LEDController v2.0

DIY aquarium LED light fixture.

Christian Rugland
christian@rugland.se

2014-09-28

Abstract

Finding the correct lighting for your salt water aquarium can be troublesome, there is tons of information out there and finding the best solution can be tricky. Metal Halide lighting, T5/T8 tubes and LED's, all have their advantages and disadvantages. Something I found valuable when choosing the kind of lighting I wanted was the flexibility and small size of different LED chips available, the prize was somewhat higher but the yield in lifetime and lower energy bill made it more economical than other alternatives.

There are some already prebuilt fixtures out there, though I found them lacking in both light spectrum and they were often expensive, buying a chip with LED's or single LED's looked as a better option.

The setup would require drivers, power supply and some kind of control device as well as a fixture to mount them on. Apart from the total prize of this setup which didn't look that amusing I found that they were not that suited for a higher number of LED's, buying and sticking 5-6 drivers looked ugly and non-logical. Since I'm close to finishing my Master's degree in electrical engineering the only solution to me was to build my own driver and controller, thus increasing the fixtures suitability to my needs and hopefully the total cost.

Since there is a growing community out there for DIY's I made a report/manual so others can copy/use the work I've done. By modifying the components slightly and changing the power supply any number of LED's can be used, thus increasing the flexibility of the fixture such as an upgrade to 2 LED chips or even more would probably only involve a new power supply. Changing to any number of single LED's connected in series of 5 would also be possible, just make sure the dimensions/properties of the components used are correct for the channels.

Orcad Capture layouts, Orcad Simulation profiles, Eagle circuit layout and BOM are provided along with this document.

1 Theory 1
1.1 Main description 1
1.2 Led brightness through PWM control 1
1.3 LED Channel Control 1
1.4 Simulation 3
1.5 Microcontroller Supply Voltage 3
1.6 Fan Control and Supply Voltage 4
1.7 Circuit 5
2 Software 7
2.1 Alarms 7
2.2 Dimmer 7
2.3 Time 7
2.4 EEPROM 7
2.6 Temperature sensor 7
2.5 Optimizations 7
3 Hardware 7
3.1 Circuit board layout 7
3.2 Cables 8
3.3 Button and Potentiometer 8
3.4 LCD 9
3.5 Temperature Sensor 9
3.6 DS3231 RTC 9
3.7 DC jacket and Rocker switch 10
3.8 Heatsink 11
3.9 Suspension 12
4 Testing 12
4.1 Cheap DIY oscilloscope 12
4.2 Measurements 12
4.3 Fan Control and PWM Frequency adjustment 13
4.4 Extended Testing 13
4.5 Finalization 15
4.6 LED Channel schedule 17
5 Summary 17
6 Acknowledgements 18
7 References 18

1 Theory

1.1 Main description

The main function of the controller to be built is to be able to control and dim the brightness of the LED channels, the LED chip i used has 5 different channels of 5 LED's each in series but in theory any number of LED's could be used with the correct power supply and maybe some value altering of the components used.

As stated the LED chip I used, the *Lumia 5.2*[1] has 5 channels of 5 LED's in series each. The different channels has different forward voltages at different currents, dimensioning of the components will be based on the maximum forward voltage stated in the datasheet (since finding the correct simulation files for all the LED's was not possible).

Aside from controlling the channels a LCD will be used to show the current time, brightness in percent for each channel, alarm times for the channels, speed of the fan used to cool the heatsink as well as the temperature inside the led chip compartment of the heatsink. By scrolling with a potentiometer and using a button these options can be changed through the LCD interface.

1.2 Led brightness through PWM control

The microcontroller used, *Arduino Pro Mini*[2] can only output a maximum of 40mA per each IO channel (with a total of 200mA for all IO channels). Each led channel will be using from 700-1000mA at full recommended brightness so using the Arduino IO channels as power supply is not viable (The +5V line of the Arduino has a maximum of 500mA i.e. not suitable as well).

Instead transistors will be used where the Arduino IO channels will control the base of the transistor, to save led lifetime, lowering heat dissipation as well as the electric bill PWM control will be used. *PWM (Pulse-Width Modulation)*[3] sends pulses which has a fixed frequency, by changing the "ON" time in each period the brightness of the LED's can be controlled (and dimmed).

The frequency provided by default in this microcontroller is around 490Hz for all PWM channels except D6 and D5 which has 980Hz, since this is above ~50Hz (used for AC voltage since human eye cannot notice flickering at frequencies above this value) these values will be used for the LED's (except Channel 1, see 4.3).

When controlling high power LED's it is common to keep the current constant since small changes in voltages will induce high changes in current. To be able to control the brightness as well as dim the LED's current is instead changed, this is done by adding a current limiting resistor to the emitter of the NPN transistor and dimension it based on the highest current used for the channel (for full brightness).

1.3 LED Channel Control

The transistor has a voltage drop between the base and emitter of ~0.7 voltage (depends on which transistor used), as can be seen in fig. 1 a Darling transistor is used so the voltage drop will be around 1.4V (2*0.7V) in this setup (Resistor subscripts used in equations is following fig. 1) . We want to minimize the heat dissipation so keeping the current limit resistor low is desirable. Current through the LED's will be approximately followed by equation 1.

$$I_c \approx 0.99 * I_E \approx \frac{V_E}{R_E} = \frac{V_B - V_{BE}}{R_E} = \frac{V_B - 1.4}{R_E}\ A \quad (1)$$

R_E is chosen as 0.56Ω for all channels then a voltage divider is used to alter V_B for the different channels to attain the desired current (The Arduino IO outputs PWM level of 5V). Datasheet for the channels can be found at [3] . The maximum recommended currents used for channel 1, 4, 5 are 700mA and for channel 2, 3 are 1000mA, the power needed to be dissipated as heat through the current limiting resistor will be:

$$P_{R_{E1,4,5}} = {I_E}^2 * R_E = 0.7^2 * 0.56 = 0.2744\ W \quad (2)$$

$$P_{R_{E2,3}} = {I_E}^2 * R_E = 1.0^2 * 0.56 = 0.56\ W \quad (3)$$

As such ceramic resistors are used (capable of up to 4-5 W).

The current then depends on the voltage V_B altered by the voltage divider according to the following equation:

$$V_B = \frac{R_4}{R_3 + R_4} * V_5 = \frac{R_4}{R_3 + R_4} * 5\ V \quad (4)$$

Current can then be described with the help of equation 1 and 4 :

$$I_c \approx \frac{\frac{R_4}{R_3 + R_4} * 5 - 1.4}{0.56} \quad (5)$$

The voltage divider resistors are then adjusted accordingly to the desired current.

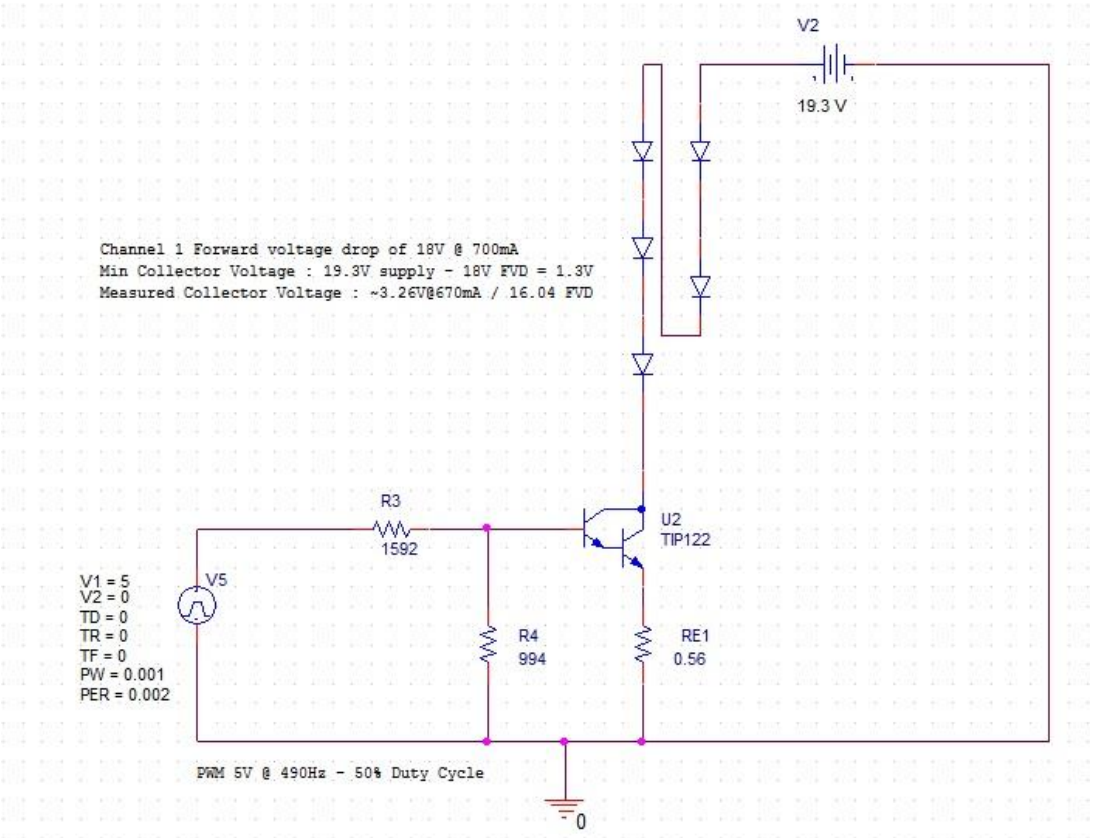

Fig 1. Channel 1 circuit diagram.

1.4 Simulation

Ordering, soldering and testing different transistors and resistors can be a troublesome and expensive matter, a better solution is to first simulate the circuits. The ”true” voltage drop of the led channels could not be simulated correctly due to problems finding the correct simulation files for each led (as well as info on what particular led brand was used for some LED’s, mainly the exotic ones), instead the stated maximum forward voltage drop was used and the circuit simulated with the power supply VCC-Max_{FD} (fig. 1. Shows the circuit with diode symbols used as LED’s, the actual simulation did not include these components, only the voltage after the total voltage drop at the collector, see included simulation files).

The transistor used was TIP122, mainly due to its low price and its ability to sustain high collector currents, simulation gave the following values for the voltage divider circuits (subscripts as fig.1):

Channel	R_3	R_4	V_B
1	1600	1000	1.75 V
2	1300	1000	2.99 V
3	1300	1000	2.98 V
4	1500	1000	2.2 V
5	1600	1000	1.7 V

Table 1. Voltage divider.

1.5 Microcontroller Supply Voltage

The microcontroller used requires 7-12 unregulated voltage, the power supply used for the LED’s for this setup is a 19V / 6.3A power adapter. It is desirable to use the same power supply for the microcontroller as well as the LED’s to reduce the amount of hardware used. The voltage needs to be regulated/lowered down within the 7-12 V level, one way of doing this is through a voltage regulator, either switched or linear but since the current required by the microcontroller probably will not exceed 100mA in this setup a cheaper implementation would be to use a Zener diode. The Zener diode will have a somewhat fixed voltage drop (depending on temperature and current), important to check is that the voltage drop is as linear as possible for different loads (i.e. check voltage drop vs. current in datasheet).

A Zener diode with a voltage drop of 10V (Max 5W) was chosen based on its linearity (11V was not available) in series with a regular diode giving a microcontroller supply voltage of 19.3-10-0.7≈ 8.6V, the setup can be seen in fig. 2 (figure without load, voltage will be a bit lower with ~100mA load).

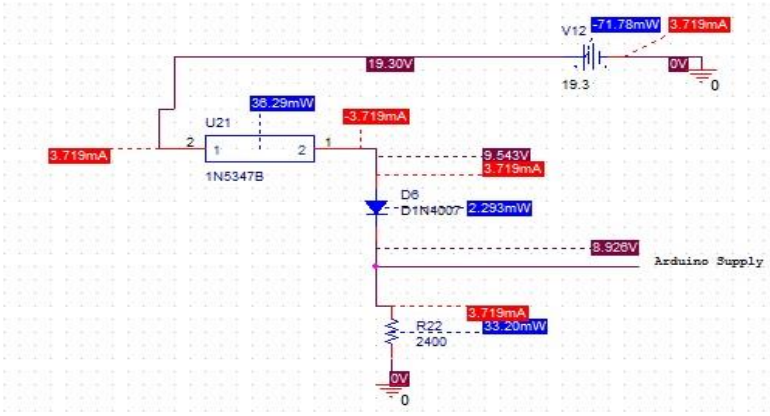

Fig. 2. Arduino Power Supply

1.6 Fan Control and Supply Voltage

A fan is to be used installed above the heatsink, the Arduino Pro Mini has 6 PWM outputs so the fan will be controlled in a similar fashion as the led channels (without the current limiting resistor, since fan will only draw the current it needs). Similarly as the Arduino supply voltage the 19.3 voltage is lowered down to the required 12V of the fan with the use of a 7.5V Zener diode. A flyback diode is used in parallel of the fan because of the induced voltage spikes when rapidly shutting the transistor on and off. The circuit can be seen in fig. 3.

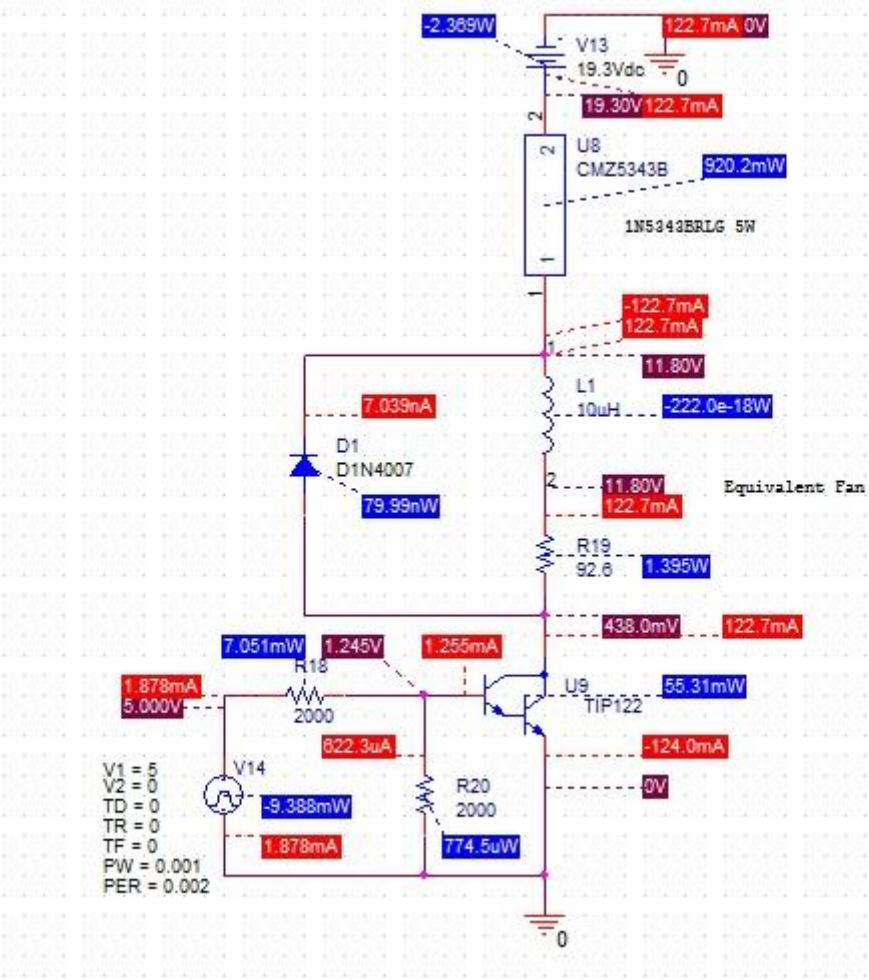

Fig. 3. Fan Control and Supply circuit.

1.7 Circuit

The final circuit is given by figure 4.

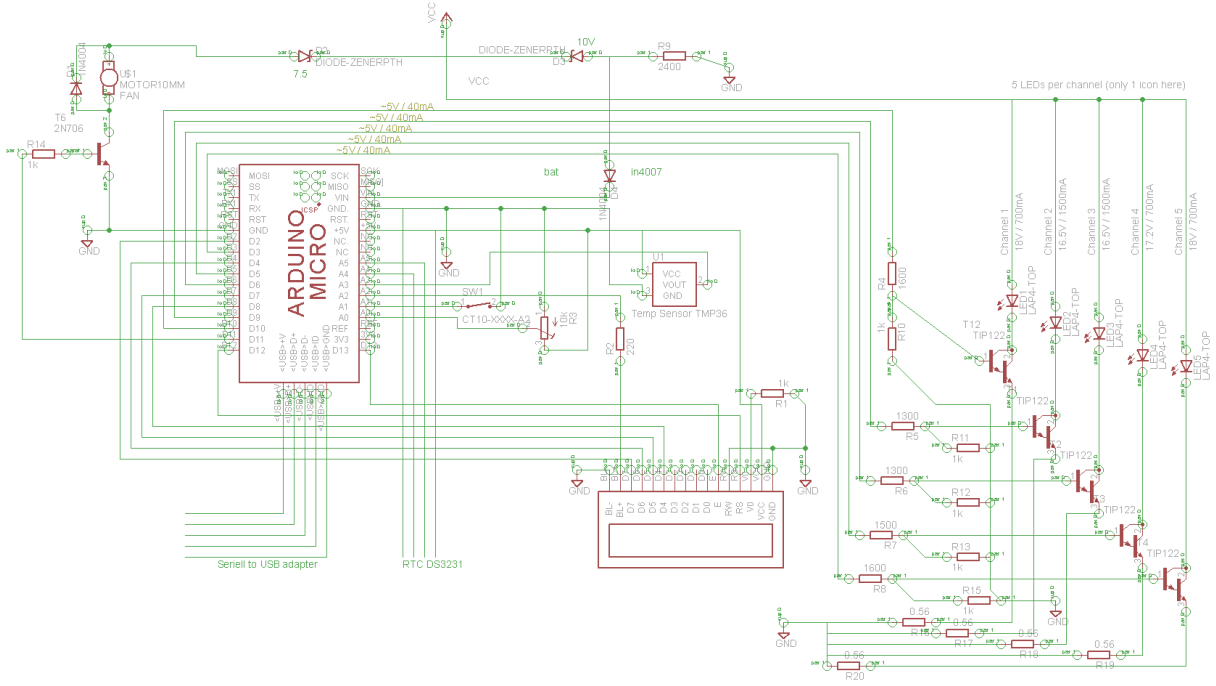

Fig. 4. Final Circuit.

A couple of pictures of the finished circuit (amateur soldered with crappy soldering tips!).

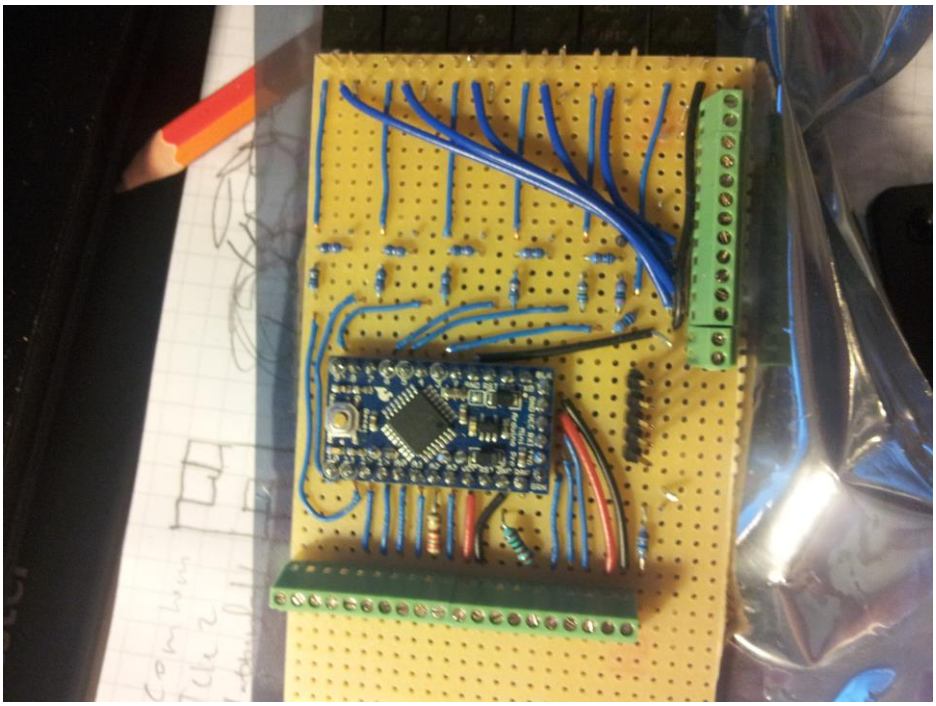

Fig. 5. Circuit topside.

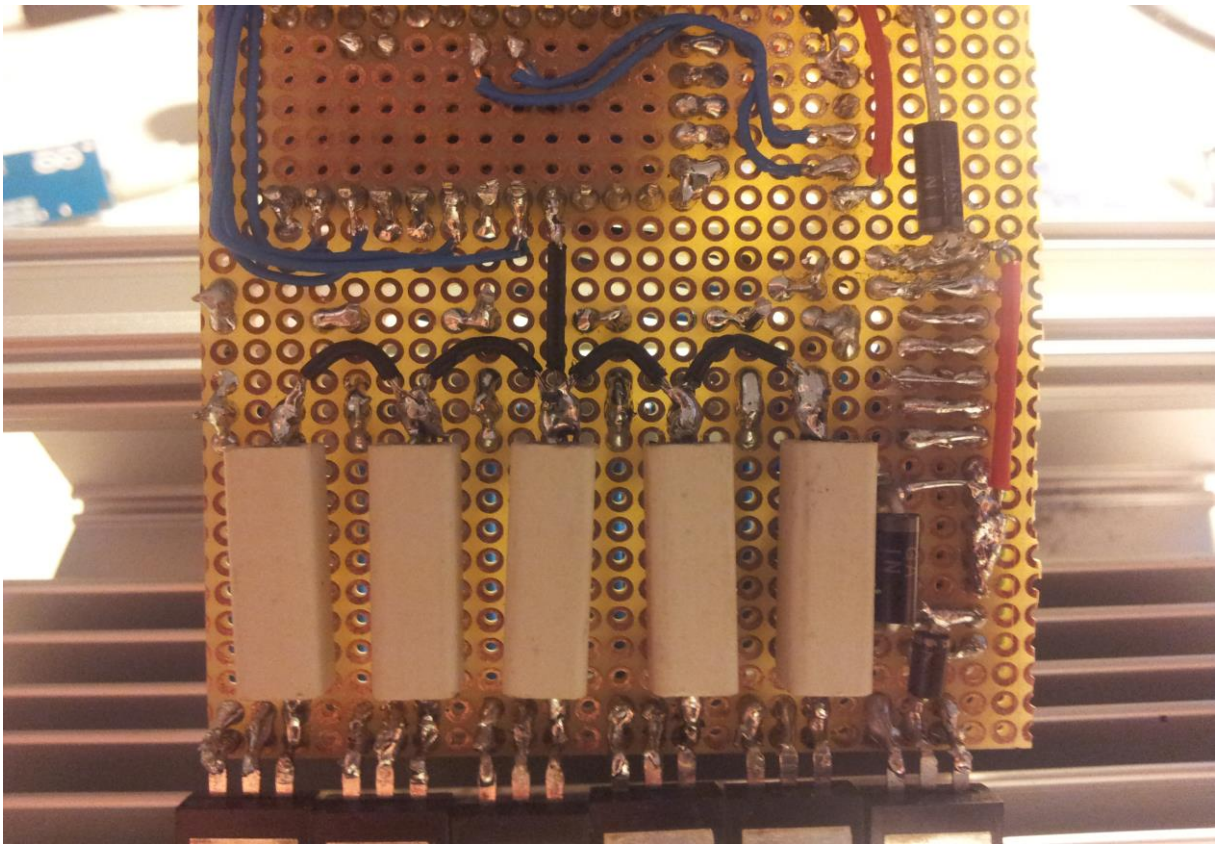

Fig. 6. Circuit Bottom 1/2.

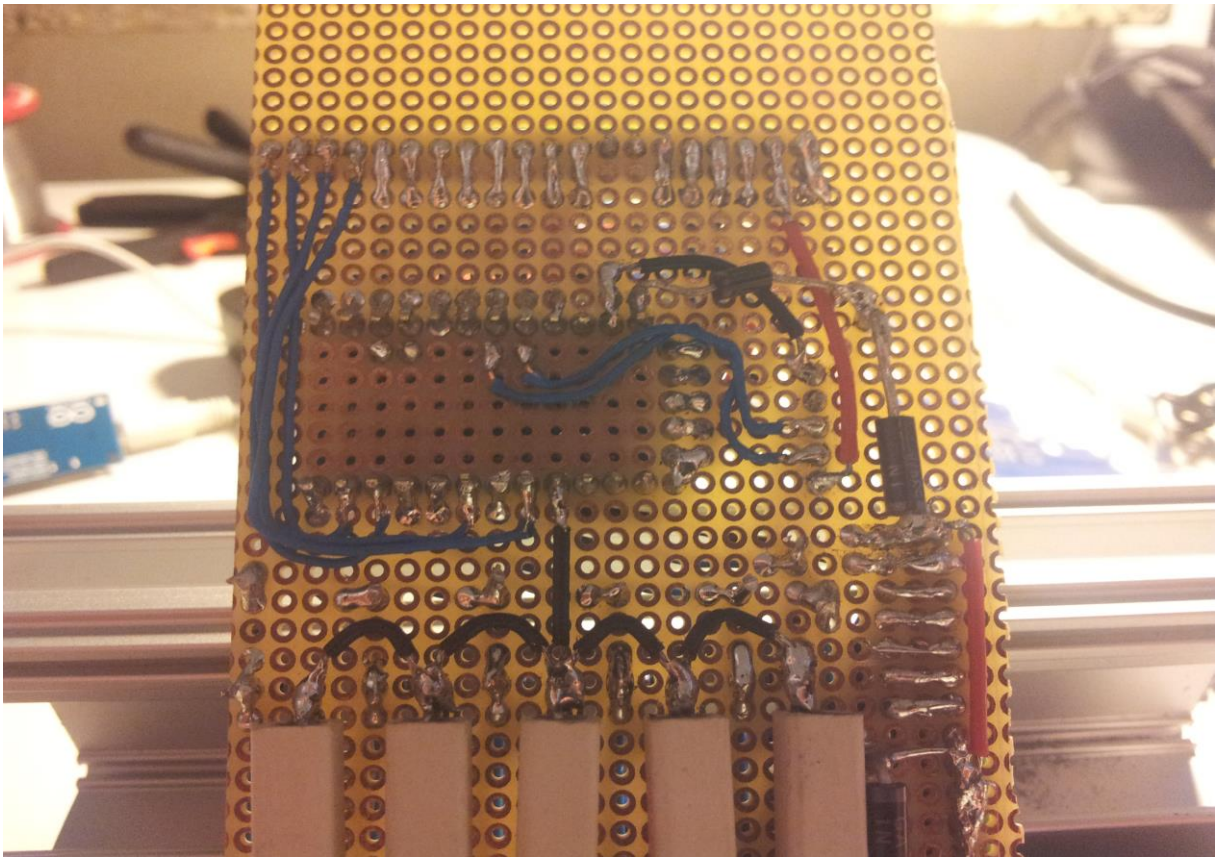

Fig. 7. Circuit Bottom 2/2.

2 Software

2.1 Alarms

Alarms are implemented for stopping and starting the individual led channels, PWM values for start/stop can be edited through the LCD (Alarm times as well), this also applies for the Fan.
The PWM values can also be adjusted directly during runtime.

2.2 Dimmer

Led brightness is not linear with its current, hence a specific formula has been implemented to linearize the brightness during the dimming of the LED's to mimic sunrise and sundown. The dimming time is default at 30min@255 PWM (Arduino PWM goes from 0-255). This is currently not adjustable (perhaps future implementation).

2.3 Time

Initially the time was based on the built-in oscillator of the Arduino microcontroller, a future update (v2.0 2014-09-23) implemented a battery included circuit to handle time, this to ensure time is measured during shutdown (power failures or manual shutdown), see 3.5 for more information.

2.4 EEPROM

Arduino has a 1kB memory which can be used for saving/loading during runtime, the parameters used for Alarms, PWM values etc. are automatically saved when changed through LCD during runtime as well as automatically loaded at startup of microcontroller.

2.6 Temperature sensor

The temperature is measured and software filtered to make sure that a more stable measurement is presented at the LCD.

2.5 Optimizations

Not needed yet, so not done yet!

3 Hardware

3.1 Circuit board layout

Putting the transistors at the end of the board, making it possible to fastened them to the heatsink predrilled holes presented itself as a wonderful idea, what I didn't know at first was that to simplify construction the manufacturers design most TO220 transistors such as their metal casing is connected to the collector pin hence all collector pins would be connected together i.e. not such a great idea. The fastened transistors can be seen in fig. 8.

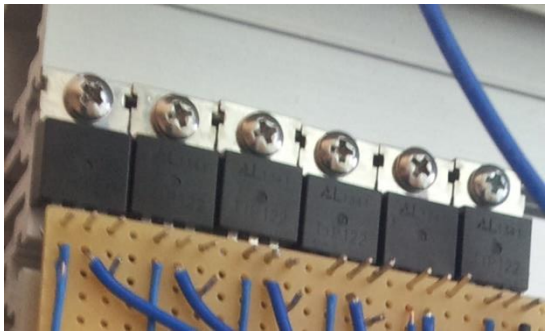

Fig. 8. Transistors fastened to heatsink.

Fortunately the heatsink came anodized so the surface does not conduct, though the fastening of the screws should be done with care since only a small scratch would make the transistor electrically connected to the heatsink (nylon screws or Mica shields would be desirable, though I'm only a poor student so the screws which followed the heatsink had to suffice). Make sure to measure after fastening the transistors that they are not electrically connected to the heatsink.

Ceramic resistors and Zener diodes are put on the bottom side connected to the heatsink to reduce footprint of circuit board and make sure they receive heat dissipation to the heatsink. This can be seen in fig. 9.

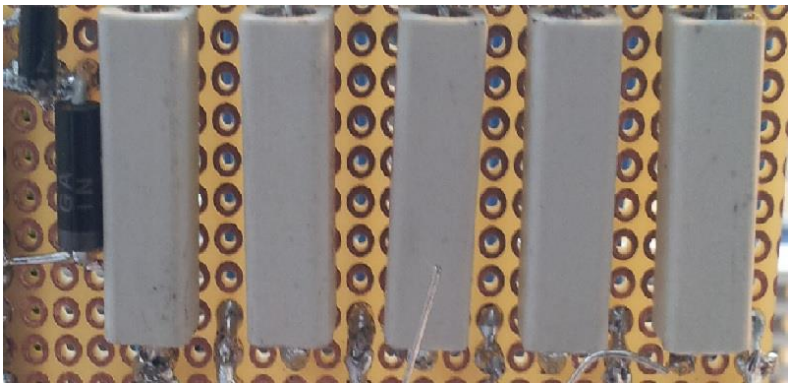

Fig. 9. Resistors and Diodes at bottom side of circuit board.

3.2 Cables

For easy soldering solid core cables are used for the signal cables, a tip for finding cheap AWG 24 solid core cables is to dismantle a CAT 5 cable which gives 8 solid core cables of different colors.

For low voltage power multicore cables was used (no red and black color in that CAT cable!), RK 0.5mm^2 cables used for Main power and ground from power adapter.

Cables going from circuit board to LCD, button and potentiometer are multicore for easier assembly.

3.3 Button and Potentiometer

A simple pushbutton is connected to one of the analog inputs of the Arduino, fastened in the box outside of the heatsink near the LCD, this applies for the 10kOhm potentiometer as well. An internal pull-up resistor (20k-50k) is used for the button enabled by software.

3.4 LCD

LCD is connected according to the circuit diagram (see attachment), if no input has been done the LCD will be shutdown after 30sec through the Arduino analog input. A 220ohm resistor makes sure that the correct level of backlight brightness is used.

3.5 Temperature Sensor

A temperature sensor (TMP36) is used to keep track of the temperature within the heatsink compartment (The DS3231 includes a built-in temperature sensor as well which can be used as reference).

3.6 DS3231 RTC

Early designs and simulations included a battery backup for the Arduino as well as a low-voltage shutdown for the power supply, since the power adapter included inbuilt capacitors and problems occurred with transistor current leakage these ideas were not used, instead a DS3231 RTC circuit has been ordered (connector pre-soldered) to keep track of the time instead (which was the whole point with the battery backup anyway). The module is a no name brand bought at eBay originally used for Raspberry Pi. As of v2.0 (2014-09-23) this circuit has been successfully implemented, a 3V CR927 coin battery ensures a estimated ~10-15years lifetime.

Fig. 10. DS3231 module.

The DS3231 needs to be connected to the SDA(A4) and the SCL(A5) on the micro controller, these have been pre-soldered together with +5v and GND for future connection.

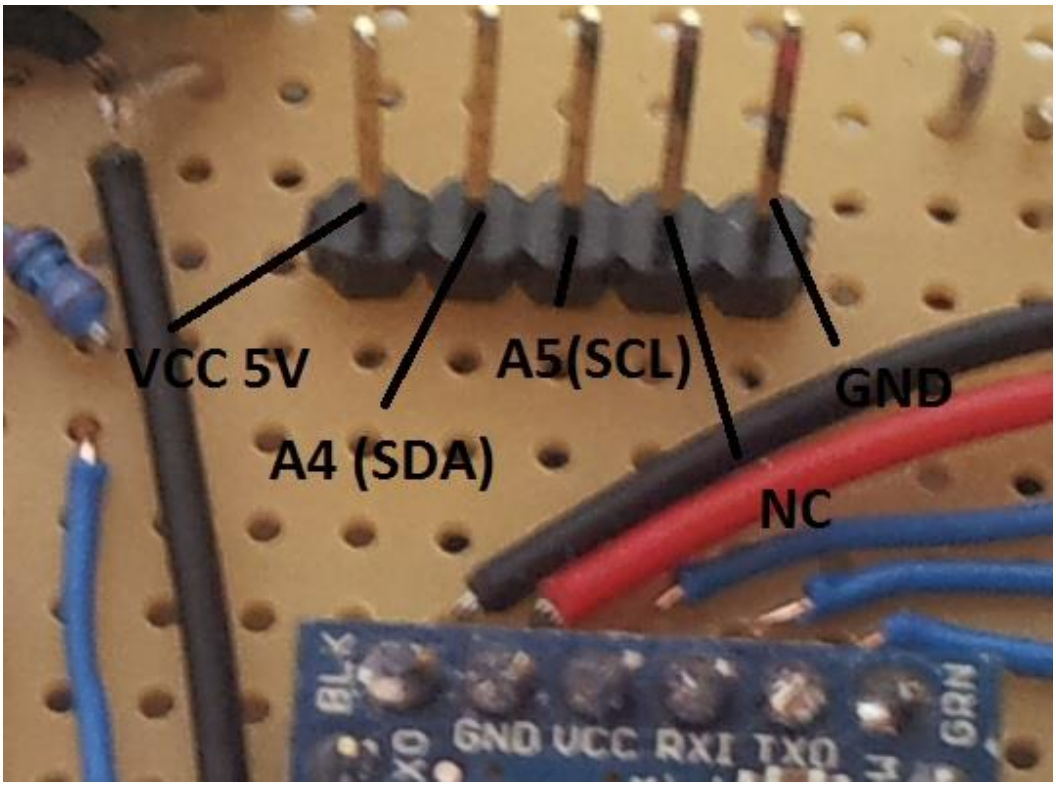

Fig. 11. Pin out for DS3231.

3.7 DC jacket and Rocker switch

A DC jack 2.5/5.5 mm is used to connect the power adapter to the circuit (fastened to the heatsink side), the power cable is connected to the Rocker switch to be able to turn off the fixture manually, ground is connected to the circuit directly.

Fig. 12. DC jacket and Rocker switch.

3.8 Heatsink

Heatsink used is the Makers led heatsink 6''[4], the circuit is fastened in parallel with the Lumia 5.2 to ensure components needed are connected to the heatsink and to reduce the size of the fixture, holes are drilled for power connectors and cables as well as for the screws holding the box with the LCD, button and potentiometer. See the following figures for more info:

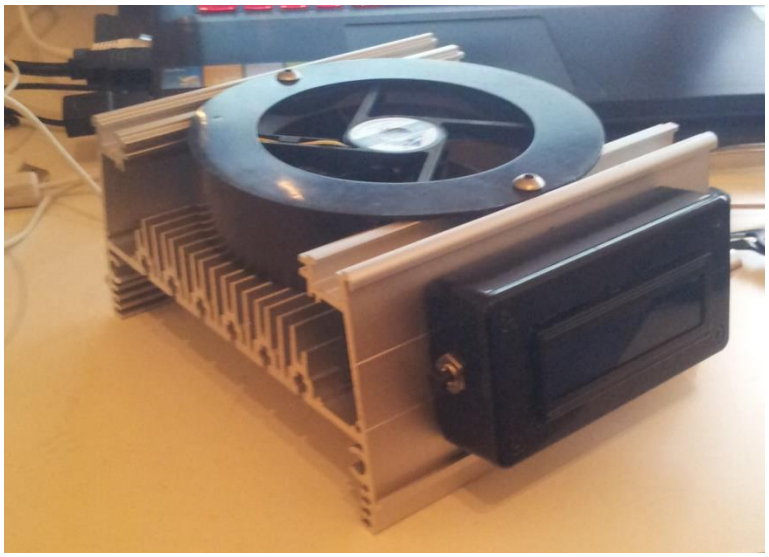

Fig. 13. Makers led heatsink Topside.

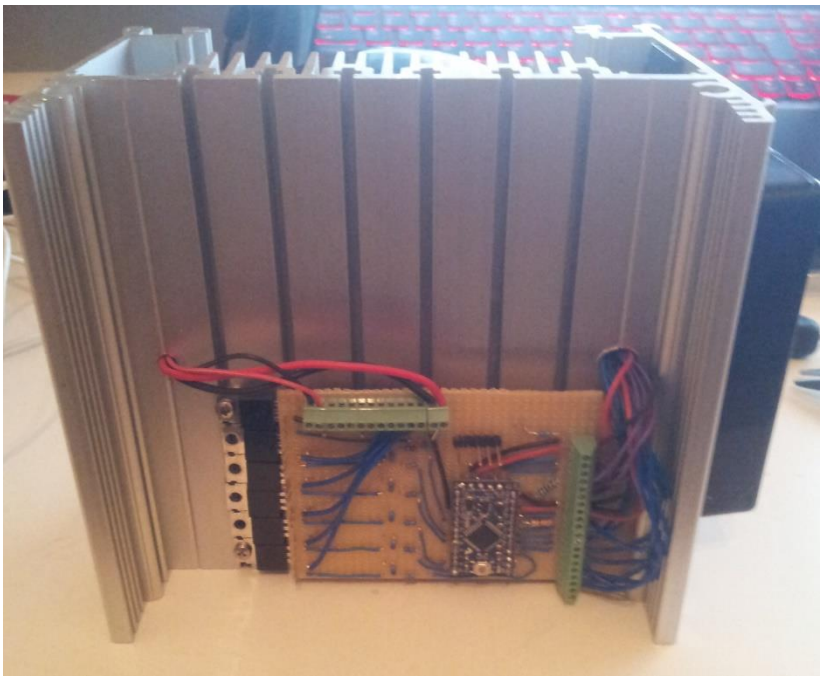

Fig. 14. Makers led Heatsink Bottom side.

3.9 Suspension

Since I didn't want any metallic links hanging from the ceiling two rust-free cantilevers were bought, these were cut to fit the heatsink and two more holes were drilled for the backside of the heatsink.

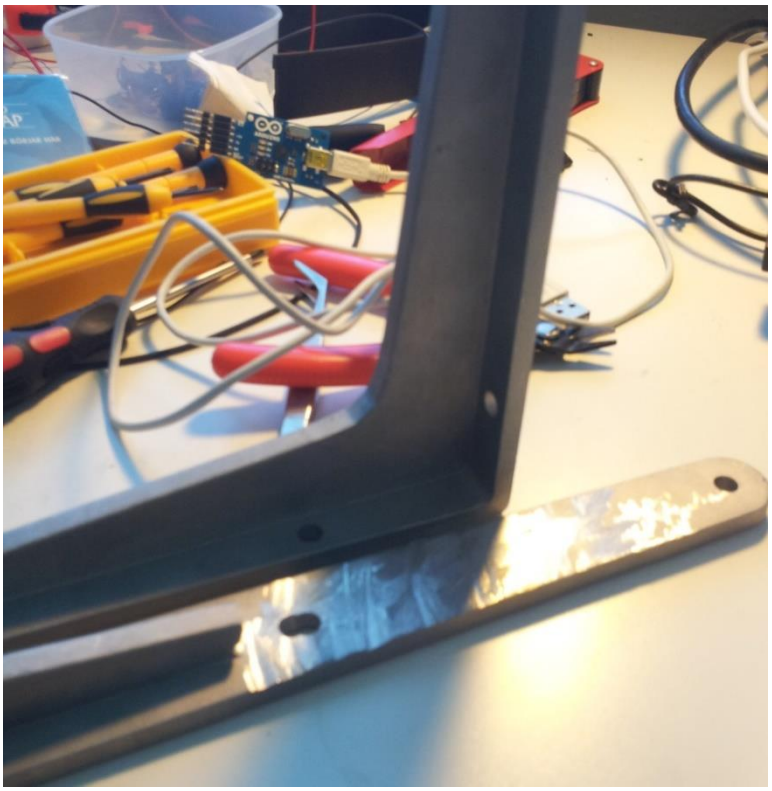

Fig. 15. Drilled and cut.

4 Testing

4.1 Cheap DIY oscilloscope

A oscilloscope could be needed if u want a more precise measurement of the PWM signal since most multimeters only show a "average value" of the voltage(since its turning on and off). Luckily I had an Arduino Uno I've used for prototyping which could be converted into a "cheap" oscilloscope to be able to measure the correct waveforms. There are many available but the one I found to be working somewhat correctly and easy to setup was the freeware version Arduinoscope , it can be found at :

http://www.instructables.com/id/Arduino-Multi-Channel-Oscilloscope-Poor-Mans-O/#step0

4.2 Measurements

After assembly of the circuit, cables and the Led chip was made I made some measurements to verify that the Leds were receiving the correct current, and to see what PWM percentage to use based on the recommended currents for each channel, the following table shows the values measured:

Channel	PWM %	V_E	I_C~ I_E	I_{REC}
1	100	0.42 V	750 mA	700 mA
1	90	0.46 V	710 mA	-
2	100	0.60 V	1071 mA	1000 mA
2	90	0.56 V	1000 mA	-
3	100	0.58 V	1036 mA	1000 mA
3	90	0.53 V	950 mA	-
4	100	0.45 V	786 mA	700 mA
4	90	0.39 V	696 mA	-
5	100	0.38 V	678 mA	700 mA
5	90	0.34 V	610 mA	-

Table 2. Measurements.

4.3 Fan Control and PWM Frequency adjustment

During testing of fan it was very quickly noticed that the brushless DC fan provided with the heatsink did not enjoy getting controlled by PWM, since the fan is repeatedly turned off and on its losing its torque and hence "kicking" when starting. The behavior of the kicking introduces an annoying buzzing sound when using a PWM value below 95%, probably in the same frequency as the frequency provided by PWM output D11 on the Arduino board (ca 500Hz). One solution would be to connect a small ceramic capacitor around 0.1uF value from base of transistor to ground which would slow down the slew rate of the PWM switching i.e. smoothing the PWM signal into a more like DC signal. This would of course imply that the transistor is always on and consume more power and decrease the lifetime of the components.

Another solution is to increase the frequency of the PWM signals such as the noise is moved to the non-human audible spectrum i.e. over 20kHz, probably not recommended if dogs are present(though increasing the frequency could hopefully remove the phenomenon altogether). Make sure components and voltage supply can handle the frequency though (Check Small-signal Current Gain chart of transistor).

The Arduino has 3 different timers, where the PWM D11 is handled by timer 2 (timer 0, 1 and 2). Beware that changing Timer 0 can affect other functions in the microcontroller, Millis() for example is using timer 0 so time would be affected. The Frequency was in this case changed to ca 30 kHz (this also changed the frequency of LED channel 1 (PWM D3) to the same frequency.

4.4 Extended Testing

Based on the measurements taken in 4.2 the following PWM values was chosen for the testing (will be used when running the chip at 100% as well):

Channel	PWM %
1	90
2	90
3	95
4	90
5	100

Table 3. PWM levels for testing.

Measurements were taken at a ambient temperature of ~23.5 Celsius and with stock fan provided with the Makers Led Heatsink, measurements are provided by the TMP36 temperature sensor on the circuit board which is in contact with the heatsink surface.

Time(min)	Fan speed %	Temperature measured
00	80	23.5
15	80	64.0
30	80	66.0
45	80	67.0
60	80	67.2
75	95	66.2
90	95	65.7

Table 4. Temperature testing, stock fan.

According to the datasheet of the Lumia 5.2[1] the maximum operating temperature(surface) is 80 degrees Celsius, the measured temperature is below that temperature but keeping it as low as possible will allow the maximum lifetime and brightness of the LED chip so a new better fan was purchased (Noctua NF-B9). The stock fan was also pretty noisy at higher speeds.

The same PWM levels were used for the newly acquired fan which provided the following measurements:

Time(min)	Fan speed %	Temperature measured
00	80	23.2
15	80	57.9
30	80	60.3
45	80	61.8
60	80	61.8
75	95	61.1
90	95	61.3

Table 5. Temperature testing, Noctua NF-B9.

Another important aspect is that of the transistor casing temperature, at higher temperatures the transistor will provide a different gain so the current could be slightly off. A final measurement of each channel (at same PWM level) was done at the maximum reachable temperature:

Channel	PWM %	V_E	I_C~ I_E@61.3°C	I_{REC}
1	90	0.50 V	893 mA	700 mA
2	90	0.65 V	1160 mA	1000 mA
3	95	0.65 V	1160 mA	1000 mA
4	90	0.47 V	839 mA	700 mA
5	100	0.45 V	804 mA	700 mA

Table 6. Transistor temperature drift, current measurements.

An increase in transistor case temperature will increase its current gain (beta, in general around 0.7% per degree centigrade above room temperature[5]) hence if the temperature drift too much the current will not be as calculated. By decreasing all PWM levels by 10% the temperature decreased to 56.0 degrees Celsius and the following voltages could be measured:

Channel	PWM %	V_E	I_C~ I_E @56.0°C	I_{REC}
1	80	0.39 V	696 mA	700 mA
2	80	0.54 V	964 mA	1000 mA
3	85	0.53 V	946 mA	1000 mA
4	80	0.39 V	696 mA	700 mA
5	90	0.38 V	679 mA	700 mA

Table 7. Transistor temperature drift, current measurements.

The total power consumption of the whole fixture was measured by a simple energy meter connected before the power adapter, the following values could be measured:

PWM lvls according to:	W
Table 6	106
Table 7	91

Table 8. Power Consumption

4.5 Finalization

The complete fixture can be seen in the following pictures.

Fig. *16. Fixture side/front.*

Fig. 17. Fixture backside.

Fig. 18. Fixture Bottom.

Fig. 19. Front fixture installed.

*Fig. 20. Bottom fixture installed.**

* Picture taken before DS3231 RTC installed (the 5 pins in the middle of circuit).

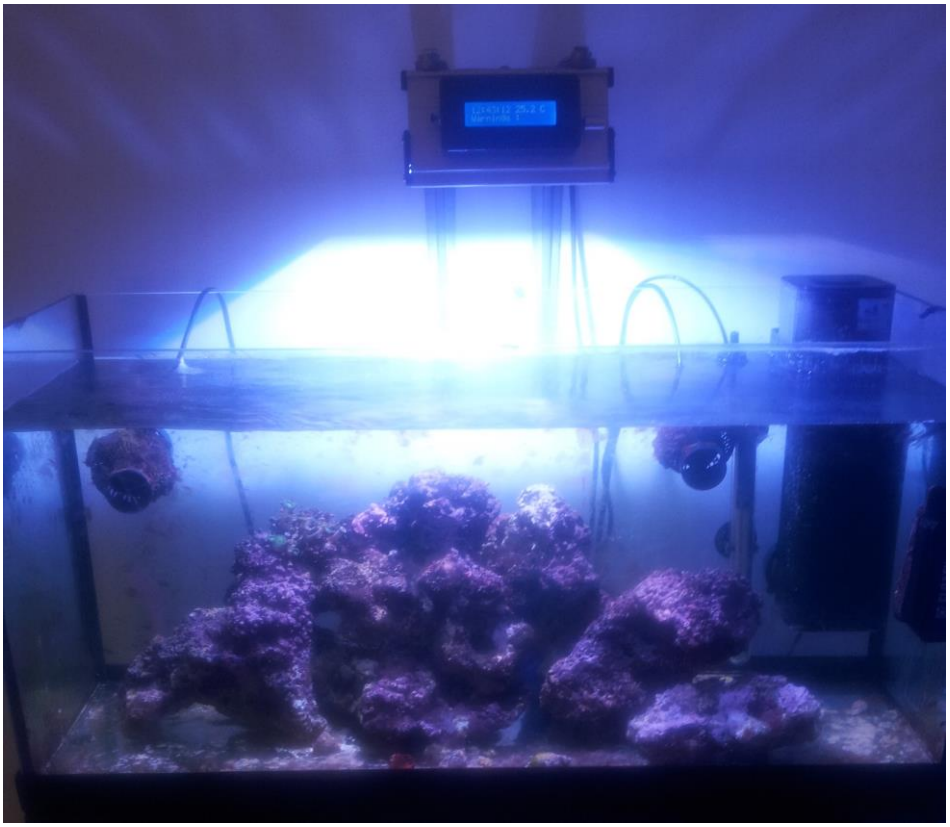

Fig. 21. Front fixture installed and On.

4.6 LED Channel schedule

Time schedule for the channels needs to be tested, but royal blues will probably run from 09-21 (CH2/4/5) and Channel 3 i.e. the White LEDS will run from 11-17 (full from 12-15 maybe).This requires more research and testing (need to purchase more corals as well).

5 Summary

Since no suitable option was found a LED lighting fixture was planned, built and tested for the purpose of growing some corals and gaining some experience in electrical simulation, construction and testing as well as testing the newly acquired knowledge as a Masters student in electrical engineering.

The final product works as expected, though much could be improved in both terms of hardware and software. Little knowledge of suitable components was known beforehand so better suited hardware could probably be found, controlling microcontrollers in software is fun and there is plenty of possibilities to be creative but there is still much to be learned.

Some design layouts could have been given more thought, for instance it was not known beforehand that to increase manufacturing efficiency the collector and the heatsink of transistors are often connected, thus introducing some problems connecting all transistors to same heatsink. Even anodized heatsinks are easy to scratch and thus u would have collector connected transistors which is probably not a good idea. The probably fake PCB connecters are cheap in comparison to the “original” ones, but it was noticed that the lesser quality is noteworthy, forcing some cables to be

soldered instead of screwed into the connectors. In the future crimped connectors will probably be favored.

Nonetheless the experience gained about transistor use and its properties, heat dissipation of components and the choice of suitable components has been valuable. Electrical construction and design are rare in the courses given at school so any practical work is educational.

Apart from being the first non-forced report written (writing a report of your work is good standard!) it provides both a manual for future upgrades and an option for the aquarium lighting community.

6 Acknowledgements

I would like to give a special thanks to Roland Timgren for tips about circuit designs and tools used for the hardware construction. Also Rune Rugland for some leftover screws!

7 References

[1] Cree Lumia 5.2;
http://www.ledgroupbuy.com/cree-lumia-5-2-70w-full-spectrum-5-channel-cree-led/
2014-09-05

[2] Arduino Pro Micro 5V / 16MHz ; http://arduino-board.com/boards/arduino-pro-mini
2014-09-05

[3] PWM – Pulse-Width Modulation ; http://en.wikipedia.org/wiki/Pulse-width_modulation
2014-09-05

[4] Makers Led Heatsink; http://www.makersled.com/
2014-09-05

[5] Transistor current gain;http://www.answers.com/Q/Does_transistor_current_gain_increase_or_decrease_with_temperature
2014-09-08