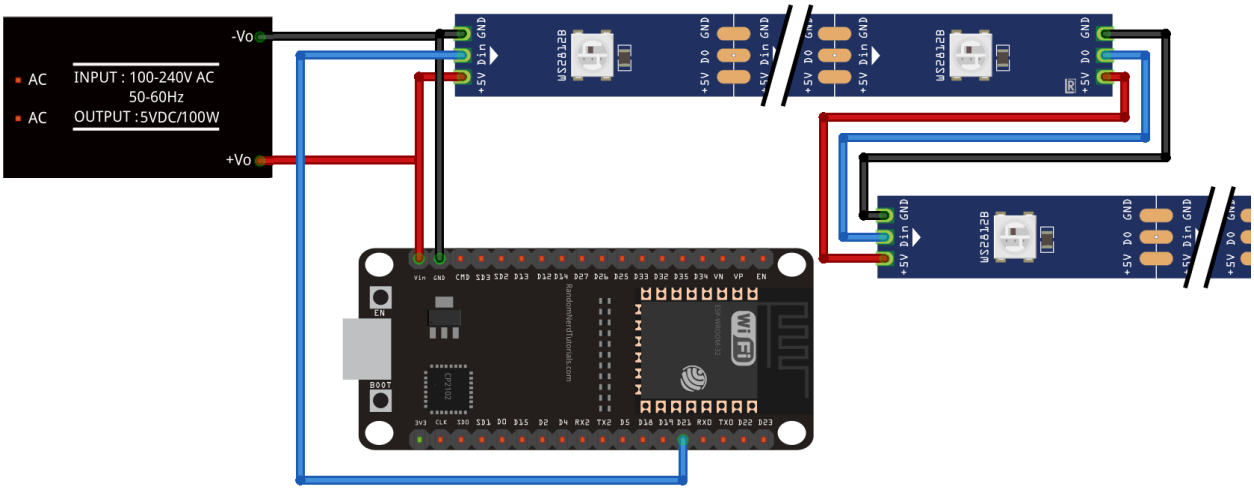
Wiring of the ESP32 is highly customizable. This can be changed and tweaked very easily by modifying the respective *main configuration* file for your chosen config.

The following table are the default values that the code comes pre configured with:

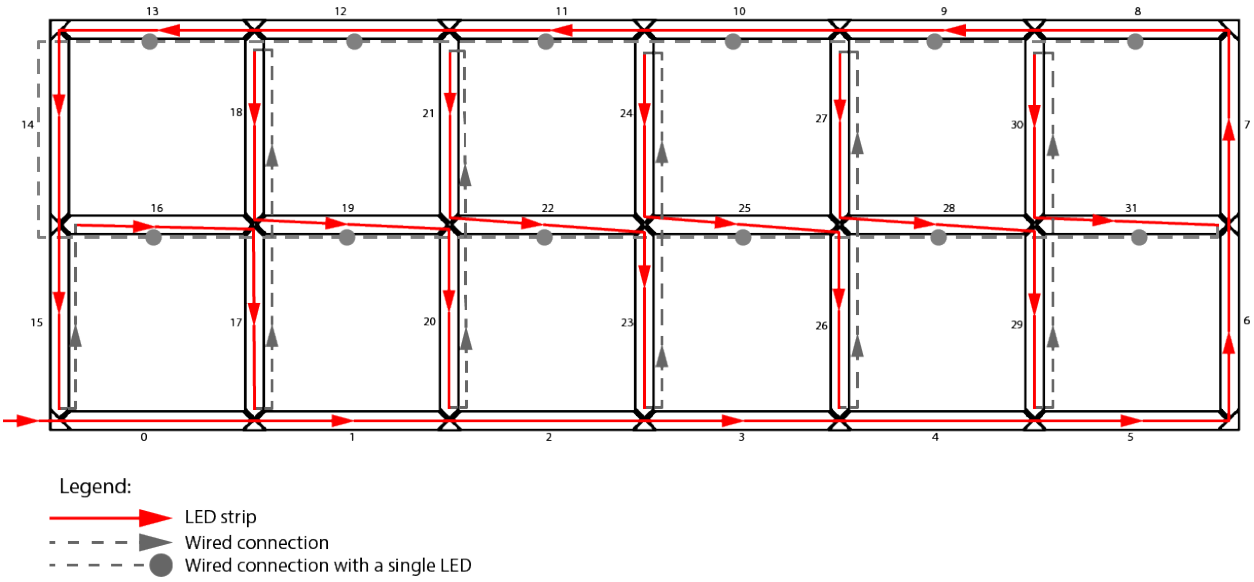
Connection	Pin	Config parameter	Comment
LED strip data	21	`LED_DATA_PIN`	(MANDATORY) This is the pin to which the LED strip is connected to.
Internal LED strip data	22	`DOWNLIGHT_LED_DATA_PIN`	(OPTIONAL) This is the pin to which the LED strip of the internal dowlighter LED's is connected to in case they are separated. This only takes effect if `APPEND_DOWN_LIGHTERS` Is set to `false`.
Light sensor	34	`LIGHT_SENSOR_PIN`	(OPTIONAL) This is the pin to which the light sensor is connected in case it is enabled by setting `ENABLE_LIGHT_SENSOR` to `true`.

Default wiring

This is the minimal wiring diagram according to the default configuration:



The WS2812B LED Strips should be wired together by connecting the pads like shown on the diagram above. The Connections of the LED strips in the default config is done like this:



Power Supply specs

The +5V and GND connections of the LED strip should be connected straight to a suitable 5V power supply.
 The ESP32 can also be connected straight to the power supply, just make sure to connect the +5V to the VIN pin and NOT the 3V pin!!
 The required wattage/Max current rating needed can be easily calculated:
 According to the manufacturer of the WS2813B LED's each LED consumes a maximum of 0.24W per piece. (figure taken form here.).

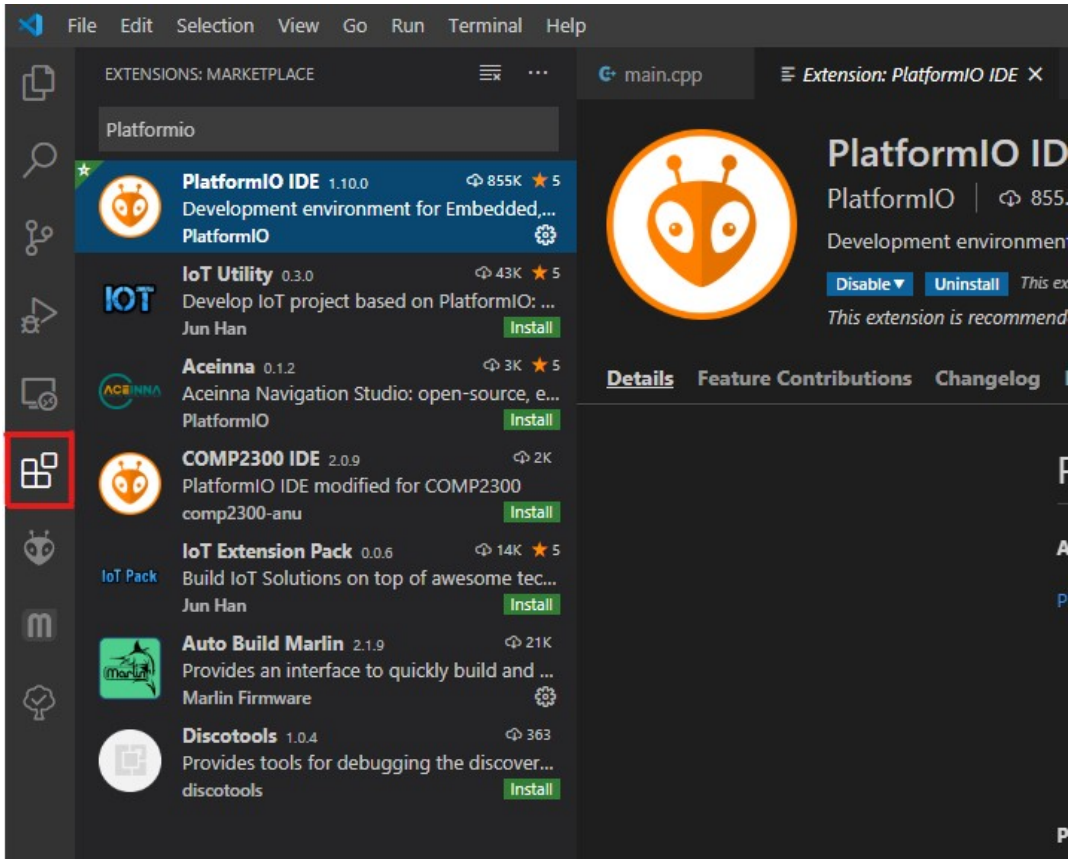
This means for the default configuration (32 Segments with 12 LED's each and an additional 12 downlighter LED's The power supply should at least be able to handle (32 * 12 + 12)*0.24 = 95W which is around 20A at 5V to have a little bit of headroom.

3.1.2 The Development environment

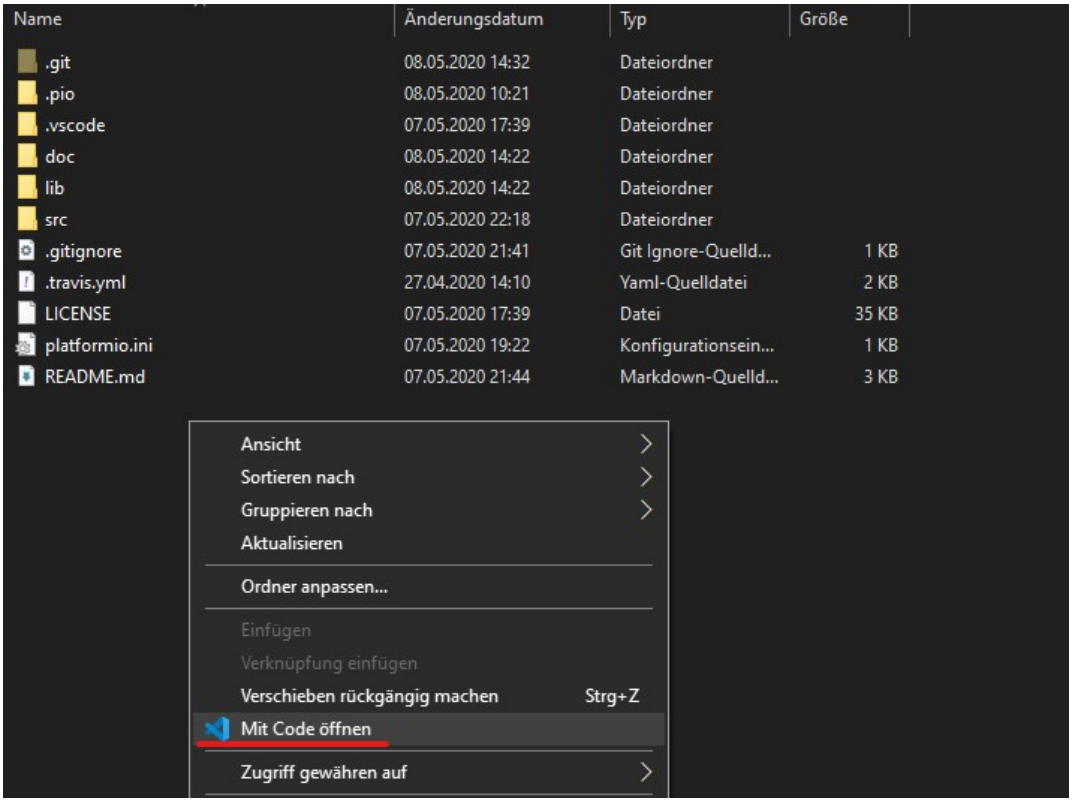
I find the default Arduino IDE very hard to use if programming more than just basic functionality. This is why I decided to code all of this using Visual Studio code with PlatformIO. It is free, easy to install, really powerful and almost as easy to use as the Arduino IDE.

Step by step installation guide

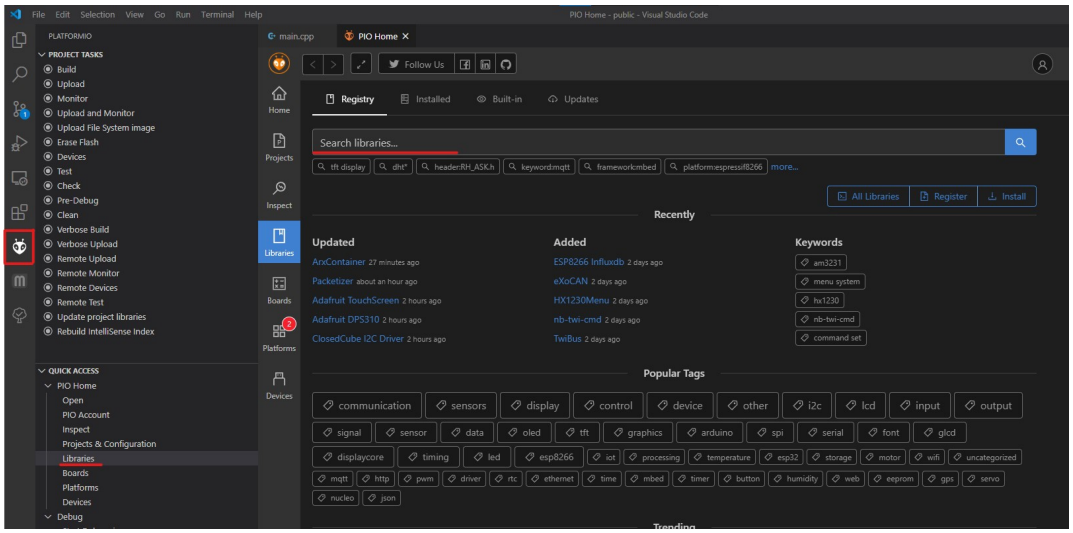
1. **Install VSCode from the official microsoft page here.**
   To make things a bit easier you can enable the right click menu entry option during the VSCode installation.
2. Start VSCode and install PlatformIO from the extension menu in the sidebar.



3. Now download or clone the code from github, and put it to your hard drive. (Unzip it if you downloaded it)
4. Navigate to the folder where you put your code and right click to open the context menu and choose open VSCode here:



5. After waiting for all the plugins to load you should see a PlatformIO button in your sidebar. Click on it and choose Libraries:



6. From here install the following libraries:

   - "Blynk" by Volodymyr Shymanskyy
   - "FastLED" by Daniel Garcia
   - "LinkedList" by Ivan Seidel

7. Now navigate to the `lib/LED_clock` folder and choose one of the `library_*.json` files and rename it to `library.json`. There are multiple already pre configured versions available. Simply choose the one which is closest to your hardware.

8. If needed you can modify the configuration parameters by editing the files in `lib/LED_clock/Config/Setup/<chosen_version>` where `<chosen_version>` should be the version of the json file you chose in step 7.
9. Now you can test if the build is working by hitting the little check mark icon in the bottom bar of VScode. Uploading to the target is done by clicking the arrow button, just like in the Arduino IDE. You don't need to worry about your COM port as PlatformIO is smart enough to figure it out automatically.
10. For information on how to setup WIFI and Blynk take a look at the *The first Startup*

3.1.3 The first Startup

After the code has been flashed to the ESP it's time to test it. Plug all the LEDs into the ESP and power it on. You will be greeted with a nice loading animation on the LEDs of your clock.

Note: This table only applies if you did not change any of the default colors

---

The color of the animation determines the state that the ESP is in:

Color	Meaning
Blue	ESP is trying to connect to WIFI
Orange	ESP is in smart config mode and waiting for WIFI credentials
Red	Connection to the WIFI network was unsuccessful

Note: This only applies if you did not turn off the `smart config` option.

- If you had the ESP you used connected to your WIFI before it will automatically reconnect and after a few seconds you should see the time showing up.
- If you are using a new ESP that does not know your WIFI details yet wait a few seconds until the LEDs indicate that the system entered smart config mode.
- Once in smart config mode you can use for example the EspTouch app (only available on android) to let the ESP know your wifi credentials.
- After a successful connection the Clock will be displaying the current time.

Note: If you would like to rather not use smart config you can also still go the good old way of hardcoding the WIFI name and password into the code. You can change this setting in your Configuration.h file.

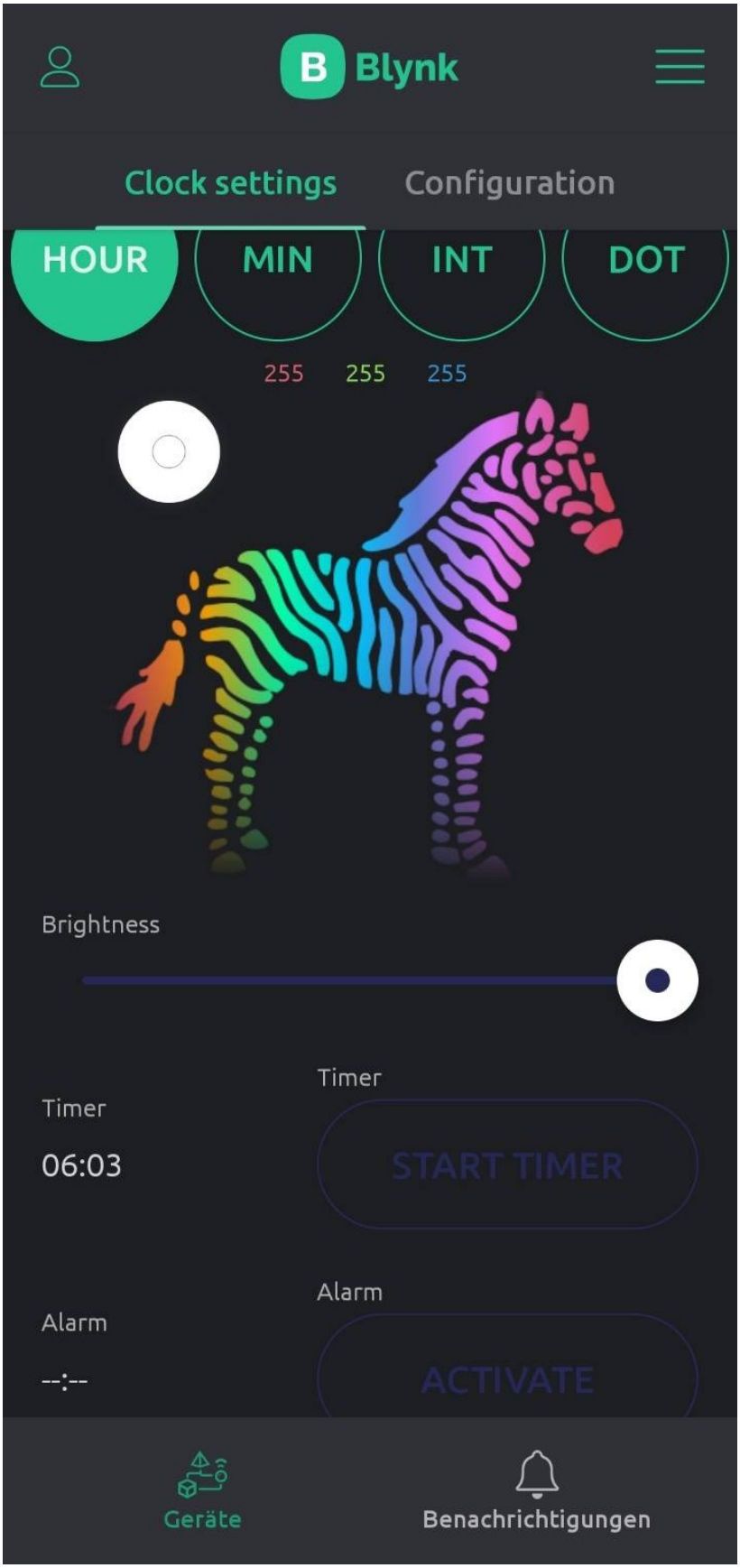
- If you want to use the Blynk functionality do not forget to set your blynk auth token and template ID in your Configuration.h file.

3.1.4 Default blynk config

This configuration is supposed to give all the basic functionality while also staying inside the free tier.

This guide will cover how to setup a dashboard and the datastreams for the new Blynk 2.0

An example of how the dashboard will look like:


Blynk
Clock settings
Configuration
HOUR
MIN
INT
DOT
255
255
255
Brightness
Timer
Timer
06:03
START TIMER
Alarm
Alarm
--:--
ACTIVATE
Geräte
Benachrichtigungen

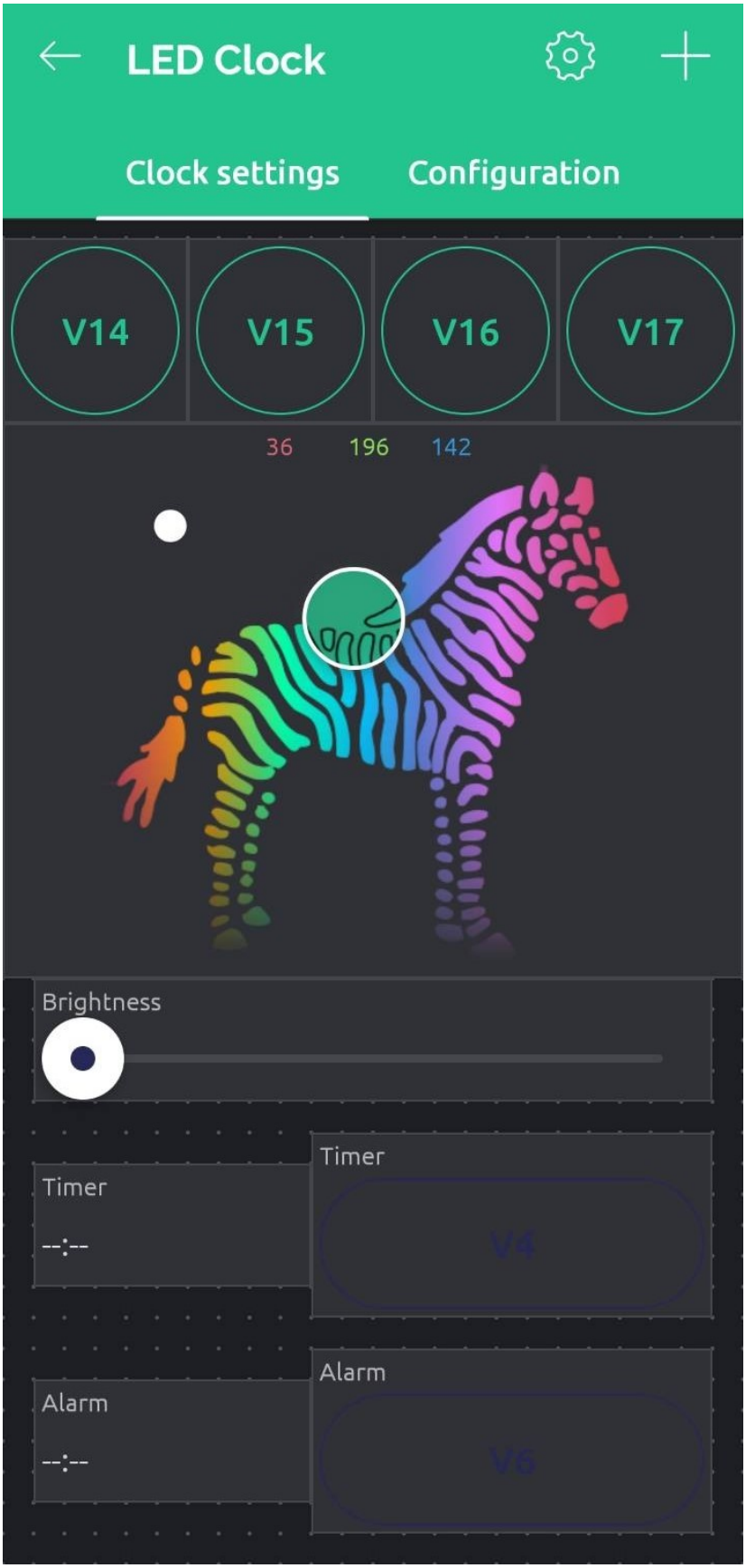
Setup guide

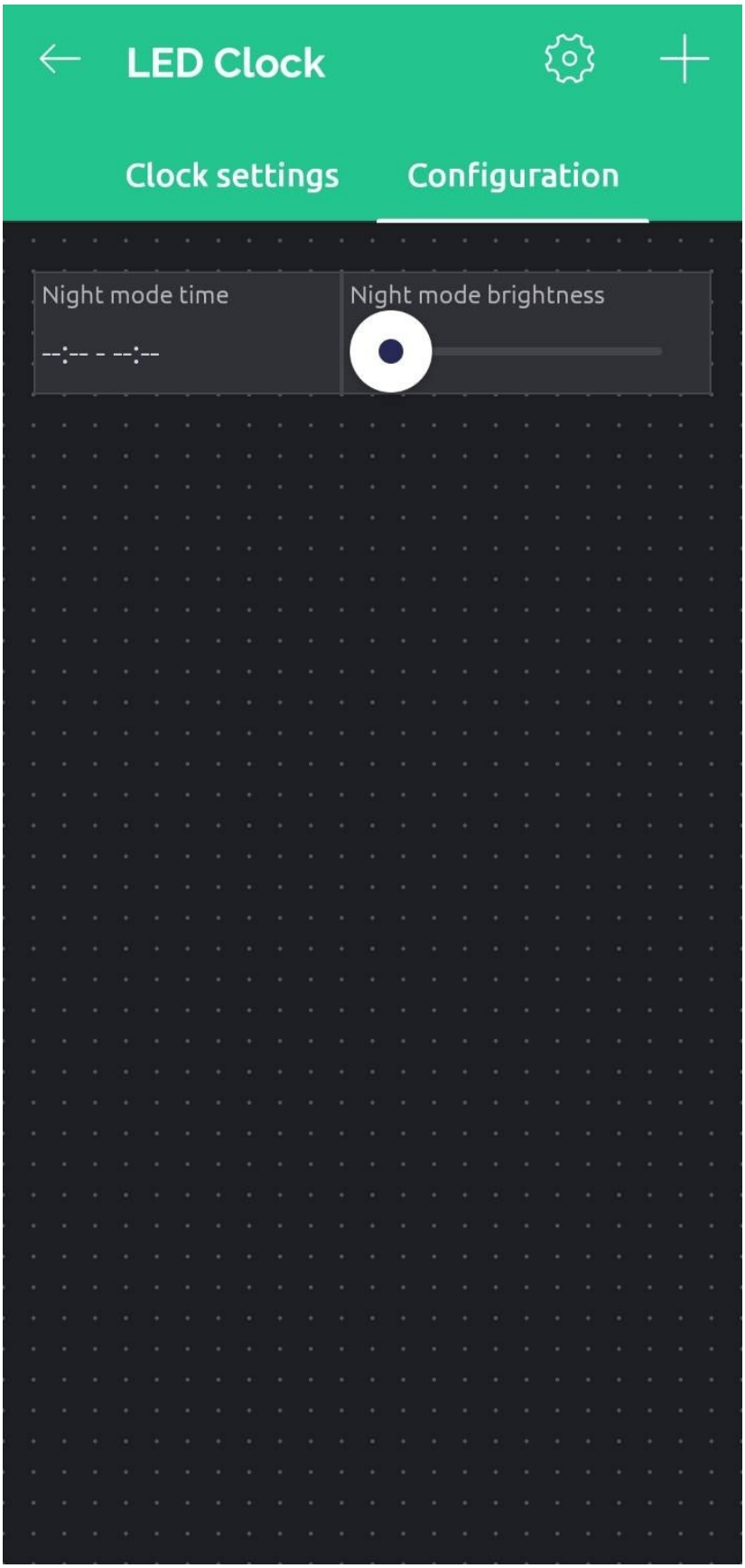
Sadly I could not find any way to export my own template, make it public, or share it in any way shape or form with the new blynk cloud. It seems like this is just simply not possible. So for now I will provide some instructions here on how to replicate my dashboard manually. If someone knows how to share templates on the new blynk cloud please let me know!

1. Follow the first setup instructions in the blynk cloud interface to create a new device and template.
2. Then in the web app open the template that you just created and go to the Datastreams tab. Here you should create the following datastreams

Name	Pin	Data Type	Min	Max	Default
LED Brightness	V0	Integer	0	255	128
Light group selector	V1	Integer	0	4	0
Current Color	V2	String	•	•	•
Timer time	V3	String	•	•	•
Timer start button	V4	Integer	0	1	0
Alarm time	V5	String	•	•	•
Alarm start button	V6	Integer	0	1	0
Night mode time	V7	String	•	•	•
Night mode Brightness	V8	Integer	0	255	0
Number of separation dots	V9	Integer	0	2	0
Hour color	V10	String	•	•	•
Minute color	V11	String	•	•	•
Internal color	V12	String	•	•	•
Dot color	V13	String	•	•	•
Selector Hours	V14	Integer	0	1	0
Selector Minutes	V15	Integer	0	1	0
Selector Internal	V16	Integer	0	1	0
Selector Dot	V17	Integer	0	1	0

1. Now you can start replicating the Dashboard in the app. Here are some screenshots of it. You can see most of the pin associations in the screenshot itself:


LED Clock
Clock settings
Configuration
V14
V15
V16
V17
36
196
142
Brightness
Timer
--:--
Timer
V4
Alarm
--:--
Alarm
V6


LED Clock
Clock settings
Configuration
Night mode time
--:-- - --:--
Night mode brightness

Some settings which are not obvious from the screenshots above:

Hour/min/int/dot selectors:
Should be set to toggle

zeRGBa:
Pin: V2

Send on release: false

Mode: Merge

send interval: 300ms

Note: Workaround for iOS users:

It seems like blynk has a bug in their RGB widget that will prevent you from selecting the correct virtual pin. To get it working you can follow this workaround:

1. Set the data stream type for V2 to integer in the blynk web dashboard and apply the change
2. Select V2 as the data stream for the RGB widget and save the changes
3. Change the data stream type of the V2 pin back to String via the web dashboard, save and apply changes

---

Brightness Slider:
Pin: V0

Send on release: false

send interval: 300ms

Timer time selector:
Pin: V3

Format: HH:mm

All other options are set to false

Alarm time selector:
Pin: V5

Format: HH:mm

All other options are set to false

Alarm/timer buttons:
Should be set to toggle

Night mode time selector:
Pin: V7

Switch start/stop time input to true

Night time brightness slider:
Pin: V8

Send on release: false

send interval: 300ms

Selection buttons:
Pins: V14 - V17

Behavior: Toggle

3.1.5 General Configuration

Configuration parameters

group **MainConfiguration**

Main configuration settings.

Defines

RUN_WITHOUT_WIFI

If you want to run the system in a minimal mode to test some basic functionality or debug something it could be useful to disable wifi functionality completely.

IS_BLYNK_ACTIVE

If you want Blynk functionality set this to true and set your authentication token. Otherwise set it to false.

BLYNK_AUTH_TOKEN

If you want Blynk functionality paste your authentication token here.

BLYNK_TEMPLATE_ID

Template ID for this device. If you want to use your own custom Template you will have to change this.

BLYNK_DEVICE_NAME

Name of this device in the Blynk app.

BLYNK_PRINT

In case the blynk communication is not working this line causes Blynk to send debug output to the serial port. If you are not worried about Blynk or have to diagnose some other issue you can comment this line out.

BLYNK_SERVER

Set the Blynk server address.

Note: I had troubles with using the proper blynk domain so I am using the IP address instead. Maybe this could create problems in the future so it is recommended to use the official domain.

ENABLE_OTA_UPLOAD

If you want to use OTA upload instead or in addition to the normal cable upload set this option to true. To actually flash something via OTA you have to uncomment the OTA flash lines in the platformio.ini file This is a nice addition to cable upload but it doesn't replace it completely. If the microcontroller crashes because of bad configuration you still have to use a cable.

OTA_UPDATE_HOST_NAME

The host name that shall be used for OTA updates. If you change this here it must also be changed in the platformio.ini file.

NUM_RETRIES

The number of times the controller tries to connect to wifi before it fails and goes into smartConfig mode (if that is enabled)

USE_ESPTOUCH_SMART_CONFIG

Use the ESP smart config to setup the wifi network. If you want to set it manually set this to false.

WIFI_SSID

WIFI_SSID and WIFI_PW are only needed if smart setup is disabled.

WIFI_PW

HOUR_COLOR

Color of the hour segments, this will be the default color if blynk functionality is disabled.

MINUTE_COLOR

Color of the minute segments, this will be the default color if blynk functionality is disabled.

INTERNAL_COLOR

Color of the internal LEDs, this will be the default color if blynk functionality is disabled.

SEPARATION_DOT_COLOR

Color of the separation dot LEDs, this will be the default color if blynk functionality is disabled.

OTA_UPDATE_COLOR

Color of the LEDs for the OTA update progress bar.

WIFI_CONNECTING_COLOR

Color of the LEDs while searching for a WIFI network.

WIFI_CONNECTION_SUCCESSFUL_COLOR

Color of the LEDs signaling a successful WIFI connection.

WIFI_SMART_CONFIG_COLOR

Color of the LEDs if system is waiting for WIFI smart config.

ERROR_COLOR

Color of the LEDs signaling an error of some sort.

NTP_SERVER

Server for the time.

TIMEZONE_INFO

Enter the string for your timezone according to this webpage: https://remotemonitoringsystems.ca/time-zone-abbreviations.php.

TIME_SYNC_INTERVAL

Time in seconds for the interval in which the time should be synchronized with the time server.

TIMER_FLASH_TIME

Flash the current time in case a timer is expired instead of flashing 00:00.

TIMER_FLASH_COUNT

Number of flashes until an alarm is considered complete and the system goes back to normal.

ALARM_NOTIFICATION_PERIOD

For how long the Display should flash when an alarm was fired in seconds.

NOTIFICATION_BRIGHTNESS

How bright the clock should blink when an alarm or timer was triggered 0 - 255.

TIME_UPDATE_INTERVAL

How often the time is checked and the displays are updated.

DEFAULT_CLOCK_BRIGHTNESS

Default brightness of the display. If you are using blynk you may ignore this setting.

USE_NIGHT_MODE

Whether to activate night mode or not. If you want the clock to reduce brightness/switch off during certain hours set this to true. If you are using Blynk to control the settings of your clock you may ignore the default settings as they can be changed dynamically during runtime in that case.

DEFAULT_NIGHT_MODE_START_HOUR

Start hour for the night mode.

DEFAULT_NIGHT_MODE_START_MINUTE

Start minute for the night mode.

DEFAULT_NIGHT_MODE_END_HOUR

End hour for the night mode.

DEFAULT_NIGHT_MODE_END_MINUTE

End minute for the night mode.

DEFAULT_NIGHT_MODE_BRIGHTNESS

Brightness that the clock should be set to while night mode is active.

LED_DATA_PIN

Pin to which the led strip data pin is connected to.

NUM_SEGMENTS

Total number of segments that have LEDs in the shelf.

NUM_LEDS_PER_SEGMENT

Number of LEDs in each segment.

APPEND_DOWN_LIGHTERS

If you wired the down lighter LEDs to the end of the LED strips set this to true.

ADDITIONAL_LEDS

Number of LEDs For interior lights.

NUM_LEDS

Automatically calculated total number of LEDs used.

NUM_DISPLAYS

Number of displays in the shelf.

DISPLAY_0_AT_MIDNIGHT

If set to true the display will show 0 at midnight and 12 otherwise.

DISPLAY_SWITCH_OFF_AT_0

If set to true the higher displays will turn off in case they would show 0.

USE_24_HOUR_FORMAT

If set to true 24 hour format will be used. For this one additional column is needed in the shelf to display it correctly.

NUM_SEGMENTS_PROGRESS

The number of segments to use for displaying a progress bar for the OTA updates.

LOADING_ANIMATION_DURATION

The time is shall take for one iteration of the loading animation.

BRIGHTNESS_INTERPOLATION

How fast the brightness interpolation shall react to brightness changes.

DISPLAY_FOR_SEPARATION_DOT

If set to -1 the flashing middle dot is disabled, otherwise this is the index of the Display segment that should display the dot.

ANIMATION_TARGET_FPS

Target Frames per second for the smoothness of animations.

ANIMATION_AFTERGLOW

Length of sooth animation transition from fully on to black and vice versa in percent NOTE: The higher this number the less obvious easing effects like bounce or elastic will be.

DOT_FLASH_SPEED

Length of the dot/s fading animation. One flash fades in and out.

DOT_FLASH_INTERVAL

Interval in which the dot/s should flash.

NUM_SEPARATION_DOTS

Number of separation dots to use by default (or if no blynk functionality is available) allowed values are 1, 2 and 0 to turn it off.

ENABLE_LIGHT_SENSOR

Enable automatic brightness adjustments based on a light sensor.

LIGHT_SENSOR_PIN

ADC pin to which the light sensor is connected to.

LIGHT_SENSOR_AVERAGE

How many measurements shall be averaged. Higher number -> smoother but slower change.

LIGHT_SENSOR_MEDIAN_WIDTH

Width of the median calculation. Higher number -> smoother change Should never be higher than the LIGHT_SENSOR_AVERAGE.

LIGHT_SENSOR_READ_DELAY

Time that should pass before the light sensor is read again. Higher number -> slower adjustments but also changes will be more sudden.

LIGHT_SENSOR_MIN

AnalogRead value if the light sensor reads complete darkness.

LIGHT_SENSOR_MAX

AnalogRead value if the light sensor reads the brightest.

LIGHT_SENSOR_SENSITIVITY

Value between 0 and 255 that determines how much the light sensor values can influence the led brightness.

TIME_MANAGER_DEMO_MODE

enable for wifi less operation or to demo all the animations

DIGIT_ANIMATION_SPEED

The time it takes for one digit to morph into another.

FASTLED_SAFE_DELAY_MS

the minimum delay between calls of FastLED.show()

RUN_WITHOUT_WIFI

If you want to run the system in a minimal mode to test some basic functionality or debug something it could be useful to disable wifi functionality completely.

IS_BLYNK_ACTIVE

If you want Blynk functionality set this to true and set your authentication token. Otherwise set it to false.

BLYNK_AUTH_TOKEN

If you want Blynk functionality paste your authentication token here.

BLYNK_TEMPLATE_ID

Template ID for this device. If you want to use your own custom Template you will have to change this.

BLYNK_DEVICE_NAME

Name of this device in the Blynk app.

BLYNK_PRINT

In case the blynk communication is not working this line causes Blynk to send debug output to the serial port. If you are not worried about Blynk or have to diagnose some other issue you can comment this line out.

BLYNK_SERVER

Set the Blynk server address.

ENABLE_OTA_UPLOAD

If you want to use OTA upload instead or in addition to the normal cable upload set this option to true. To actually flash something via OTA you have to uncomment the OTA flash lines in the platformio.ini file This is a nice addition to cable upload but it doesn't replace it completely. If the microcontroller crashes because of bad configuration you still have to use a cable.

OTA_UPDATE_HOST_NAME

The host name that shall be used for OTA updates. If you change this here it must also be changed in the platformio.ini file.

NUM_RETRIES

The number of times the controller tries to connect to wifi before it fails and goes into smartConfig mode (if that is enabled)

USE_ESPTOUCH_SMART_CONFIG

Use the ESP smart config to setup the wifi network. If you want to set it manually set this to false.

WIFI_SSID

WIFI_SSID and WIFI_PW are only needed if smart setup is disabled.

WIFI_PW

HOUR_COLOR

Color of the hour segments, this will be the default color if blynk functionality is disabled.

MINUTE_COLOR

Color of the minute segments, this will be the default color if blynk functionality is disabled.

INTERNAL_COLOR

Color of the internal LEDs, this will be the default color if blynk functionality is disabled.

SEPARATION_DOT_COLOR

Color of the separation dot LEDs, this will be the default color if blynk functionality is disabled.

OTA_UPDATE_COLOR

Color of the LEDs for the OTA update progress bar.

WIFI_CONNECTING_COLOR

Color of the LEDs while searching for a WIFI network.

WIFI_CONNECTION_SUCCESSFUL_COLOR

Color of the LEDs signaling a successful WIFI connection.

WIFI_SMART_CONFIG_COLOR

Color of the LEDs if system is waiting for WIFI smart config.

ERROR_COLOR

Color of the LEDs signaling an error of some sort.

NTP_SERVER

Server for the time.

TIMEZONE_INFO

Enter the string for your timezone according to this webpage: https://remotemonitoringsystems.ca/time-zone-abbreviations.php.

TIME_SYNC_INTERVAL

Time in seconds for the interval in which the time should be synchronized with the time server.

TIMER_FLASH_TIME

Flash the current time in case a timer is expired instead of flashing 00:00.

TIMER_FLASH_COUNT

Number of flashes until an alarm is considered complete and the system goes back to normal.

ALARM_NOTIFICATION_PERIOD

For how long the Display should flash when an alarm was fired in seconds.

NOTIFICATION_BRIGHTNESS

How bright the clock should blink when an alarm or timer was triggered : 0 - 255.

TIME_UPDATE_INTERVAL

How often the time is checked and the displays are updated.

DEFAULT_CLOCK_BRIGHTNESS

Default brightness of the display. If you are using blynk you may ignore this setting.

USE_NIGHT_MODE

Whether to activate night mode or not. If you want the clock to reduce brightness/switch off during certain hours set this to true. If you are using Blynk to control the settings of your clock you may ignore the default settings as they can be changed dynamically during runtime in that case.

DEFAULT_NIGHT_MODE_START_HOUR

Start hour for the night mode.

DEFAULT_NIGHT_MODE_START_MINUTE

Start minute for the night mode.

DEFAULT_NIGHT_MODE_END_HOUR

End hour for the night mode.

DEFAULT_NIGHT_MODE_END_MINUTE

End minute for the night mode.

DEFAULT_NIGHT_MODE_BRIGHTNESS

Brightness that the clock should be set to while night mode is active.

LED_DATA_PIN

Pin to which the led strip data pin is connected to.

NUM_SEGMENTS

Total number of segments that have LEDs in the shelf.

NUM_LEDS_PER_SEGMENT

Number of LEDs in each segment.

APPEND_DOWN_LIGHTERS

If you wired the down lighter LEDs to the end of the LED strips set this to true.

ADDITIONAL_LEDS

Number of LEDs For interior lights.

NUM_LEDS

Automatically calculated total number of LEDs used.

NUM_DISPLAYS

Number of displays in the shelf.

DISPLAY_0_AT_MIDNIGHT

If set to true the display will show 0 at midnight and 12 otherwise.

DISPLAY_SWITCH_OFF_AT_0

If set to true the higher displays will turn off in case they would show 0.

USE_24_HOUR_FORMAT

If set to true 24 hour format will be used. For this one additional column is needed in the shelf to display it correctly.

NUM_SEGMENTS_PROGRESS

The number of segments to use for displaying a progress bar for the OTA updates.

LOADING_ANIMATION_DURATION

The time is shall take for one iteration of the loading animation.

BRIGHTNESS_INTERPOLATION

How fast the brightness interpolation shall react to brightness changes.

DISPLAY_FOR_SEPARATION_DOT

If set to -1 the flashing middle dot is disabled, otherwise this is the index of the Display segment that should display the dot.

ANIMATION_TARGET_FPS

Target Frames per second for the smoothness of animations.

ANIMATION_AFTERGLOW

Length of sooth animation transition from fully on to black and vice versa in percent NOTE: The higher this number the less obvious easing effects like bounce or elastic will be.

DOT_FLASH_SPEED

Length of the dot/s fading animation. One flash fades in and out.

DOT_FLASH_INTERVAL

Intervale in which the dot/s should flash.

NUM_SEPARATION_DOTS

Number of separation dots to use by default (or if no blynk functionality is available) allowed values are 1, 2 and 0 to turn it off.

ENABLE_LIGHT_SENSOR

Enable automatic brightness adjustments based on a light sensor.

LIGHT_SENSOR_PIN

ADC pin to which the light sensor is connected to.

LIGHT_SENSOR_AVERAGE

How many measurements shall be averaged. Higher number -> smoother but slower change.

LIGHT_SENSOR_MEDIAN_WIDTH

Width of the median calculation. Higher number -> smoother change Should never be higher than the LIGHT_SENSOR_AVERAGE.

LIGHT_SENSOR_READ_DELAY

Time that should pass before the light sensor is read again. Higher number -> slower adjustments but also changes will be more sudden.

LIGHT_SENSOR_MIN

AnalogRead value if the light sensor reads complete darkness.

LIGHT_SENSOR_MAX

AnalogRead value if the light sensor reads the brightest.

LIGHT_SENSOR_SENSITIVITY

Value between 0 and 255 that determines how much the light sensor values can influence the led brightness.

TIME_MANAGER_DEMO_MODE

enable for wifi less operation or to demo all the animations

DIGIT_ANIMATION_SPEED

The time it takes for one digit to morph into another.

FASTLED_SAFE_DELAY_MS

the minimum delay between calls of FastLED.show()

Enums

enum **DisplayIDs**

These enum definitions are used in the code do address the different Seven segment displays. The numbers have to match with the place of the display in the #DisplayManager::SegmentDisplayModes array in the file DisplayConfiguration.cpp.

Values:

enumerator **HIGHER_DIGIT_HOUR_DISPLAY**

enumerator **FIRST_INTERMEDIATE_DISPLAY**

enumerator **LOWER_DIGIT_HOUR_DISPLAY**

enumerator **SECOND_INTERMEDIATE_DISPLAY**

enumerator **HIGHER_DIGIT_MINUTE_DISPLAY**

enumerator **THIRD_INTERMEDIATE_DISPLAY**

enumerator **LOWER_DIGIT_MINUTE_DISPLAY**

enumerator **HIGHER_DIGIT_HOUR_DISPLAY**

enumerator **FIRST_INTERMEDIATE_DISPLAY**

enumerator **LOWER_DIGIT_HOUR_DISPLAY**

enumerator **SECOND_INTERMEDIATE_DISPLAY**

enumerator **HIGHER_DIGIT_MINUTE_DISPLAY**

enumerator **THIRD_INTERMEDIATE_DISPLAY**

enumerator **LOWER_DIGIT_MINUTE_DISPLAY**

enum **DisplayIDs**

These enum definitions are used in the code do address the different Seven segment displays. The numbers have to match with the place of the display in the #DisplayManager::SegmentDisplayModes array in the file DisplayConfiguration.cpp.

Values:

enumerator **HIGHER_DIGIT_HOUR_DISPLAY**

enumerator **FIRST_INTERMEDIATE_DISPLAY**

enumerator **LOWER_DIGIT_HOUR_DISPLAY**

enumerator **SECOND_INTERMEDIATE_DISPLAY**

enumerator **HIGHER_DIGIT_MINUTE_DISPLAY**

enumerator **THIRD_INTERMEDIATE_DISPLAY**

enumerator **LOWER_DIGIT_MINUTE_DISPLAY**

enumerator **HIGHER_DIGIT_HOUR_DISPLAY**

enumerator **FIRST_INTERMEDIATE_DISPLAY**

enumerator **LOWER_DIGIT_HOUR_DISPLAY**

enumerator **SECOND_INTERMEDIATE_DISPLAY**

enumerator **HIGHER_DIGIT_MINUTE_DISPLAY**

enumerator **THIRD_INTERMEDIATE_DISPLAY**

enumerator **LOWER_DIGIT_MINUTE_DISPLAY**

Display configuration

group **DisplayConfiguration**

Configuration to tell the system how the LEDs are wired together and arranged.

Variables

static *SevenSegment*::*SegmentPosition* **SegmentPositions**[NUM_SEGMENTS]

Each segment belongs to some display. This array defines the segment position within this one display. The order of these has to mach the order in which the LEDs are wired.

static *Segment*::*direction* **SegmentDirections**[NUM_SEGMENTS]

Each segment has a direction, this is important for animation. The order of them is the same as #Display-Manager::SegmentPositions and the direction has to match the sequence in which the LEDs are wired.

static *SevenSegment*::*SevenSegmentMode* **SegmentDisplayModes**[NUM_DISPLAYS] = {*SevenSegment*::*ONLY_ONE*, *SevenSegment*::*HALF_SEGMENT*, *SevenSegment*::*FULL_SEGMENT*, *SevenSegment*::*HALF_SEGMENT*, *SevenSegment*::*FULL_SEGMENT*, *SevenSegment*::*HALF_SEGMENT*, *SevenSegment*::*FULL_SEGMENT* }

Displays that are present. These define the displays in the order that is set in the #DisplayManager::diplayIndex array.

static uint8_t **diplayIndex**[NUM_SEGMENTS]

These indicies correspond to the index of a Diplay in the array above (#DisplayManager::SegmentDisplayModes). They define which segment belongs to which Display in the order that they are wired in. The enum *DisplayIDs* from Configuration.h can also be used to create a more readable config.

3.1.6 Available Animations

Only the default animations are listed here as this documentation only covers one configuration

Default

Warning: doxygenfile: Found multiple matches for file "Animations.h

3.1.7 Library API

Page Hierarchy

Class Hierarchy

Full API

Classes and Structs

Struct Animator::animationStep

- Defined in file_lib_LED_clock_Modules_Animator_inc_Animator.h

Nested Relationships

This struct is a nested type of *Class Animator*.

Struct Documentation

struct **animationStep**

Configuration structure used in the linked list to construct an animation chain.

Note: All three lists have to have the same length. The length also must be consistent across all animation steps. If the system crashes when calling an animation it is most likeley due to missing arrays or missmatched array lengths.

Param arrayIndex
index of the array position where the objects that shall be animated is located. Set to -1 to ignore

Param animationEffects
array of animation effects that shall be played back

Param easingEffects
array of easing effect ("modifiers") that shall be applied to the animation

Public Members

int16_t *arrayIndex

AnimatableObject::*AnimationFunction* *animationEffects

EasingBase **easingEffects

Struct Animator::ComplexAmination

- Defined in file_lib_LED_clock_Modules_Animator_inc_Animator.h

Nested Relationships

This struct is a nested type of *Class Animator*.

Struct Documentation

struct ComplexAmination

Configuration structure for a complex animation.

Note: All list elements must have the same array length

Param animationComplexity
Maximum of how many animations can be triggered at the same time

Param LengthPerAnimation
How long one of the animations in the chain should last for

Param animations
list of animation steps that shall be played in sequence

Public Members

uint8_t animationComplexity

uint16_t LengthPerAnimation

LinkedList<*animationStep**> *animations

Struct Animator::ComplexAnimationInstance

- Defined in file_lib_LED_clock_Modules_Animator_inc_Animator.h

Nested Relationships

This struct is a nested type of *Class Animator*.

Struct Documentation

struct **ComplexAnimationInstance**

Public Members

ComplexAmination ***animation**

bool **loop**

uint16_t **counter**

AnimatableObject ****objects**

bool **running**

Struct DisplayManager::SegmentInstanceError

- Defined in file_lib_LED_clock_Modules_DisplayManager_inc_DisplayManager.h

Nested Relationships

This struct is a nested type of *Class DisplayManager*.

Struct Documentation

struct **SegmentInstanceError**

Public Members

SegmentPositions_t **segmentPosition**

DisplayIDs **Display**

Struct TimeManager::TimeInfo

- Defined in file_lib_LED_clock_Modules_TimeManager_inc_TimeManager.h

Nested Relationships

This struct is a nested type of *Class TimeManager*.

Struct Documentation

struct **TimeInfo**

Saves a time in hours, minutes and seconds.

Public Members

uint8_t **hours**

uint8_t **minutes**

uint8_t **seconds**

Class AnimatableObject

- Defined in file_lib_LED_clock_Modules_Animator_inc_AnimatableObject.h

Inheritance Relationships

Derived Type

- `public Segment` (*Class Segment*)

Class Documentation

class **AnimatableObject**

Base class which every object that can be animated by the *Animator* should inherit from.

Subclassed by *Segment*

Public Types

typedef void **AnimationCallBack**(void)

Callback typedef to use when an animation starts/finishes.

typedef void (***AnimationFunction**)(CRGB *leds, uint16_t length, CRGB animationColor, uint16_t totalSteps, int32_t currentStep, bool invert)

Typedef for animation effect functions. These should be implemented explicitly for every object that inherits from *AnimatableObject*.

Public Functions

void **setAnimationDoneCallback**(*AnimationCallBack* *callback)

Set a callback to be executued once an animation has finished running.

Parameters
callback – function to call

void **setAnimationStartCallback**(*AnimationCallBack* *callback)

Set a callback to be executued once an animation is started.

Parameters
callback – function to call

Protected Functions

AnimatableObject()

AnimatableObject(uint16_t OverallDuration, uint16_t steps)

~AnimatableObject()

void **setAnimationDuration**(uint16_t duration)

Set the overall duration of any animation called on this object.

Parameters
duration – animation duration in ms

uint16_t **getAnimationDuration**()

Set the overall duration of any animation called on this object.

Parameters
duration – animation duration in ms

void **setAnimationFps**(uint16_t setAnimationFps)

Set the target Frames Per Second for any animation called on this object.

Note: This does not guarantee that the animation is actually running on that refresh rate. If it is set too fast the system will run it at the maximum possible framerate instead.

Parameters

setAnimationFps – how often the animation should be updated on the actual LEDs in Frames/Second

void **start**()

Enables an animation to run when #AnimatableObject::tick() gets called.

void **stop**()

Disables an animation to run when #AnimatableObject::tick() gets called but does not reset it's current state. Could be thought of as "pausing" it.

void **reset**()

Stops (if not already done) and resets the animation to its starting point.

void **handle**(uint32_t state = -1)

Gets called by the *Animator Animator::handle* method when the animation is finished.

Parameters

state – if not -1 any animations currently running are going to be set to an exact state

virtual void **setAnimationEffect**(*AnimatableObject*::*AnimationFunction* newEffect)

Set the animation effect to the current object.

Parameters

newEffect – effect to execute the next time an animation is started on this object

virtual void **setAnimationEasing**(*EasingBase* *easingEffect)

Set the animation easing effect to the current object.

Parameters

easingEffect – effect to apply to the animation as an additional "modifier"

Class AnimationEffects

- Defined in file_lib_LED_clock_Modules_SevenSegment_inc_AnimationEffects.h

Class Documentation

class **AnimationEffects**

Public Functions

~AnimationEffects()

Public Static Attributes

static *AnimatableObject*::*AnimationFunction* **AnimateOutToRight** = &OutToRight

static *AnimatableObject*::*AnimationFunction* **AnimateOutToBottom** = &OutToRight

static *AnimatableObject*::*AnimationFunction* **AnimateOutToLeft** = &OutToLeft

static *AnimatableObject*::*AnimationFunction* **AnimateOutToTop** = &OutToLeft

static *AnimatableObject*::*AnimationFunction* **AnimateInToRight** = &InToRight

static *AnimatableObject*::*AnimationFunction* **AnimateInToBottom** = &InToRight

static *AnimatableObject*::*AnimationFunction* **AnimateInToLeft** = &InToLeft

static *AnimatableObject*::*AnimationFunction* **AnimateInToTop** = &InToLeft

static *AnimatableObject*::*AnimationFunction* **AnimateInToMiddle** = &InToMiddle

static *AnimatableObject*::*AnimationFunction* **AnimateOutToMiddle** = &OutToMiddle

static *AnimatableObject*::*AnimationFunction* **AnimateOutFromMiddle** = &OutFromMiddle

static *AnimatableObject*::*AnimationFunction* **AnimateInFromMiddle** = &InFromMiddle

static *AnimatableObject*::*AnimationFunction* **AnimateMiddleDotFlash** = &MiddleDotFlash

Class Animator

- Defined in file_lib_LED_clock_Modules_Animator_inc_Animator.h

Nested Relationships

Nested Types

- *Struct Animator::ComplexAmination*
- *Struct Animator::ComplexAnimationInstance*
- *Struct Animator::animationStep*

Class Documentation

class **Animator**

The *Animator* class is responsible for handling all animations of objects that inherit from *AnimatableObject* In the system there can be more than one *Animator* running at the same time.

Public Functions

~Animator()

Destroy the *Animator* object.

void **add**(*AnimatableObject* *animationToAdd)

Add an animatable object to the *Animator*. The object is then updated by it.

Parameters
animationToAdd – Pointer to the object whose animations should be handled by this animator.

void **remove**(*AnimatableObject* *animationToRemove)

Remove an animatable object from the *Animator*. The object is then no longer updated by it.

Parameters
animationToRemove – Pointer to the object whose animations should not be handled anymore by this animator.

Pre
The object must have been added earlier by using *Animator::add*

void **handle**(uint32_t state = -1)

To be called periodically as fast as possible. Updates all animation states of all #AnimatableObjects assigned to this *Animator*.

Parameters
state – if not -1 any animations currently running are going to be set to an exact state

void **setAnimation**(*AnimatableObject* *object, *AnimatableObject*::*AnimationFunction* animationEffect, uint16_t duration, *EasingBase* *easing = nullptr, uint8_t fps = ANIMATION_TARGET_FPS)

Setup all parameters for an animation of an object assigned to this *Animator* but do not start it.

Parameters

- **object** – Object for which to change the animation for
- **animationEffect** – Animation effect that should be used next time an animation for this object is started.

- **duration** – Total duration of the animation effect once it is started.
- **easing** – [optional] default = *NO_EASING*; Easing effect to apply "on top" of the animation
- **fps** – [optional] default = *ANIMATION_TARGET_FPS*; Target FPS to run the animation at

void **startAnimation**(*AnimatableObject* *object, *AnimatableObject*::*AnimationFunction* animationEffect, uint16_t duration, *EasingBase* *easing = nullptr, uint8_t fps = ANIMATION_TARGET_FPS)

Setup all parameters for an animation of an object assigned to this *Animator* and start it right away.

Parameters

- **object** – Object for which to start the animation for
- **animationEffect** – Animation effect that should be started
- **duration** – Total duration of the animation effect
- **easing** – [optional] default = *NO_EASING*; Easing effect to apply "on top" of the animation
- **fps** – [optional] default = *ANIMATION_TARGET_FPS*; Target FPS to run the animation at

void **startAnimation**(*AnimatableObject* *object, *AnimatableObject*::*AnimationFunction* animationEffect, *EasingBase* *easing = nullptr)

Setup the most important parameters for an animation of an object assigned to this *Animator* and start it right away.

Parameters

- **object** – Object for which to start the animation for
- **animationEffect** – Animation effect that should be started
- **easing** – [optional] default = *NO_EASING*; Easing effect to apply "on top" of the animation

void **startAnimation**(*AnimatableObject* *object)

Starts an animation which was previously setup.

Parameters
object – Object for which to start the animation for

Pre
An animation must already be setup for this object. Either though a call of *Animator::setAnimation* or a previous *Animator::startAnimation* call.

void **setAnimationDuration**(*AnimatableObject* *object, uint16_t duration)

Set the animation duration of an object assigned to this *Animator*.

Parameters

- **object** – Object for which to set the suration for
- **duration** – Total animation duration in ms once it is started.

void **stopAnimation**(*AnimatableObject* *object)

Calls the *AnimatableObject::stop* function on the given object.

Parameters
object – Object for which to stop the animation for

void **resetAnimation**(*AnimatableObject* *object)

Calls the *AnimatableObject::reset* function on the given object.

Parameters
object – Object for which to reset the animation for

AnimatableObject::*AnimationFunction* **getAnimationEffect**(*AnimatableObject* *object)

Get the current animation effect of the object.

Parameters
object – Object for which to get the current animation for

Returns
AnimatableObject::AnimationFunction pointer to the current animation effect function

ComplexAnimationInstance ***PlayComplexAnimation**(*ComplexAmination* *animation, *AnimatableObject* *animationObjectsArray[], bool looping = false)

Starts a complex chain of animations.

Parameters

- **animation** – pointer to the animation that shall be played
- **animationObjectsArray** – Array of the objects that shall be animated. The indices for the array are defined in the animation itself
- **looping** – Whether the animation shall be looped or not

Returns
uint32_t The animation ID of the newly started animation -1 which results to the max 32 bit value represents an error while starting the animation

ComplexAnimationInstance ***BuildComplexAnimation**(*ComplexAmination* *animation, *AnimatableObject* *animationObjectsArray[], bool looping = false)

Builds a complex animation but does not start it.

Parameters

- **animation** – pointer to the animation that shall be played
- **animationObjectsArray** – Array of the objects that shall be animated. The indices for the array are defined in the animation itself
- **looping** – Whether the animation shall be looped or not

Returns
uint32_t The animation ID of the newly started animation -1 which results to the max 32 bit value represents an error while starting the animation

void **setComplexAnimationStep**(*ComplexAnimationInstance* *animationInst, uint8_t step, uint32_t state)

set a complex animation to a specific step and state

Parameters

- **animationInst** – animation to use, retrived by calling *Animator::BuildComplexAnimation*
- **step** – Step of the complex animation which shall be executed
- **state** – state of the current step

void **ComplexAnimationStopLooping**(*ComplexAnimationInstance* *animationInst)

disables looping of the complex animation so that it sops running after the current cycle is done running

Parameters
animationID – ID of the animation which shall be stopped

void **WaitForComplexAnimationCompletion**(*ComplexAnimationInstance* *animationInst)

Blocks exectution of further code until the currently running animation is complete.

Parameters
animationID – ID of the animation which shall be waited for

void **delay**(uint32_t delayInMs)

Delays further execution of code without blocking any currently ongoing animations.

Parameters
delayInMs – time to wait before moving on in ms

Public Static Functions

static *Animator* ***getInstance**()

get an instance of the *Animator* object

struct **animationStep**

Configuration structure used in the linked list to construct an animation chain.

Note: All three lists have to have the same length. The length also must be consistent across all animation steps. If the system crashes when calling an animation it is most likeley due to missing arrays or missmatched array lengths.

Param arrayIndex
index of the array position where the objects that shall be animated is located. Set to -1 to ignore

Param animationEffects
array of animation effects that shall be played back

Param easingEffects
array of easing effect ("modifiers") that shall be applied to the animation

Public Members

int16_t ***arrayIndex**

AnimatableObject::*AnimationFunction* ***animationEffects**

EasingBase ****easingEffects**

struct **ComplexAmination**

Configuration structure for a complex animation.

Note: All list elements must have the same array length

Param animationComplexity
Maximum of how many animations can be triggered at the same time

Param LengthPerAnimation
How long one of the animations in the chain should last for

Param animations
list of animation steps that shall be played in sequence

Public Members

uint8_t **animationComplexity**

uint16_t **LengthPerAnimation**

LinkedList<*animationStep**> ***animations**

struct **ComplexAnimationInstance**

Public Members

ComplexAmination ***animation**

bool **loop**

uint16_t **counter**

AnimatableObject ****objects**

bool **running**

Class BackEase

- Defined in file_lib_Easings_src_easetypes_BackEase.h

Inheritance Relationships

Base Type

- `public EasingBase` (*Class EasingBase*)

Class Documentation

class **BackEase** : public *EasingBase*

Public Functions

BackEase()

BackEase(*easingType_t* type_, *NUMBER* overshoot_)

virtual *NUMBER* **easeIn**(*NUMBER* time_) const

virtual *NUMBER* **easeOut**(*NUMBER* time_) const

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const

void **setOvershoot**(*NUMBER* overshoot_)

Class BlynkConfig

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Class Documentation

class **BlynkConfig**

Configuration class for storing all Blynk related information.

Public Types

enum **ColorSelector**

possible selection options of the segmented switch responsible for selecting which color should be changed in the App

Values:

enumerator **CHANGE_HOURS_COLOR**

enumerator **CHANGE_MINUTES_COLOR**

enumerator **CHANGE_INTERIOR_COLOR**

enumerator **CHANGE_DOT_COLOR**

Public Functions

~BlynkConfig()

Destroy the Blynk Config object. Ensure proper deletion.

void **setup**()

to be called as part of the setup function

call Blynk.config and start the separate thread on the second core

Pre
WIFI connection has to be established already by calling #wifiSetup before this method

void **stop**()

Stop the execution of Blynk.

Terminates the blynk task running on the second core.

void **updateUI**()

Call this if a UI update is needed Possible cases for this are that the timer ticked or was triggered or the alarm was triggered.

Notify the Blynk thread that a UI update is needed. What exactly needs to be updated will be figured out in the thread loop itself.

void **changeSelection**(*ColorSelector* selector, bool state)

Change the selection of the segmented switch responsible for selecting which color should be changed.

Figure out which changes have to be made in the UI to represent the correct state.

Public Members

uint8_t **ColorSelection**

CRGB **InternalColor**

CRGB **HourColor**

CRGB **MinuteColor**

CRGB **DotColor**

bool **blynkUIUpdateRequired**

DisplayManager ***ShelfDisplays**

bool **isClearAction**

Public Static Functions

static *BlynkConfig* ***getInstance**()

Get the Instance object. Use this to get an instance to the Blynk config singleton instead of its constructor.

Either instantiate a new *BlynkConfig* object by calling the private constructor and return it's address or it an instance of it already exists just return that.

Returns
BlynkConfig* pointer to the already existing or newly created *BlynkConfig* object.

Returns
BlynkConfig* address of the new/already existing blynk config object

Class BounceEase

- Defined in file_lib_Easings_src_easetypes_BounceEase.h

Inheritance Relationships

Base Type

- `public EasingBase` (*Class EasingBase*)

Class Documentation

class **BounceEase** : public *EasingBase*

Public Functions

virtual *NUMBER* **easeIn**(*NUMBER* time_) const

virtual *NUMBER* **easeOut**(*NUMBER* time_) const

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const

Class CircularEase

- Defined in file_lib_Easings_src_easetypes_CircularEase.h

Inheritance Relationships

Base Type

- `public EasingBase` (*Class EasingBase*)

Class Documentation

class **CircularEase** : public *EasingBase*

Public Functions

virtual *NUMBER* **easeIn**(*NUMBER* time_) const

virtual *NUMBER* **easeOut**(*NUMBER* time_) const

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const

Class ClockState

- Defined in file_lib_LED_clock_Modules_ClockState_inc_ClockState.h

Class Documentation

class **ClockState**

The clockState is responsible to hold all the data that needs to be communicated between components. It can be imagined as kind of like an "object oriented global variable".

Public Types

enum **ClockStates**

Avaliable clock modes each with a different behaviour.

Values:

enumerator **CLOCK_MODE**

enumerator **TIMER_MODE**

enumerator **TIMER_NOTIFICATION**

enumerator **ALARM_NOTIFICATION**

Public Functions

~ClockState()

Destroys the *ClockState* object and cause *ClockState::getInstance* to create a new object the next time it is called.

void **switchMode**(*ClockStates* newState)

Switch the current mode of the clock.

ClockStates **getMode**()

Returns the current mode of the clock.

void **handleStates**()

Has to be called periodically to update the screen and process state transitions within the state machine.

Public Members

uint8_t **clockBrightness**

Base brightness of the clock. The actual brightness can still change if a light sensor is used.

uint8_t **nightModeBrightness**

Brightness of the clock driing nighttime hours define by *ClockState::NightModeStartTime* and *ClockState::NightModeStopTime*.

TimeManager::*TimeInfo* **NightModeStartTime**

Any time after will be considered nighttime as long as it is still lower than *ClockState::NightModeStopTime*.

TimeManager::*TimeInfo* **NightModeStopTime**

Any time before will be considered nighttime as long as it is still higher than *ClockState::NightModeStartTime*.

uint8_t **numDots**

defines the number of dots. 0 -> no dot; 1 -> one dot; 2 -> two dots; other-> one dot

Public Static Functions

static *ClockState* ***getInstance**()

Get the instance of the Clock object or create it if it was not yet instantiated.

Returns

ClockState* returns the address to the *ClockState* object

Class CubicEase

- Defined in file_lib_Easings_src_easetypes_CubicEase.h

Inheritance Relationships

Base Type

- `public EasingBase` (*Class EasingBase*)

Class Documentation

class **CubicEase** : public *EasingBase*

Public Functions

virtual *NUMBER* **easeIn**(*NUMBER* time_) const

virtual *NUMBER* **easeOut**(*NUMBER* time_) const

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const

Class DisplayManager

- Defined in file_lib_LED_clock_Modules_DisplayManager_inc_DisplayManager.h

Nested Relationships

Nested Types

- *Struct DisplayManager::SegmentInstanceError*

Class Documentation

class **DisplayManager**

The display manager is responsible to Manage all displays. It holds an instance to every display avaliable on the clock and manages the updating of all displays together.

Public Functions

~DisplayManager()

Destroys the Display Manager object and cause *DisplayManager::getInstance* to create a new object the next time it is called.

void **InitSegments**(uint16_t indexOfFirstLed, uint8_t ledsPerSegment, CRGB initialColor, uint8_t initBrightness = 128)

Initialize all the segment using the configuration from DisplayConfiguration.cpp.

Parameters

- **indexOfFirstLed** – Index of the first led in the string that is part of a segment (usually 0)
- **ledsPerSegment** – Sets the number of LEDs that are in one segment. this will be the same for all segments
- **initialColor** – Sets the initial color of all the segments. This does not switch any segments on by it's own
- **initBrightness** – Sets the initial brightness of all the segments to avoid brigness jumps during startup

void **setAllSegmentColors**(CRGB color)

Sets the color of all segments and updates it immediately for all segments that are currently switched on.

Parameters
color – Color to set the LEDs to

void **setHourSegmentColors**(CRGB color)

Sets the color of the the segments which are displaying hours and updates it immediately for all segments that are currently switched on.

Parameters
color – Color to set the LEDs to

void **setMinuteSegmentColors**(CRGB color)

Sets the color of the the segments which are displaying minutes and updates it immediately for all segments that are currently switched on.

Parameters
color – Color to set the LEDs to

void **displayRaw**(uint8_t Hour, uint8_t Minute)

Displays the numbers given as they are on the respective displays.

Parameters

- **Hour** – Number to show on the hours display
- **Minute** – Number to show on the minutes display

void **displayTime**(uint8_t hours, uint8_t minutes)

Display the time, Automatically convert between 24h and 12h formats.

Parameters

- **hours** – Hours in a range of 0 to 24
- **minutes** – Hours in a range of 0 to 59

void **displayTimer**(uint8_t hours, uint8_t minutes, uint8_t seconds)

Display the remaining time on the timer. Always displays the highest possible output. For example: If hour is anything else than 1 minutes will be displayed in the hour spot and seconds on the minute spot on the display if possible.

Parameters

- **hours** – Hours in a range of 0 to 24
- **minutes** – Hours in a range of 0 to 59
- **seconds** – Seconds in a range of 0 to 59

void **handle**()

Has to be called cyclicly in the loop to enable live updating of the LEDs.

void **setInternalLEDColor**(CRGB color)

Sets the color of the interrior LEDs and displays it immediately.

void **setDotLEDColor**(CRGB color)

Sets the color of the seperation dot LEDs and displays it immediately.

void **showLoadingAnimation**()

Starts the loading animation.

void **stopLoadingAnimation**()

Stops the currently running animation after it is finished. This causes a looping animation to stop after its current cycle.

void **waitForLoadingAnimationFinish**()

Wait until the currently set complex animation is finished.

void **turnAllSegmentsOff**()

Turns all displays off completely, Does not affect interior lights.

void **turnAllLEDsOff**()

Turn off all LEDs including internal LEDs.

void **displayProgress**(uint32_t total)

start displaying a progress bar on the LEDs

Parameters

total – How much progress there is to do in total

void **updateProgress**(uint32_t progress)

Update the progress bar.

Parameters

progress – How much progress was done already

Pre

DisplayManager::displayProgress has to be called once before updating the progress with this function

void **delay**(uint32_t timeInMs)

Use this delay instead of the Arduino delay to enable Display updates during the delay.

Parameters

timeInMs – Delay time in ms

void **setGlobalBrightness**(uint8_t brightness, bool enableSmoothTransition = true)

Sets the Brightness globally for all leds.

Parameters

- **brightness** – value between 0 for lowest, and 255 for the highes brightness
- **enableSmoothTransition** – If true the LEDs will transition to the new value smoothly

void **flashSeparationDot**(uint8_t numDots)

Calling the Flash dot animation for the appropriate segments in the middle of the clock face.

void **test**()

Used for testing purposes.

Public Static Functions

static *DisplayManager* ***getInstance**()

Get the instance of the *DisplayManager* object or create it if it was not yet instantiated.

Returns

DisplayManager* returns the address to the *DisplayManager* object

static int16_t **getGlobalSegmentIndex**(*SegmentPositions_t* segmentPosition, *DisplayIDs* Display)

get the index of a segment in regards to it's position on the clock face. This makes writing animations a lot easier as it will act as an abstraction layer between the animation config and the display config

Parameters

- **segmentPosition** – Position of a segment in the seven segment display
- **Display** – Which display should be targeted

Returns

int16_t index of the *Segment* in the #DisplayManager::SegmentPositions array

static void **printAnimationInitErrors**()

Debugging function which will print out any errors that occurred when using the *DisplayManager::getGlobalSegmentIndex* function. Especially useful for debugging weirdly behaving animations as animations will be configured before the Serial connection is up. If an error was detected by the *DisplayManager::getGlobalSegmentIndex* function a message will be added to a buffer which can then be printed out using this function.

Class EasingBase

- Defined in file_lib_Easings_src_EasingBase.h

Inheritance Relationships

Derived Types

- public BackEase (*Class BackEase*)
- public BounceEase (*Class BounceEase*)
- public CircularEase (*Class CircularEase*)
- public CubicEase (*Class CubicEase*)
- public ElasticEase (*Class ElasticEase*)
- public ExponentialEase (*Class ExponentialEase*)
- public LinearEase (*Class LinearEase*)
- public QuadraticEase (*Class QuadraticEase*)
- public QuarticEase (*Class QuarticEase*)
- public QuinticEase (*Class QuinticEase*)
- public SineEase (*Class SineEase*)

Class Documentation

class **EasingBase**

Subclassed by *BackEase*, *BounceEase*, *CircularEase*, *CubicEase*, *ElasticEase*, *ExponentialEase*, *LinearEase*, *QuadraticEase*, *QuarticEase*, *QuinticEase*, *SineEase*

Public Functions

EasingBase()

EasingBase(*easingType_t* type_)

virtual **~EasingBase**()

void **setType**(*easingType_t* type_)

NUMBER **ease**(*NUMBER* time_) const

virtual *NUMBER* **easeIn**(*NUMBER* time_) const = 0

virtual *NUMBER* **easeOut**(*NUMBER* time_) const = 0

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const = 0

void **setDuration**(*NUMBER* duration_)

void **setTotalChangeInPosition**(*NUMBER* totalChangeInPosition_)

Protected Attributes

NUMBER **_change**

NUMBER **_duration**

easingType_t **_type**

Class ElasticEase

- Defined in file_lib_Easings_src_easetypes_ElasticEase.h

Inheritance Relationships

Base Type

- `public EasingBase` (*Class EasingBase*)

Class Documentation

class **ElasticEase** : public *EasingBase*

Public Functions

ElasticEase()

ElasticEase(*easingType_t* type_, *NUMBER* period_, *NUMBER* amplitude_)

virtual *NUMBER* **easeIn**(*NUMBER* time_) const

virtual *NUMBER* **easeOut**(*NUMBER* time_) const

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const

void **setPeriod**(*NUMBER* period_)

void **setAmplitude**(*NUMBER* amplitude_)

Class ExponentialEase

- Defined in file_lib_Easings_src_easetypes_ExponentialEase.h

Inheritance Relationships

Base Type

- `public EasingBase` (*Class EasingBase*)

Class Documentation

class **ExponentialEase** : public *EasingBase*

Public Functions

virtual *NUMBER* **easeIn**(*NUMBER* time_) const

virtual *NUMBER* **easeOut**(*NUMBER* time_) const

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const

Class LinearEase

- Defined in file_lib_Easings_src_easetypes_LinearEase.h

Inheritance Relationships

Base Type

- `public EasingBase` (*Class EasingBase*)

Class Documentation

class **LinearEase** : public *EasingBase*

Public Functions

virtual *NUMBER* **easeIn**(*NUMBER* time_) const

virtual *NUMBER* **easeOut**(*NUMBER* time_) const

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const

Class QuadraticEase

- Defined in file_lib_Easings_src_easetypes_QuadraticEase.h

Inheritance Relationships

Base Type

- `public EasingBase` (*Class EasingBase*)

Class Documentation

class **QuadraticEase** : public *EasingBase*

Public Functions

virtual *NUMBER* **easeIn**(*NUMBER* time_) const

virtual *NUMBER* **easeOut**(*NUMBER* time_) const

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const

Class QuarticEase

- Defined in file_lib_Easings_src_easetypes_QuarticEase.h

Inheritance Relationships

Base Type

- `public EasingBase` (*Class EasingBase*)

Class Documentation

class **QuarticEase** : public *EasingBase*

Public Functions

virtual *NUMBER* **easeIn**(*NUMBER* time_) const

virtual *NUMBER* **easeOut**(*NUMBER* time_) const

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const

Class QuinticEase

- Defined in file_lib_Easings_src_easetypes_QuinticEase.h

Inheritance Relationships

Base Type

- `public EasingBase` (*Class EasingBase*)

Class Documentation

class **QuinticEase** : public *EasingBase*

Public Functions

virtual *NUMBER* **easeIn**(*NUMBER* time_) const

virtual *NUMBER* **easeOut**(*NUMBER* time_) const

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const

Class Segment

- Defined in file_lib_LED_clock_Modules_SevenSegment_inc_Segment.h

Inheritance Relationships

Base Type

- `public AnimatableObject` (*Class AnimatableObject*)

Class Documentation

class **Segment** : public *AnimatableObject*

Single segment class definition. used to store which LEDs belong to one segment and applying animations on it.

Public Types

enum **direction**

Direction in which the LED strip is wired in.

Values:

enumerator **LEFT_TO_RIGHT**

enumerator **RIGHT_TO_LEFT**

enumerator **TOP_TO_BOTTTOM**

enumerator **BOTTOM_TO_TOP**

Public Functions

Segment(CRGB LEDBuffer[], uint16_t indexOfFirstLEDInSegment, uint8_t segmentLength, *direction* Direction, CRGB segmentColor = CRGB::Black)

Construct a new *Segment* object.

Parameters

- **LEDBuffer** – Array of all LEDs connected in one string. Substitution of this buffer is done internally
- **indexOfFirstLEDInSegment** – Index of the first LED in the whole LED string that belongs to this segment
- **segmentLength** – Number of LEDs which belong to this segment
- **Direction** – Defines which way the LED segment is wired in
- **segmentColor** – initial color of the segment

~Segment()

Destroy the *Segment* object.

virtual void **tick**(int32_t currentState)

Set the current animation state of the segment to a defined value.

Parameters

currentState – The current state of the animation starting at 0, relative to it's absolute animation length. Over and undershoot is also possible.

void **setColor**(CRGB SegmentColor)

sets the color of the segment without displaying the change

void **updateColor**(CRGB SegmentColor)

sets the color of the segment and updates it automatically in case the segment is turned on

void **display**()

Write the current animation color to all LEDs that belong to this segment. Writes to the LED buffer but relies on an external FastLED.show() call to actually write that change to the LEDs.

void **off**()

Turns off the leds of this segment but doesn't change the stored color of the segment.

void **updateAnimationColor**(CRGB newColor)

Change tha animation color, also possible to do while an animation is in progress.

Parameters

newColor – new color to use in for the current segment and animation

Class SevenSegment

- Defined in file_lib_LED_clock_Modules_SevenSegment_inc_SevenSegment.h

Class Documentation

class **SevenSegment**

Class definition for *SevenSegment* which groups together all *Segment* objects which belong to together and provides some wrapper functions to manage all seven segments together.

Public Types

enum **SegmentPosition**

Segment positions for addressing the internal segment mapping table - Please use *SegmentPositions_t* for all external uses instead.

Values:

enumerator **LeftTopSegment**

enumerator **MiddleTopSegment**

enumerator **RightTopSegment**

enumerator **CenterSegment**

enumerator **LeftBottomSegment**

enumerator **MiddleBottomSegment**

enumerator **RightBottomSegment**

enum **SevenSegmentMode**

The mode of the seven segment display. This also defines which segments ave to be linked and which ones can be left out, also which digits can be displayed by this paricular instance of the seven segment display.

Values:

enumerator **FULL_SEGMENT**

enumerator **HALF_SEGMENT**

has 7 segments

enumerator **ONLY_ONE**

has only the 3 horizontal segments has only the right 2 vertical segments

Public Functions

SevenSegment(*SevenSegmentMode* mode, *Animator* *DisplayAnimationHandler)

Construct a new Seven *Segment* object.

Parameters

- **mode** – defines which digits can be displayed and which segments will not be assigned
- **DisplayAnimationHandler** – Animation handler which will be resposible for refreshing this display.

~SevenSegment()

Destroy the Seven *Segment* object.

void **add**(*Segment* *segmentToAdd, *SegmentPosition* positionInDisplay)

Add a single segment to the Seven segment display.

Parameters

- **segmentToAdd** – *Segment* which shall be added
- **positionInDisplay** – Position of the added segment withing the seven segment display

void **DisplayNumber**(uint8_t value)

Display a number on the display. If the display is not able to display the passed number nothing will happen.

Parameters

value – Number to display 0 - 9

void **DisplayChar**(char value)

Display a character on a Display. NOTE: Currently not implemented (only supports 0-9 as characters)

Parameters

value – Number to display '0' - '9'

void **FlashMiddleDot**(uint8_t numDots)

Plays a fading in and out animation on the middle (or two middle, if segment length is even) LEDs of one or multiple segment/s The function will internally display the animation on the right segments depending on the *SevenSegment::SevenSegmentMode*.

Parameters

numDots – Number of dots to display 0-2

bool **canDisplay**(char charToCheck)

checks if a particular character can be displayed on this display.

Parameters
charToCheck – Char which shall be checked for

Returns
true if the char can be displayed

Returns
false if the char cannot be displayed

void **setColor**(CRGB color)

Sets the animation color which will be displayed on the LEDs the next time an animation ticks.

Parameters
color – Color to set

void **updateColor**(CRGB color)

Sets the current and also the animation color which will be displayed on the LEDs the next time the LEDs are updated.

Parameters
color – Color to set

void **off**()

Turn all LEDs in this seven segment display off. will be pushed to the LEDs with the next call of FastLED.show()

Class SineEase

- Defined in file_lib_Easings_src_easetypes_SineEase.h

Inheritance Relationships

Base Type

- `public EasingBase` (*Class EasingBase*)

Class Documentation

class **SineEase** : public *EasingBase*

Public Functions

virtual *NUMBER* **easeIn**(*NUMBER* time_) const

virtual *NUMBER* **easeOut**(*NUMBER* time_) const

virtual *NUMBER* **easeInOut**(*NUMBER* time_) const

Class TimeManager

- Defined in file_lib_LED_clock_Modules_TimeManager_inc_TimeManager.h

Nested Relationships

Nested Types

- *Struct TimeManager::TimeInfo*

Class Documentation

class **TimeManager**

The *TimeManager* is responsible for synchronizing the time to the NTP servers and keeping track of it offline if the WIFI connection was lost. Also manages alarms and timers.

Public Types

enum **Weekdays**

Definition of weekdays ids.

Values:

enumerator **NONE**

enumerator **MONDAY**

enumerator **TUESDAY**

enumerator **WEDNESDAY**

enumerator **THURSDAY**

enumerator **FRIDAY**

enumerator **SATURDAY**

enumerator **SUNDAY**

typedef void (*__TimerCallBack__)(void)

Timer callback function type which is called if a timer ticks or is elapsed or an alrm is triggered.

Public Functions

~TimeManager()

Destroy the Time Manager object.

bool **init**()

Initialize the time manager and synchronize to ntp for the first time.

Pre

prerequisite is that WIFI is already up and running

Returns

true if init was successful

void **handle**()

Handle any Time manager tasks that need to be handled outside of interrupts.

void **disableTimer**()

Disable the timer and deactivate the interrupt.

bool **synchronize**()

Synchronize the time with the NTP server.

TimeInfo **getCurrentTime**()

get the current time in a struct for displaying it on the clock

TimeInfo **getRemainingTimerTime**()

get the remaining time of the active timer

String **getCurrentTimeString**()

get the current time as a string

void **setTimerDuration**(*TimeInfo* newTimerDuration)

Set the duration for the Timer.

void **startTimer**()

Start the timer.

void **stopTimer**()

Stop the timer.

bool **isInBetween**(*TimeInfo* timeStart, *TimeInfo* timeStop)

Check if the current time is in a given time period.

Parameters

- **timeStart** – Start of the time period
- **timeStop** – End of the time period

Returns

true is returned if the current time is in between the two specified times

Returns
false is returned if the current time is not in between the two specified times

TimeInfo **addSeconds**(*TimeInfo* time, uint16_t secondsToAdd)

Adds the specified amount of seconds to a given time.

Parameters

- **time** – Base time element
- **secondsToAdd** – Seconds that should be added to the base time

Returns
TimeInfo Time struct with the new time where the seconds were already added

void **setTimerTickCallback**(*TimerCallBack* callback)

Set the Timer Tick Callback function.

Parameters
callback – Function which shall be called every time a timer ticks (once every second)

void **setTimerDoneCallback**(*TimerCallBack* callback)

Set the Timer Done Callback function.

Parameters
callback – Function which shall be called once a timer has fired

void **setAlarmTime**(*TimeInfo* alarmTime, *Weekdays* activeDays)

Set the time at which an alarm shall be triggered.

Parameters

- **alarmTime** – Time of the alarm
- **activeDays** – Weekdays on which the alarm shall be triggered

void **setAlarmMode**(bool active)

Set if the alarm is active or not.

Parameters
active – set to true if the alarm is supposed to be active, set to false to deactivate

void **setAlarmCallback**(*TimerCallBack* callback)

Set the Alarm Callback function which is called once an alarm fired.

Parameters
callback – Function to call if the alarm was triggered

bool **isAlarmActive**()

check if the alarm is active

Returns
true if the alarm is active

Returns
false if the alarm is deactivated

void **clearAlarm**()

Clear a currently triggered alarm without disabling it.

Public Static Functions

static *TimeManager* *getInstance()

Get the singelton instance ot the Time Manager.

struct TimeInfo

Saves a time in hours, minutes and seconds.

Public Members

uint8_t hours

uint8_t minutes

uint8_t seconds

Enums

Enum DisplayIDs

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Enum Documentation

enum DisplayIDs

These enum definitions are used in the code do address the different Seven segment displays. The numbers have to match with the place of the display in the #DisplayManager::SegmentDisplayModes array in the file DisplayConfiguration.cpp.

Values:

enumerator HIGHER_DIGIT_HOUR_DISPLAY

enumerator FIRST_INTERMEDIATE_DISPLAY

enumerator LOWER_DIGIT_HOUR_DISPLAY

enumerator SECOND_INTERMEDIATE_DISPLAY

enumerator HIGHER_DIGIT_MINUTE_DISPLAY

enumerator THIRD_INTERMEDIATE_DISPLAY

enumerator **LOWER_DIGIT_MINUTE_DISPLAY**

enumerator **HIGHER_DIGIT_HOUR_DISPLAY**

enumerator **FIRST_INTERMEDIATE_DISPLAY**

enumerator **LOWER_DIGIT_HOUR_DISPLAY**

enumerator **SECOND_INTERMEDIATE_DISPLAY**

enumerator **HIGHER_DIGIT_MINUTE_DISPLAY**

enumerator **THIRD_INTERMEDIATE_DISPLAY**

enumerator **LOWER_DIGIT_MINUTE_DISPLAY**

Enum easingType_t

- Defined in file_lib_Easings_src_EasingConstants.h

Enum Documentation

enum **easingType_t**

Values:

enumerator **EASE_IN**

enumerator **EASE_OUT**

enumerator **EASE_IN_OUT**

Enum SegmentPositions_t

- Defined in file_lib_LED_clock_Modules_SevenSegment_inc_SevenSegment.h

Enum Documentation

enum **SegmentPositions_t**

Enum for addressing certain Segments in the display for animation them.

Values:

enumerator **TOP_LEFT_SEGMENT**

enumerator **TOP_MIDDLE_SEGMENT**

enumerator **TOP_RIGHT_SEGMENT**

enumerator **CENTER_SEGMENT**

enumerator **BOTTOM_LEFT_SEGMENT**

enumerator **BOTTOM_MIDDLE_SEGMENT**

enumerator **BOTTOM_RIGHT_SEGMENT**

Functions

Function BLYNK_CONNECTED

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_CONNECTED” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V0)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V1)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V14)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V15)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V16)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V17)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V2)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V10)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V11)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V12)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V13)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V3)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V4)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V7)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V8)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V9)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V5)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BLYNK_WRITE(V6)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “BLYNK_WRITE” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function BlynkLoopCode(void *)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Function Documentation

void **BlynkLoopCode**(void *pvParameters)

Loop function which is executed on the second core of the ESP.

Loop function which is executed on the second core of the ESP.

Function BlynkLoopCode(void *)

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Function Documentation

void **BlynkLoopCode**(void *pvParameters)

Loop function which is executed on the second core of the ESP.

Loop function which is executed on the second core of the ESP.

Function InitAnimate0to1

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function "InitAnimate0to1" in doxygen xml output for project "LED-Clock" from directory: ./doxygen/xml

Function InitAnimate0to5

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function "InitAnimate0to5" in doxygen xml output for project "LED-Clock" from directory: ./doxygen/xml

Function InitAnimate0to9

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function "InitAnimate0to9" in doxygen xml output for project "LED-Clock" from directory: ./doxygen/xml

Function InitAnimate1to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function "InitAnimate1to0" in doxygen xml output for project "LED-Clock" from directory: ./doxygen/xml

Function InitAnimate1to2

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate1to2” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate1toOFF

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate1toOFF” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate2to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate2to0” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate2to1

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate2to1” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate2to3

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate2to3” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate3to2

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate3to2” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate3to4

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate3to4” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate4to3

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate4to3” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate4to5

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate4to5” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate5to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate5to0” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate5to4

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate5to4” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate5to6

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate5to6” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate6to5

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate6to5” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate6to7

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate6to7” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate7to6

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate7to6” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate7to8

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate7to8” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate8to7

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate8to7” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate8to9

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate8to9” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate9to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate9to0” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimate9to8

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimate9to8” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitAnimateOFFto1

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitAnimateOFFto1” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitIndefiniteLoadingAnimation(uint16_t)

- Defined in file_lib_LED_clock_Config_Animations_12h-extended_Animations.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitIndefiniteLoadingAnimation” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitIndefiniteLoadingAnimation(uint16_t)

- Defined in file_lib_LED_clock_Config_Animations_24h-clock_Animations.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitIndefiniteLoadingAnimation” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitLoadingProgressAnimation(uint16_t)

- Defined in file_lib_LED_clock_Config_Animations_12h-extended_Animations.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitLoadingProgressAnimation” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function InitLoadingProgressAnimation(uint16_t)

- Defined in file_lib_LED_clock_Config_Animations_24h-clock_Animations.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “InitLoadingProgressAnimation” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Function onTimer

- Defined in file_lib_LED_clock_Modules_TimeManager_src_TimeManager.cpp

Function Documentation

friend void *TimeManager*::**onTimer**()

Function SortFunction_SmallerThan

- Defined in file_lib_LED_clock_Modules_DisplayManager_src_DisplayManager.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “SortFunction_SmallerThan” in doxygen xml output for project “LED-Clock” from directory: ./doxygen/xml

Variables

Variable Animate0to1

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate0to1**

Variable Animate0to1

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate0to1**

Variable Animate0to5

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate0to5**

Variable Animate0to5

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate0to5**

Variable Animate0to9

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate0to9**

Variable Animate0to9

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate0to9**

Variable Animate1to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate1to0**

Variable Animate1to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate1to0**

Variable Animate1to2

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate1to2**

Variable Animate1to2

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate1to2**

Variable Animate1toOFF

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate1toOFF**

Variable Animate1toOFF

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate1toOFF**

Variable Animate2to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate2to0**

Variable Animate2to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate2to0**

Variable Animate2to1

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate2to1**

Variable Animate2to1

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate2to1**

Variable Animate2to3

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate2to3**

Variable Animate2to3

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate2to3**

Variable Animate3to2

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate3to2**

Variable Animate3to2

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate3to2**

Variable Animate3to4

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate3to4**

Variable Animate3to4

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate3to4**

Variable Animate4to3

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate4to3**

Variable Animate4to3

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate4to3**

Variable Animate4to5

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate4to5**

Variable Animate4to5

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate4to5**

Variable Animate5to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate5to0**

Variable Animate5to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate5to0**

Variable Animate5to4

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate5to4**

Variable Animate5to4

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate5to4**

Variable Animate5to6

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate5to6**

Variable Animate5to6

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate5to6**

Variable Animate6to5

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate6to5**

Variable Animate6to5

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate6to5**

Variable Animate6to7

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate6to7**

Variable Animate6to7

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate6to7**

Variable Animate7to6

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate7to6**

Variable Animate7to6

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate7to6**

Variable Animate7to8

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate7to8**

Variable Animate7to8

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate7to8**

Variable Animate8to7

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate8to7**

Variable Animate8to7

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate8to7**

Variable Animate8to9

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate8to9**

Variable Animate8to9

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate8to9**

Variable Animate9to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate9to0**

Variable Animate9to0

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate9to0**

Variable Animate9to8

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***Animate9to8**

Variable Animate9to8

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***Animate9to8**

Variable AnimateOFFto1

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***AnimateOFFto1**

Variable AnimateOFFto1

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***AnimateOFFto1**

Variable BlynkC

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Variable Documentation

Warning: doxygenvariable: Cannot find variable "BlynkC" in doxygen xml output for project "LED-Clock" from directory: ./doxygen/xml

Variable bounceEaseOut

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

BounceEase ***bounceEaseOut** = new *BounceEase*(*EASE_OUT*)

Variable ClockS

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Variable Documentation

Warning: doxygenvariable: Cannot find variable "ClockS" in doxygen xml output for project "LED-Clock" from directory: ./doxygen/xml

Variable cubicEaseIn

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

CubicEase ***cubicEaseIn** = new *CubicEase*(*EASE_IN*)

Variable cubicEaseInOut

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

CubicEase ***cubicEaseInOut** = new *CubicEase*(*EASE_IN_OUT*)

Variable cubicEaseOut

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

CubicEase ***cubicEaseOut** = new *CubicEase*(*EASE_OUT*)

Variable IndefiniteLoadingAnimation

- Defined in file_lib_LED_clock_Config_Animations_12h-extended_Animations.cpp

Variable Documentation

Animator::*ComplexAmination* ***IndefiniteLoadingAnimation**

Animation to be used during some indefinite loading operation. Loops along the outside of all segments in a "circle". Suppposed to be used as a endlessly looping animation and to be stopped by calling the stopLooping method as soon as loading is finished.

Animation to be used during some indefinite loading operation. Loops along the outside of all segments in a "circle". Suppposed to be used as a endlessly looping animation and to be stopped by calling the stopLooping method as soon as loading is finished.

Variable IndefiniteLoadingAnimation

- Defined in file_lib_LED_clock_Config_Animations_24h-clock_Animations.cpp

Variable Documentation

Animator::*ComplexAmination* ***IndefiniteLoadingAnimation**

Animation to be used during some indefinite loading operation. Loops along the outside of all segments in a "circle". Suppposed to be used as a endlessly looping animation and to be stopped by calling the stopLooping method as soon as loading is finished.

Animation to be used during some indefinite loading operation. Loops along the outside of all segments in a "circle". Suppposed to be used as a endlessly looping animation and to be stopped by calling the stopLooping method as soon as loading is finished.

Variable IndefiniteLoadingAnimation

- Defined in file_lib_LED_clock_Config_Animations_12h-extended_Animations.h

Variable Documentation

Animator::*ComplexAmination* *`IndefiniteLoadingAnimation`

Animation to be used during some indefinite loading operation. Loops along the outside of all segments in a "circle". Suppposed to be used as a endlessly looping animation and to be stopped by calling the stopLooping method as soon as loading is finished.

Animation to be used during some indefinite loading operation. Loops along the outside of all segments in a "circle". Suppposed to be used as a endlessly looping animation and to be stopped by calling the stopLooping method as soon as loading is finished.

Variable IndefiniteLoadingAnimation

- Defined in file_lib_LED_clock_Config_Animations_24h-clock_Animations.h

Variable Documentation

Animator::*ComplexAmination* *`IndefiniteLoadingAnimation`

Animation to be used during some indefinite loading operation. Loops along the outside of all segments in a "circle". Suppposed to be used as a endlessly looping animation and to be stopped by calling the stopLooping method as soon as loading is finished.

Animation to be used during some indefinite loading operation. Loops along the outside of all segments in a "circle". Suppposed to be used as a endlessly looping animation and to be stopped by calling the stopLooping method as soon as loading is finished.

Variable LoadingProgressAnimation

- Defined in file_lib_LED_clock_Config_Animations_12h-extended_Animations.cpp

Variable Documentation

Animator::*ComplexAmination* *`LoadingProgressAnimation`

Animation which is used to display a progress with a defined end point. Similar to a progress bar.

Variable LoadingProgressAnimation

- Defined in file_lib_LED_clock_Config_Animations_24h-clock_Animations.cpp

Variable Documentation

Animator::*ComplexAmination* ***LoadingProgressAnimation**

Animation which is used to display a progress with a defined end point. Similar to a progress bar.

Variable LoadingProgressAnimation

- Defined in file_lib_LED_clock_Config_Animations_12h-extended_Animations.h

Variable Documentation

Animator::*ComplexAmination* ***LoadingProgressAnimation**

Animation which is used to display a progress with a defined end point. Similar to a progress bar.

Variable LoadingProgressAnimation

- Defined in file_lib_LED_clock_Config_Animations_24h-clock_Animations.h

Variable Documentation

Animator::*ComplexAmination* ***LoadingProgressAnimation**

Animation which is used to display a progress with a defined end point. Similar to a progress bar.

Variable TimeM

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.cpp

Variable Documentation

Warning: doxygenvariable: Cannot find variable "TimeM" in doxygen xml output for project "LED-Clock" from directory: ./doxygen/xml

Variable TransformationLookupTable

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Variable Documentation

Animator::*ComplexAmination* ***TransformationLookupTable**[11][11]

Lookup table to know which animation to call for which transition.

Lookup table to know which animation to call for which transition.

Variable TransformationLookupTable

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.h

Variable Documentation

Animator::*ComplexAmination* ***TransformationLookupTable**[11][11]

Lookup table to know which animation to call for which transition.

Lookup table to know which animation to call for which transition.

Defines

Define __35B04FAD_DF21_40e9_8652_8E61F19D3912

- Defined in file_lib_Easings_src_easing.h

Define Documentation

__35B04FAD_DF21_40e9_8652_8E61F19D3912

Define __C7168DD6_B2B7_4753_833B_914C84EF332E

- Defined in file_lib_Easings_src_easetypes_QuarticEase.h

Define Documentation

__C7168DD6_B2B7_4753_833B_914C84EF332E

Define ADDITIONAL_LEDS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

ADDITIONAL_LEDS

Number of LEDs For interior lights.

Define ALARM_NOTIFICATION_PERIOD

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

ALARM_NOTIFICATION_PERIOD

For how long the Display should flash when an alarm was fired in seconds.

Define ANIMATION_AFTERGLOW

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

ANIMATION_AFTERGLOW

Length of sooth animation transition from fully on to black and vice versa in percent NOTE: The higher this number the less obvious easing effects like bounce or elastic will be.

Define ANIMATION_TARGET_FPS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

ANIMATION_TARGET_FPS

Target Frames per second for the smoothness of animations.

Define APPEND_DOWN_LIGHTERS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

APPEND_DOWN_LIGHTERS

If you wired the down lighter LEDs to the end of the LED strips set this to true.

Define BLYNK_AUTH_TOKEN

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

BLYNK_AUTH_TOKEN

If you want Blynk functionality paste your authentication token here.

Define BLYNK_CHANNEL_ALARM_START_BUTTON

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_ALARM_START_BUTTON

Define BLYNK_CHANNEL_ALARM_TIME_INPUT

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_ALARM_TIME_INPUT

Define BLYNK_CHANNEL_BRIGHTNESS_SLIDER

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_BRIGHTNESS_SLIDER

Define BLYNK_CHANNEL_CURRENT_COLOR_PICKER

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_CURRENT_COLOR_PICKER

Define BLYNK_CHANNEL_DOT_COLOR_SAVE

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_DOT_COLOR_SAVE

Define BLYNK_CHANNEL_HOUR_COLOR_SAVE

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_HOUR_COLOR_SAVE

Define BLYNK_CHANNEL_INTERNAL_COLOR_SAVE

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_INTERNAL_COLOR_SAVE

Define BLYNK_CHANNEL_LIGHT_GROUP_SELECTOR

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_LIGHT_GROUP_SELECTOR

Define BLYNK_CHANNEL_MINUTE_COLOR_SAVE

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_MINUTE_COLOR_SAVE

Define BLYNK_CHANNEL_NIGHT_MODE_BRIGHTNESS

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_NIGHT_MODE_BRIGHTNESS

Define BLYNK_CHANNEL_NIGHT_MODE_TIME_INPUT

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_NIGHT_MODE_TIME_INPUT

Define BLYNK_CHANNEL_NUM_SEPARATION_DOTS

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_NUM_SEPARATION_DOTS

Define BLYNK_CHANNEL_SELECTOR_DOT

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_SELECTOR_DOT

Define BLYNK_CHANNEL_SELECTOR_HOURS

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_SELECTOR_HOURS

Define BLYNK_CHANNEL_SELECTOR_INTERIOR

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_SELECTOR_INTERIOR

Define BLYNK_CHANNEL_SELECTOR_MINUTES

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_SELECTOR_MINUTES

Define BLYNK_CHANNEL_TIMER_START_BUTTON

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_TIMER_START_BUTTON

Define BLYNK_CHANNEL_TIMER_TIME_INPUT

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

BLYNK_CHANNEL_TIMER_TIME_INPUT

Define BLYNK_DEVICE_NAME

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

BLYNK_DEVICE_NAME

Name of this device in the Blynk app.

Define BLYNK_PRINT

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

BLYNK_PRINT

In case the blynk communication is not working this line causes Blynk to send debug output to the serial port. If you are not worried about Blynk or have to diagnose some other issue you can comment this line out.

Define BLYNK_SERVER

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

BLYNK_SERVER

Set the Blynk server address.

Note: I had troubles with using the proper blynk domain so I am using the IP address instead. Maybe this could create problems in the future so it is recommended to use the official domain.

Define BLYNK_TEMPLATE_ID

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

BLYNK_TEMPLATE_ID

Template ID for this device. If you want to use your own custom Template you will have to change this.

Define BRIGHTNESS_INTERPOLATION

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

BRIGHTNESS_INTERPOLATION

How fast the brightness interpolation shall react to brightness changes.

Define CALL_MEMBER_FN

- Defined in file_lib_LED_clock_Modules_Animator_inc_AnimatableObject.h

Define Documentation

CALL_MEMBER_FN(object, ptrToMember)

Define DEFAULT_CLOCK_BRIGHTNESS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DEFAULT_CLOCK_BRIGHTNESS

Default brightness of the display. If you are using blynk you may ignore this setting.

Define DEFAULT_NIGHT_MODE_BRIGHTNESS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DEFAULT_NIGHT_MODE_BRIGHTNESS

Brightness that the clock should be set to while night mode is active.

Define DEFAULT_NIGHT_MODE_END_HOUR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DEFAULT_NIGHT_MODE_END_HOUR

End hour for the night mode.

Define DEFAULT_NIGHT_MODE_END_MINUTE

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DEFAULT_NIGHT_MODE_END_MINUTE

End minute for the night mode.

Define DEFAULT_NIGHT_MODE_START_HOUR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DEFAULT_NIGHT_MODE_START_HOUR

Start hour for the night mode.

Define DEFAULT_NIGHT_MODE_START_MINUTE

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DEFAULT_NIGHT_MODE_START_MINUTE

Start minute for the night mode.

Define DIGIT_ANIMATION_SPEED

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DIGIT_ANIMATION_SPEED

The time it takes for one digit to morph into another.

Define DISPLAY_0_AT_MIDNIGHT

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DISPLAY_0_AT_MIDNIGHT

If set to true the display will show 0 at midnight and 12 otherwise.

Define DISPLAY_FOR_SEPARATION_DOT

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DISPLAY_FOR_SEPARATION_DOT

If set to -1 the flashing middle dot is disabled, otherwise this is the index of the Display segment that should display the dot.

Define DISPLAY_SWITCH_OFF_AT_0

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DISPLAY_SWITCH_OFF_AT_0

If set to true the higher displays will turn off in case they would show 0.

Define DOT_FLASH_INTERVAL

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DOT_FLASH_INTERVAL

Interval in which the dot/s should flash.

Define DOT_FLASH_SPEED

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

DOT_FLASH_SPEED

Length of the dot/s fading animation. One flash fades in and out.

Define ENABLE_LIGHT_SENSOR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

ENABLE_LIGHT_SENSOR

Enable automatic brightness adjustments based on a light sensor.

Define ENABLE_OTA_UPLOAD

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

ENABLE_OTA_UPLOAD

If you want to use OTA upload instead or in addition to the normal cable upload set this option to true. To actually flash something via OTA you have to uncomment the OTA flash lines in the platformio.ini file This is a nice addition to cable upload but it doesn't replace it completely. If the microcontroller crashes because of bad configuration you still have to use a cable.

Define ERROR_COLOR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

ERROR_COLOR

Color of the LEDs signaling an error of some sort.

Define FASTLED_INTERNAL

- Defined in file_lib_LED_clock_Config_Blynk_default_BlynkConfig.h

Define Documentation

FASTLED_INTERNAL

Define FASTLED_INTERNAL

- Defined in file_lib_LED_clock_Modules_Animator_inc_AnimatableObject.h

Define Documentation

FASTLED_INTERNAL

Define FASTLED_INTERNAL

- Defined in file_lib_LED_clock_Modules_Animator_inc_Animator.h

Define Documentation

FASTLED_INTERNAL

Define FASTLED_INTERNAL

- Defined in file_lib_LED_clock_Modules_DisplayManager_inc_DisplayManager.h

Define Documentation

FASTLED_INTERNAL

Define FASTLED_INTERNAL

- Defined in file_lib_LED_clock_Modules_SevenSegment_inc_Segment.h

Define Documentation

FASTLED_INTERNAL

Define FASTLED_SAFE_DELAY_MS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

FASTLED_SAFE_DELAY_MS

the minimum delay between calls of FastLED.show()

Define HOUR_COLOR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

HOUR_COLOR

Color of the hour segments, this will be the default color if blynk functionality is disabled.

Define INTERNAL_COLOR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

INTERNAL_COLOR

Color of the internal LEDs, this will be the default color if blynk functionality is disabled.

Define IS_BLYNK_ACTIVE

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

IS_BLYNK_ACTIVE

If you want Blynk functionality set this to true and set your authentication token. Otherwise set it to false.

Define LED_DATA_PIN

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

LED_DATA_PIN

Pin to which the led strip data pin is connected to.

Define LENGTH

- Defined in file_lib_LED_clock_Config_Animations_12h-extended_Animations.cpp

Define Documentation

Warning: doxygendefine: Cannot find define "LENGTH" in doxygen xml output for project "LED-Clock" from directory: ./doxygen/xml

Define LENGTH

- Defined in file_lib_LED_clock_Config_Animations_24h-clock_Animations.cpp

Define Documentation

Warning: doxygendefine: Cannot find define "LENGTH" in doxygen xml output for project "LED-Clock" from directory: ./doxygen/xml

Define LENGTH

- Defined in file_lib_LED_clock_Config_Transitions_default_SegmentTransitions.cpp

Define Documentation

Warning: doxygendefine: Cannot find define "LENGTH" in doxygen xml output for project "LED-Clock" from directory: ./doxygen/xml

Define LIGHT_SENSOR_AVERAGE

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

LIGHT_SENSOR_AVERAGE

How many measurements shall be averaged. Higher number -> smoother but slower change.

Define LIGHT_SENSOR_MAX

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

LIGHT_SENSOR_MAX

AnalogRead value if the light sensor reads the brightest.

Define LIGHT_SENSOR_MEDIAN_WIDTH

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

LIGHT_SENSOR_MEDIAN_WIDTH

Width of the median calculation. Higher number -> smoother change Should never be higher than the LIGHT_SENSOR_AVERAGE.

Define LIGHT_SENSOR_MIN

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

LIGHT_SENSOR_MIN

AnalogRead value if the light sensor reads complete darkness.

Define LIGHT_SENSOR_PIN

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

LIGHT_SENSOR_PIN

ADC pin to which the light sensor is connected to.

Define LIGHT_SENSOR_READ_DELAY

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

LIGHT_SENSOR_READ_DELAY

Time that should pass before the light sensor is read again. Higher number -> slower adjustments but also changes will be more sudden.

Define LIGHT_SENSOR_SENSITIVITY

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

LIGHT_SENSOR_SENSITIVITY

Value between 0 and 255 that determines how much the light sensor values can influence the led brightness.

Define LOADING_ANIMATION_DURATION

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

LOADING_ANIMATION_DURATION

The time is shall take for one iteration of the loading animation.

Define MINUTE_COLOR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

MINUTE_COLOR

Color of the minute segments, this will be the default color if blynk functionality is disabled.

Define NO_ANIMATION

- Defined in file_lib_LED_clock_Modules_Animator_inc_AnimatableObject.h

Define Documentation

NO_ANIMATION

Macro should be used as a placeholder when an animation step is set to *NO_SEGMENTS*.

Define NO_EASING

- Defined in file_lib_LED_clock_Modules_Animator_inc_AnimatableObject.h

Define Documentation

NO_EASING

Macro that should be used as a placeholder when an animation step does not require any easing.

Define NO_SEGMENTS

- Defined in file_lib_LED_clock_Modules_Animator_inc_AnimatableObject.h

Define Documentation

NO_SEGMENTS

Macro should be used as a placeholder when an animation step is shorter than the animation complexity and no additional segments need to be animated. For that particular position in the step sequence the #animationEffects property should be set to *NO_ANIMATION*.

Define NOTIFICATION_BRIGHTNESS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

NOTIFICATION_BRIGHTNESS

How bright the clock should blink when an alarm or timer was triggered 0 - 255.

Define NTP_SERVER

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

NTP_SERVER

Server for the time.

Define NUM_DISPLAYS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

NUM_DISPLAYS

Number of displays in the shelf.

Define NUM_LEDS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

NUM_LEDS

Automatically calculated total number of LEDs used.

Define NUM_LEDS_PER_SEGMENT

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

NUM_LEDS_PER_SEGMENT

Number of LEDs in each segment.

Define NUM_RETRIES

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

NUM_RETRIES

The number of times the controller tries to connect to wifi before it fails and goes into smartConfig mode (if that is enabled)

Define NUM_SEGMENTS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

NUM_SEGMENTS

Total number of segments that have LEDs in the shelf.

Define NUM_SEGMENTS_PROGRESS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

NUM_SEGMENTS_PROGRESS

The number of segments to use for displaying a progress bar for the OTA updates.

Define NUM_SEPARATION_DOTS

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

NUM_SEPARATION_DOTS

Number of separation dots to use by default (or if no blynk functionality is available) allowed values are 1, 2 and 0 to turn it off.

Define OTA_UPDATE_COLOR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

OTA_UPDATE_COLOR

Color of the LEDs for the OTA update progress bar.

Define OTA_UPDATE_HOST_NAME

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

OTA_UPDATE_HOST_NAME

The host name that shall be used for OTA updates. If you change this here it must also be changed in the platformio.ini file.

Define RUN_WITHOUT_WIFI

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

RUN_WITHOUT_WIFI

If you want to run the system in a minimal mode to test some basic functionality or debug something it could be useful to disable wifi functionality completely.

Define SEGMENT

- Defined in file_lib_LED_clock_Modules_DisplayManager_inc_DisplayManager.h

Define Documentation

SEGMENT(POSITION, DISPLAY)

Macro to shorten then name of the function to make usage easier in the animation config files.

Define SEGMENT_OFF

- Defined in file_lib_LED_clock_Modules_SevenSegment_inc_SevenSegment.h

Define Documentation

SEGMENT_OFF

Helper for easier readability of the complex animation definitions.

Define SEPARATION_DOT_COLOR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

SEPARATION_DOT_COLOR

Color of the separation dot LEDs, this will be the default color if blynk functionality is disabled.

Define TIME_MANAGER_DEMO_MODE

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

TIME_MANAGER_DEMO_MODE

enable for wifi less operation or to demo all the animations

Define TIME_SYNC_INTERVAL

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

TIME_SYNC_INTERVAL

Time in seconds for the interval in which the time should be synchronized with the time server.

Define TIME_UPDATE_INTERVAL

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

TIME_UPDATE_INTERVAL

How often the time is checked and the displays are updated.

Define TIMER_FLASH_COUNT

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

TIMER_FLASH_COUNT

Number of flashes until an alarm is considered complete and the system goes back to normal.

Define TIMER_FLASH_TIME

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

TIMER_FLASH_TIME

Flash the current time in case a timer is expired instead of flashing 00:00.

Define TIMEZONE_INFO

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

TIMEZONE_INFO

Enter the string for your timezone according to this webpage: https://remotemonitoringsystems.ca/time-zone-abbreviations.php.

Define USE_24_HOUR_FORMAT

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

USE_24_HOUR_FORMAT

If set to true 24 hour format will be used. For this one additional column is needed in the shelf to display it correctly.

Define USE_ESPTOUCH_SMART_CONFIG

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

USE_ESPTOUCH_SMART_CONFIG

Use the ESP smart config to setup the wifi network. If you want to set it manually set this to false.

Define USE_NIGHT_MODE

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

USE_NIGHT_MODE

Whether to activate night mode or not. If you want the clock to reduce brightness/switch off during certain hours set this to true. If you are using Blynk to control the settings of your clock you may ignore the default settings as they can be changed dynamically during runtime in that case.

Define WIFI_CONNECTING_COLOR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

WIFI_CONNECTING_COLOR

Color of the LEDs while searching for a WIFI network.

Define WIFI_CONNECTION_SUCCESSFUL_COLOR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

WIFI_CONNECTION_SUCCESSFUL_COLOR

Color of the LEDs signaling a successful WIFI connection.

Define WIFI_PW

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

WIFI_PW

Define WIFI_SMART_CONFIG_COLOR

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

WIFI_SMART_CONFIG_COLOR

Color of the LEDs if system is waiting for WIFI smart config.

Define WIFI_SSID

- Defined in file_lib_LED_clock_Config_Setup_24h-clock_Configuration.h

Define Documentation

WIFI_SSID

WIFI_SSID and WIFI_PW are only needed if smart setup is disabled.

Typedefs

Typedef NUMBER

- Defined in file_lib_Easings_src_EasingConstants.h

Typedef Documentation

typedef double **NUMBER**

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

A

B

C

D

E

F

H

I

L

M

N

O

Q

R

S

T

U

W