

## Stilo

### LED Lights and Aquarium Controller

- Two separate PWM signals for independent control of white and blue LEDs;
- dusk and dawn effect;
- real time clock;
- water and LEDs heatsink temperature sensors;
- speed control for heatsink fans;
- all settings are stored in EEPROM.

## Changes:

### v.2.5

- Updated to latest ITDB02\_Graph16 library that support more screen types;
- LEDs output is set in percent (not 0-255 values);
- Some general visual changes;
- Added configuration of LED fans temperatures;
- Safety feature - dimming LEDs if heatsink is too hot;
- Can restore settings on start or by option in menu;
- Temperature alarm can be acknowledged to turn off buzzer;
- Removed control of heater and cooler;
- Added 25kHz signal to control LEDs heatsink fans speed.

### v. 2.1

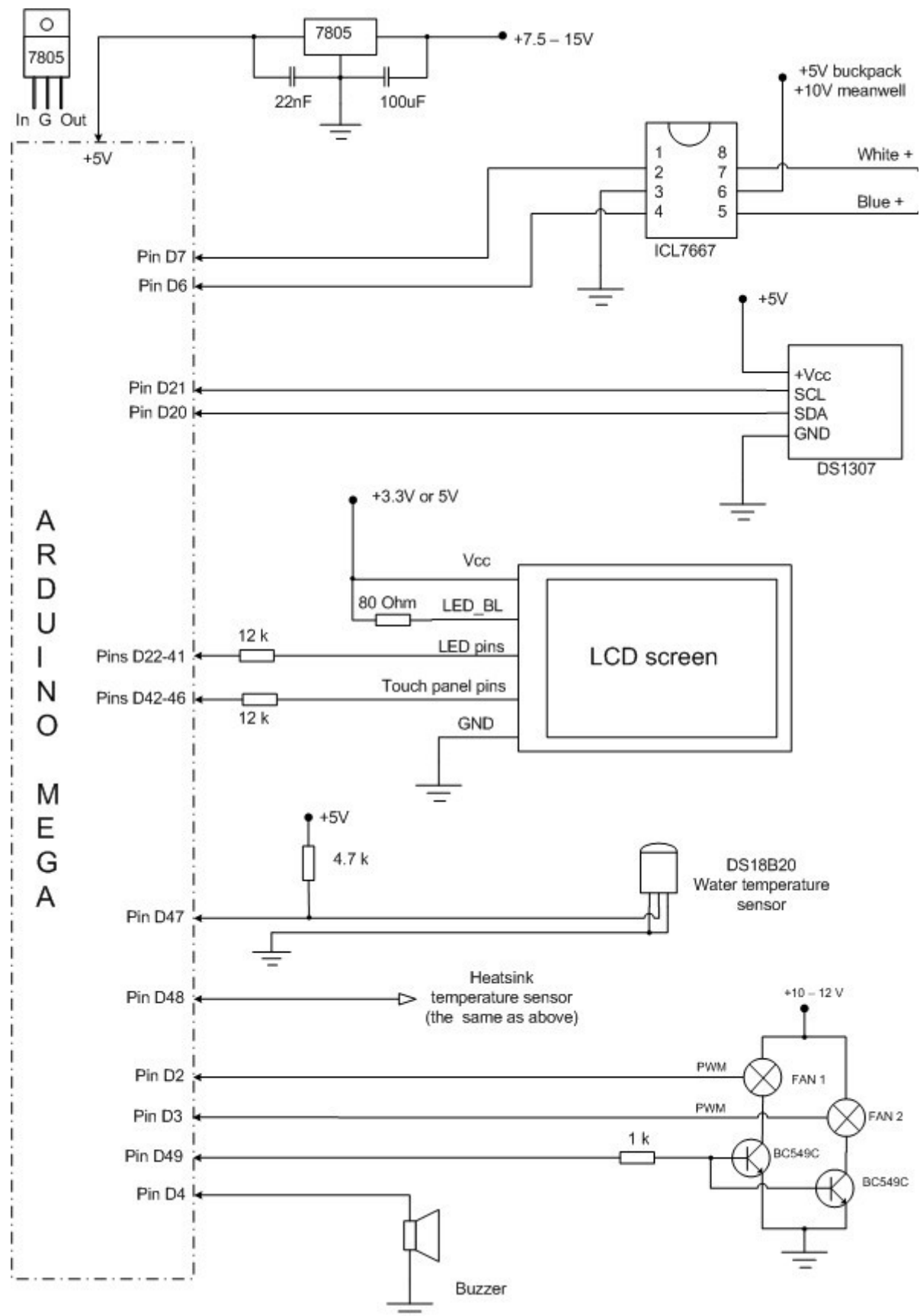
- Water temperature sensor changed to digital DS18B20;
- Pins to control water heater, cooler and for alarm buzzer;
- Led heatsink temperature sensor (DS18B20);
- Speed control for heatsink fans;
- Led values can be changed from menu;
- Setting can be stored in EEPROM.

### v. 1.5

- Minor changes to design;
- Added temperature probe and temperature control page;
- Tidied up the code.

### v.1 initial release

## Schematic:



## LCD screen pins:

40-pin LCD connector		Arduino	40-pin LCD connector		Arduino
1	GND	GND	21	LCD_DB0	37
2	VCC	*	22	LCD_DB1	36
3	N.C.		23	LCD_DB2	35
4	LCD_RS	38	24	LCD_DB3	34
5	LCD_WR	39	25	LCD_DB4	33
6	LCD_RD	+3.3V	26	LCD_DB5	32
7	LCD_DB8	22	27	LCD_DB6	31
8	LCD_DB9	23	28	LCD_DB7	30
9	LCD_DB10	24	29	TouchP_CLK	46
10	LCD_DB11	25	30	TouchP_CS	45
11	LCD_DB12	26	31	TouchP_DIN	44
12	LCD_DB13	27	32	TouchP_BUSY	
13	LCD_DB14	28	33	TouchP_OUT	43
14	LCD_DB15	29	34	TouchP_Penirq	42
15	LCD_CS	40	35	SD_OUT	
16	N.C.		36	SD_SCK	
17	LCD_RESET	41	37	SD_DIN	
18	N.C.		38	SD_CS	
19	LED_BL	*	39	N.C.	
20	N.C.		40	N.C.	

## PARTS

---

**LCD screen** – the screen is 320\*240 pixels, 3.2" TFT LCD module and must have one of following controllers (and working with 16-bit interface):

- HX8347-A
- ILI9325D
- ILI9327
- SSD1289

and has ADS7843 compatible touch panel controller.

I bought mine on ebay but they are available from number of online shops including <http://iteadstudio.com/store>, <http://www.electronicsfreaks.com/store> and others.

**Arduino Mega** - older version with ATmega1280 microcontroller but the sketch should work with newer Mega 2560 as well.

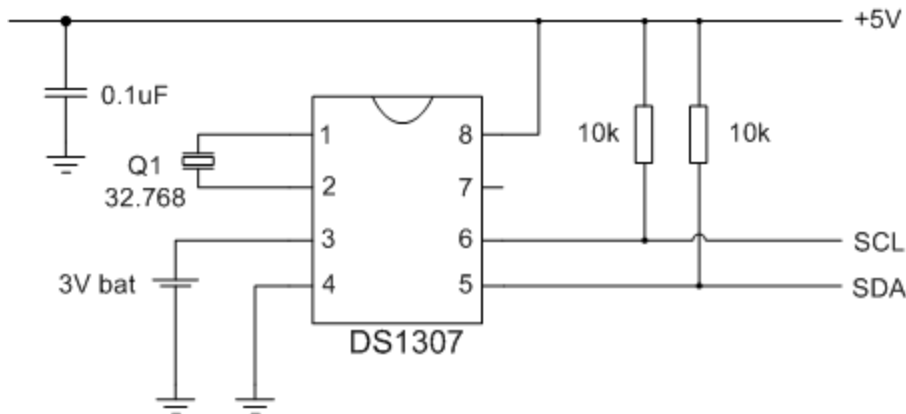
**Proto shield** for Arduino Mega, I have made one myself from generic proto board but they are available from many shops.

**RTC** (real time clock), already assembled or it can be made from parts:

- DS1307 chip
- 32.768kHz Crystal

- CR2032 battery and holder
- 2 x 10kOhm resistors
- 0.1uF capacitor

Below is schematic if you build it from parts:



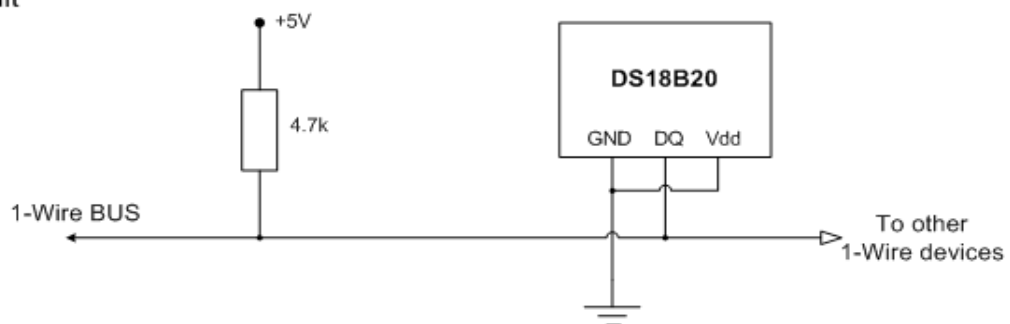
**ICL7667 chip** - its role is to adjust signal from Arduino (5v, max 40mA) to whatever is required by LED drivers. In my case to 10V required by MeanWell drivers and I also use 10V to power all electronics and fans.

**7805** voltage regulator to change 10V from my power supply to 5V required by Arduino and LCD.

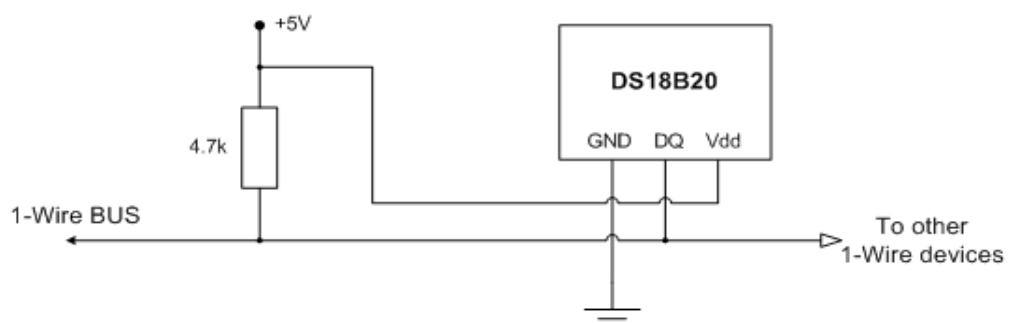
2 x **DS18B20** temperature sensors (one waterproof to monitor water temperature).

They can be connected to Arduino using 2 wires or 3 wires cables as per below schematic:

2 wire circuit



3 wire circuit



I have used both sensors on separate Arduino pins for simplicity but they can be connected to single pin, though it will require small change to the code so each sensor is addressed by its unique serial number.

**Various** resistors, transistors, connectors, wires.

**Heatsink fan** - PC fan with PWM control, they are recognized by having 4 wires instead of usual three. The fourth wire (usually blue) is used to control fan speed. The specification says PWM frequency must be between 21-28 kHz so Stilo is changing the timer on pins 2 and 3 to adjust frequency of PWM signal. The fans I use are Arctic Cooling 12 PWM AC. They are very quiet but they don't turn off at PWM 0%, so I have used transistors to switch them off.

**Arduino 0023 software** - can be downloaded from [www.arduino.cc](http://www.arduino.cc)  
Current sketch is not compatible with Arduino 1.0 yet as most of libraries I use are not working with new Arduino release.

## INSTALATION

---

I really recommend to buy a proto breadboard and jump wires, it's not very expensive and it can save a lots of frustration while building and testing projects.

If you are new to Arduino then first make sure it's all working: install Arduino software and necessary drivers, connect USB cable to Arduino board and upload one of simple examples like 'Blink' to ensure everything is fine. Then play with some example sketches to get familiar with Arduino environment.

To start with Stilo - first copy libraries from zip to /arduino/libraries.

Next thing is to make screen running - connect it to Mega board (i.e. using breadboard). The screen has 40-pin connector so I used ATA cable (40-wire, old type). Data pins are connected through 12k resistors and the 80ohm resistor controls LCD brightness (the brightness can be adjusted by changing resistor value).

Check if the screen is working by uploading example from /arduino/libraries/ITDB02\_Graph16/examples/ITDB02\_Graph16\_Demo\_Landscape.

The touch screen - upload touch example from /arduino/libraries/ITDB02\_Touch/examples/ITDB01\_Touch\_ButtonTest and check if it's working.

Touch screen needs to be calibrated, this is done by running calibration sketch located in examples and then following on screen instructions.

At this point we can connect RTC to the Arduino and upload Stilo sketch and check if it display time and date.

The last thing is adding the rest of components - temperature sensors, resistors, transistors and test all Stilo functions.

## LEDS and drivers

The controller can be used with any LED driver that is PWM dimmable (and will accept 480Hz signal from Arduino). The most popular are Meanwell and Buckpucks but there are plenty of others.

LEDs and drivers - I'm not going into much details as it always depends on types of LEDs and drivers used. There is plenty of information on the net, specifically check DIY and lightning sections on all popular marine forums.

Choice of LEDs - another topic I'm not able to cover here. I have started with (recommended at the time) 50/50 XPE royal blues and HR-E cold whites but soon discovered (as other people using the same setup) that this combo gives washed out effect. Since then I changed some of cold whites for neutral whites but still I'm not 100% happy with the effect.

There are couple of good threads on marine forums about this topic (search for 'leds aesthetics'). Right now I'm thinking about changing some of royal blues for cold blues, adding few violet leds and maybe couple of reds for good measure.

## Pins settings

Those settings specifies pins on board used for specific parts, change them if needed:

<pre>//ITDB02 myGLCD(38,39,40,41, ITDB32WC); //ITDB02 myGLCD(38,39,40,41, ITDB32S); //ITDB02 myGLCD(38,39,40,41, TFT01_24); <b>ITDB02</b> <b>myGLCD(38,39,40,41);</b></pre>	Uncomment one line depending on the screen type.  Also adjust pins numbers if needed.
<pre><b>ITDB02_Touch</b> <b>myTouch(46,45,44,43,42);</b></pre>	Pins for touch controller
<pre><b>#define ONE_WIRE_BUS_W 47</b> <b>#define ONE_WIRE_BUS_H 48</b></pre>	Pins for temperature sensors
<pre><b>const int ledPinBlue = 6;</b> <b>const int ledPinWhite = 7;</b></pre>	Pins for leds PWM signal
<pre><b>const int AlarmBuzzPin = 4;</b></pre>	Pin for piezo buzzer
<pre><b>const int fan1PWMpin = 2;</b> <b>const int fan2PWMpin = 3;</b></pre>	Pins for fans PWM signals (25kHz)
<pre><b>const int fan1TranzPin = 49;</b></pre>	Pin for transistor that turns on/off fans

## Sketch settings

<code>boolean BUCKPUCK = false;</code>	True if buckpuck style drivers are used
<code>const boolean readEEonStart = 0;</code>	If true the settings saved in EEPROM are loaded on start, if false settings can be loaded by option in menu
<code>const int fanHyster = 300;</code>	Temperature difference between fans turning on and off *1)
<code>float setTempC = 27.0;</code>	Desired temperature for water temp. alarm
<code>float AlarmOffTemp = 0.0;</code>	Difference from setTempC when alarm in on (if '0' alarm is off)
<code>int HtempMin = 30;</code>	Temperature when fans starts (also check fanHyster)
<code>int HtempMax = 40;</code>	Temperature when fans speed is 100%
<code>int cutOffHtemp = 70;</code>	Temperature when controller starts to dim LEDs, for each °C above this value light is dimmed 10% (if '0' this mechanism is turned off)

1) In the code used to control 25kHz PWM, value 639 represents fans speed of 100% so 300 is almost 50%. This means that fans will turn on when LEDs temperature is about half way between **HtempMin** and **HtempMax**. In example here it's 30 and 40 °C so the fans will start at 35 °C with 50% speed and stop at 30 °C. I found the fans to be very effective and without this mechanism they kept to turn on and off every few seconds.

## LEDs output settings:

```
// Blue leds output values %:
byte bled[96] = {
  0, 0, 0, 0, 0, 0, 0, 0,  //0 - 1
  0, 0, 0, 0, 0, 0, 0, 0,  //2 - 3
  0, 0, 0, 0, 0, 0, 0, 0,  //4 - 5
  0, 0, 0, 0, 0, 0, 11, 11, //6 - 7
  11, 11, 11, 12, 12, 13, 13, 14, //8 - 9
  16, 16, 16, 16, 20, 25, 30, 35, //10 - 11
  40, 47, 47, 47, 47, 47, 47, 47, //12 - 13
  47, 47, 47, 47, 47, 47, 47, 47, //14 - 15
  47, 47, 47, 47, 47, 47, 47, 47, //16 - 17
  47, 47, 47, 47, 47, 40, 35, 30, //18 - 19
  25, 20, 15, 11, 11, 11, 11, 11, //20 - 21
  0, 0, 0, 0, 0, 0, 0, 0,  //22 - 23
};
```



```
// White leds output values %:
byte wled[96] = {
  0, 0, 0, 0, 0, 0, 0, 0, //0 - 1
  0, 0, 0, 0, 0, 0, 0, 0, //2 - 3
  0, 0, 0, 0, 0, 0, 0, 0, //4 - 5
  0, 0, 0, 0, 0, 0, 0, 0, //6 - 7
  0, 0, 0, 0, 0, 0, 0, 0, //8 - 9
  0, 0, 0, 11, 14, 20, 26, 37, //10 - 11
  47, 55, 55, 55, 55, 55, 55, 55, //12 - 13
  55, 55, 55, 55, 55, 55, 55, 55, //14 - 15
  55, 55, 55, 55, 55, 55, 55, 55, //16 - 17
  55, 55, 55, 55, 47, 37, 26, 19, //18 - 19
  11, 0, 0, 0, 0, 0, 0, 0, //20 - 21
  0, 0, 0, 0, 0, 0, 0, 0 //22 - 23
};
```

Those two arrays keep LEDs output values for blue and white channel respectively. Each value is the light output in 15 min intervals. Each line represent 2 hours. So the line:

```
25, 20, 15, 11, 11, 11, 11, 11, //20 - 21
```

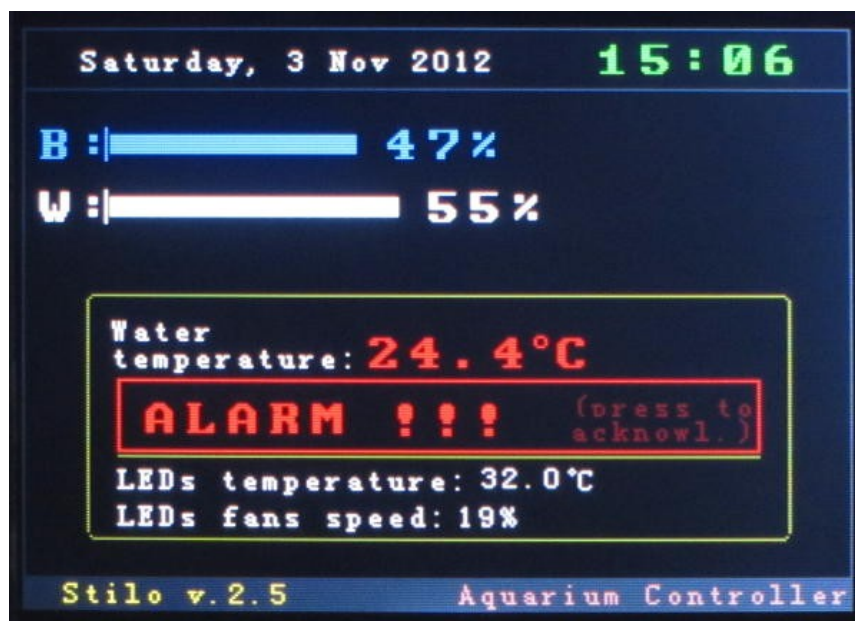
gives us 25% output at 20.00, 20% at 20.15, 15% at 20.30 and so on.

The light output is adjusted in 1 min intervals and those values are calculated in real time based on previous and next value.

Those two tables are loaded on start, however they can be overridden if the light outputs have been previously saved into EEPROM and the option `readEEonStart` is on.

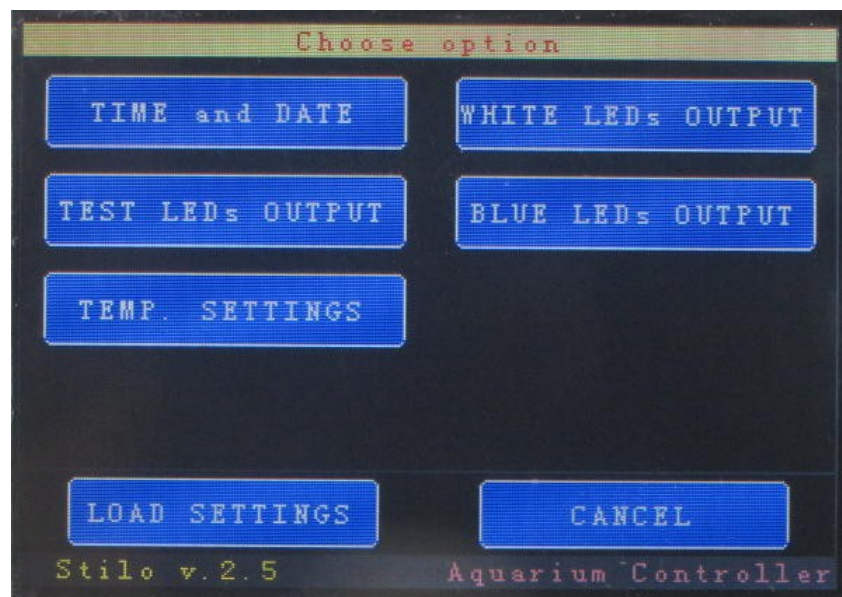
## OPERATION

### Main screen with alarm on:



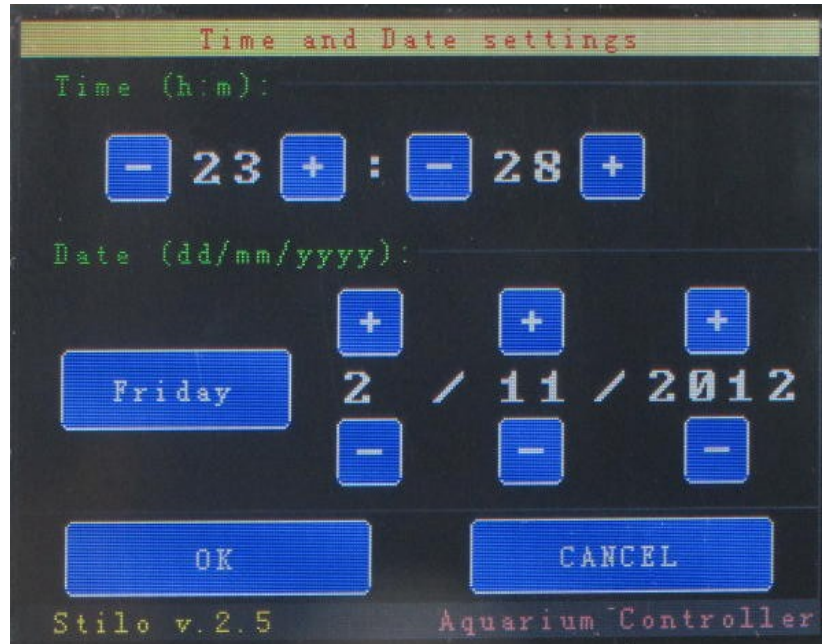
Pressing the red square acknowledges alarm and mutes the buzzer.

## Menu:



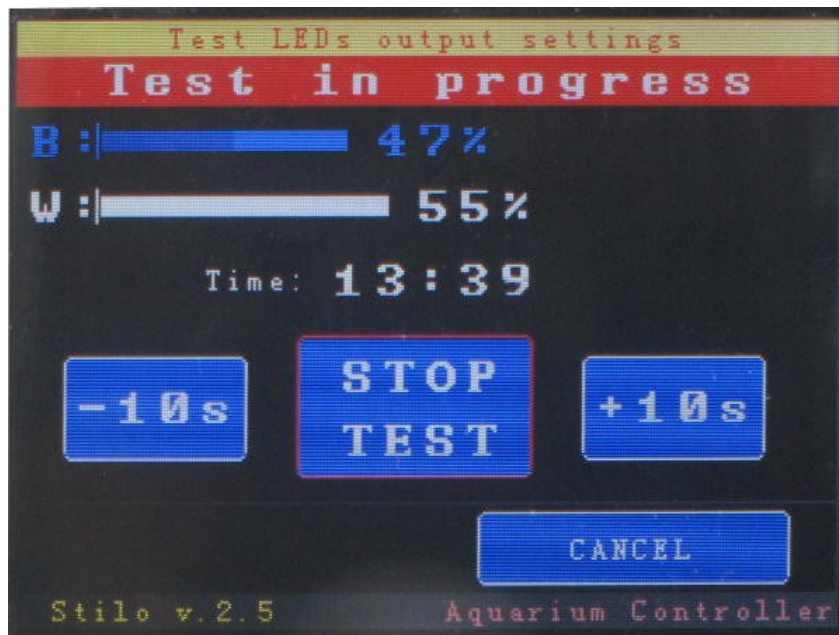
Option 'LOAD SETTINGS' appears only if `readEEonStart = 0`.  
It loads saved settings from EEPROM memory.

## TIME and DATE:



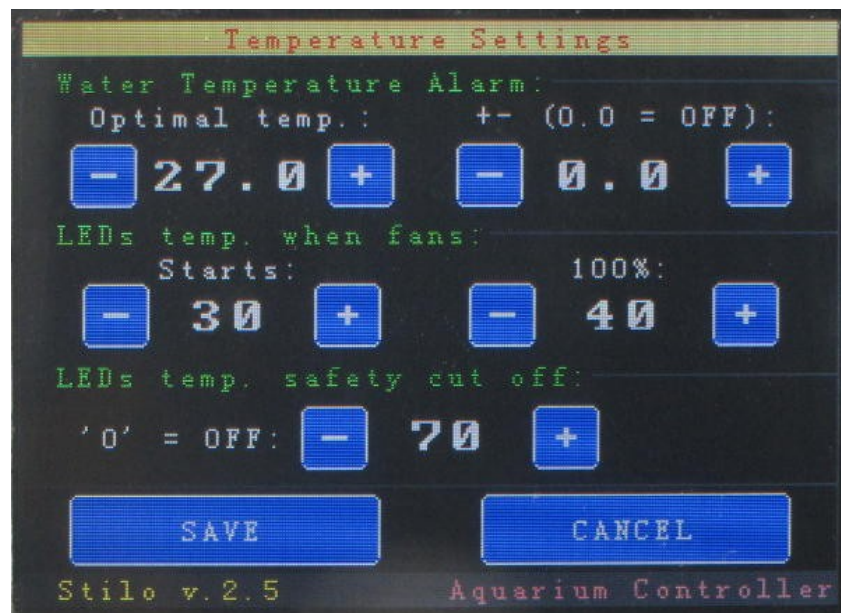
Allows changing time and date.

## TEST LEDs OUTPUT:



Allows to test configured light output over time and LED colour blending.

## TEMP. SETTINGS:

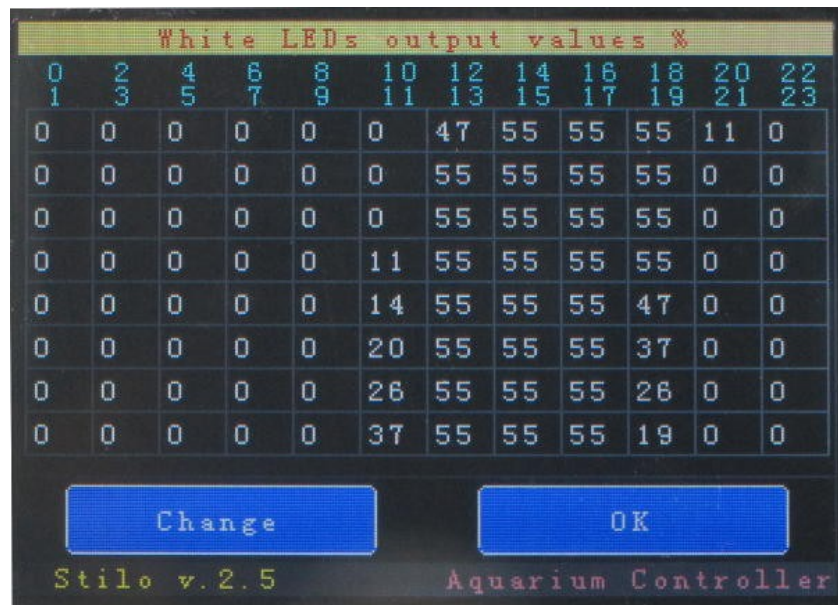


*Water temperature alarm:* optimal temperature value is self explanatory. '+' value determines at what temperature alarm goes off (when water temperature is above 'optimal plus +- value' or below 'optimal minus +- value'). If it's '0.0' then the alarm is turned off.  
*LEDs temp. when fans starts* and their speed reach 100%, see also **fanHyster** value.



*LEDs temp. safety cut off:* LEDs temperature at which controller starts to dim lights. For each °C above this value lights are dimmed 10%. Set '0' to turn off this feature.  
'SAVE' applies the changes and saves the values to EEPROM memory.

## WHITE (BLUE) LEDs OUTPUT:



Shows light outputs for chosen channel. The values can be changed by choosing 'Change':



Selecting the time on top shows light values that can be changes with '+' an '-' buttons.  
'Save' applies the changes and saves the values to EEPROM memory.