1 Introduction

I started with a 10HP 2 stroke engine which was noisy, smelly and asked quite some maintenance. So I decided to try an electric trolling motor. I quickly discovered that the trolling motor I had, uses a very inefficient resistor to control the speed. I also discovered that the resistors (speed coils) are not very durable. I burned the speed coils of two trolling motors I bought from eBay. They do not seem to last if used for several hours as I did. My brand new 105AH battery also died in the first season (luckily under warranty) which might have been caused by too deep discharge. So I was not satisfied with the motor. Time to improve things. This project!

This controller I build lowers current consumption for the trolling motor and gives it a continuous speed control. Cheap trolling motors have 3 or 5 speed control settings. The principle of lowering the speed is to put a resistor in series with motor, the infamous speedcoil. This resistor gets very hot and is water cooled (part of the motor under water). If the resistor burns it makes the motor fail.

This project let's the motor operate at it's highest speed setting (no resistors in series) and controls the voltage going to the motor using Pulse Width Modulation (PWM). PWM is very efficient (the controller gets hand warm only) and can deliver any voltage between minimum (0V or off) up to the maximum voltage of the battery. The digital implementation uses steps from 0 to 255, which gives the impression of a continuous control range.

This instruction has been build for those interested to build this project. You should use this controller if you want to improve efficiency. If you always use your motor in the highest setting, this controller does not help you. Only the lower speed settings give you a much better range. Expect a double range when using setting 3 (out of 5). And your battery will last longer.

2 The tools

The circuit has been designed using open source tools only.

Please visit http://www.geda-project.org for more details on the used tools:

- gschem for schematic entry
- gnetlist to create a netlist for spice simulation
- ngspice (http://ngspice.sourceforge.net) for simulation
- PCB to create gerber PCB file

I use Linux as my desktop so don't know how these tools work with Windows.

2.1 Schematics

Gschem is used for schematic editing:

```
sudo dnf install geda-gschem geda-gnetlist pcb geda-utils
```

Create a gafrc file containing:

```
(component-library "/home/thba/spice/symbols")
```

2.2 Simulation

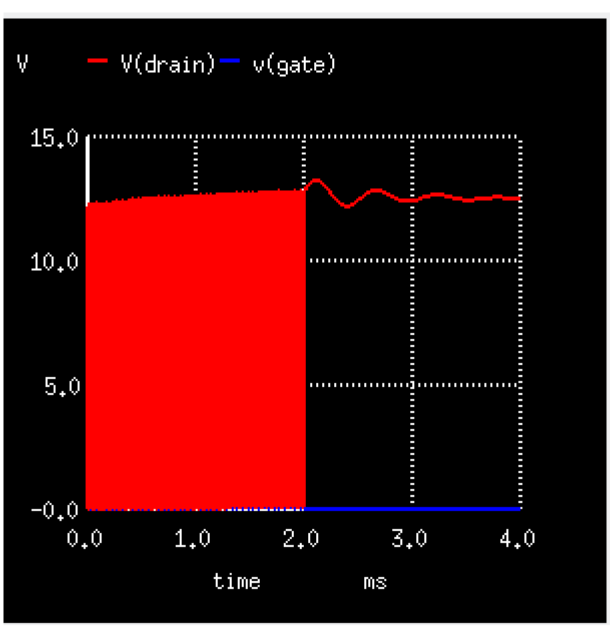
Optionally if you want to simulate, create a spice netlist for simulation:

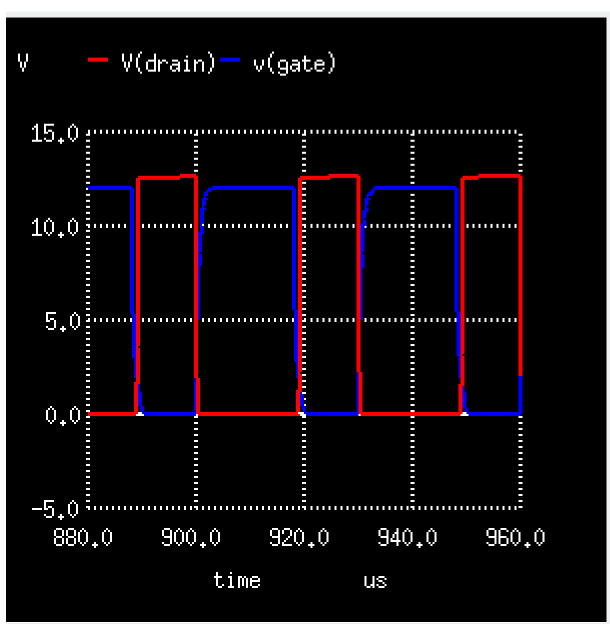
```
gnetlist -g spice-sdb -o motor_control.net motor_control.sch
```

```
ngspice motor_control.net
******
** ngspice-26 : Circuit level simulation program
** The U. C. Berkeley CAD Group
** Copyright 1985-1994, Regents of the University of California.
** Please get your ngspice manual from
http://ngspice.sourceforge.net/docs.html
** Please file your bug-reports at
http://ngspice.sourceforge.net/bugrep.html
** Creation Date: Sat Oct 10 04:33:45 UTC 2015
******

Circuit: * gnetlist -g spice-sdb -o motor_control.net
motor_control.sch

ngspice 17 -> run
Doing analysis at TEMP = 27.000000 and TNOM = 27.000000
ngspice 18 -> plot V(drain)
```





I'm not going to explain how to use ngspice is much more detail. That's a project on it's own. If you are interested in simulation you will probably already know how to do that.

2.3 PCB

Start the pcb tool after following the tutorial here:

http://www.delorie.com/pcb/docs/gs/gs.html#Your-First-Board

Any schematic updates can be re-applied to the pcb layout using this command:

```
gsch2pcb motor_control.prj
```

Finally export to png photo quality to judge the silk and solder masks before exporting to gerber.

Also use gerbv to review the final gerber files.

3 Design

The motor regulator is a classical pulse width regulated power supply (PWM). The motor is switched on/off at a fast 20kHz rate. When the MOS switch is on the loss is low due to the use of high quality mosfets. The channel resistance is less than 2mOhm and two instances are put in parallel. When the MOS switch is off, no current flows from the battery, but the motor current continues via a flyback diode. Both NMOS and flyback diode consume a few Watts and are therefor placed on a small heatsink. If the duty cycle is 100% (the NMOS is always on) the maximum allowed current depends on the amount of heat that can be dissipated. At 50A, the maximum for my trolling motor, the dissipation will be $50A^2$*2mOhm=5Watt. That's not a lot, but a suitable heatsink is needed. Also note that screw terminals can also become quite hot at 50A current. Do buy properly rated connectors and wire.

The circuit consist of two parts. A controller (Arduino) and a custom PCB with the driver circuitry.

3.1 Schematic diagram driver circuitry

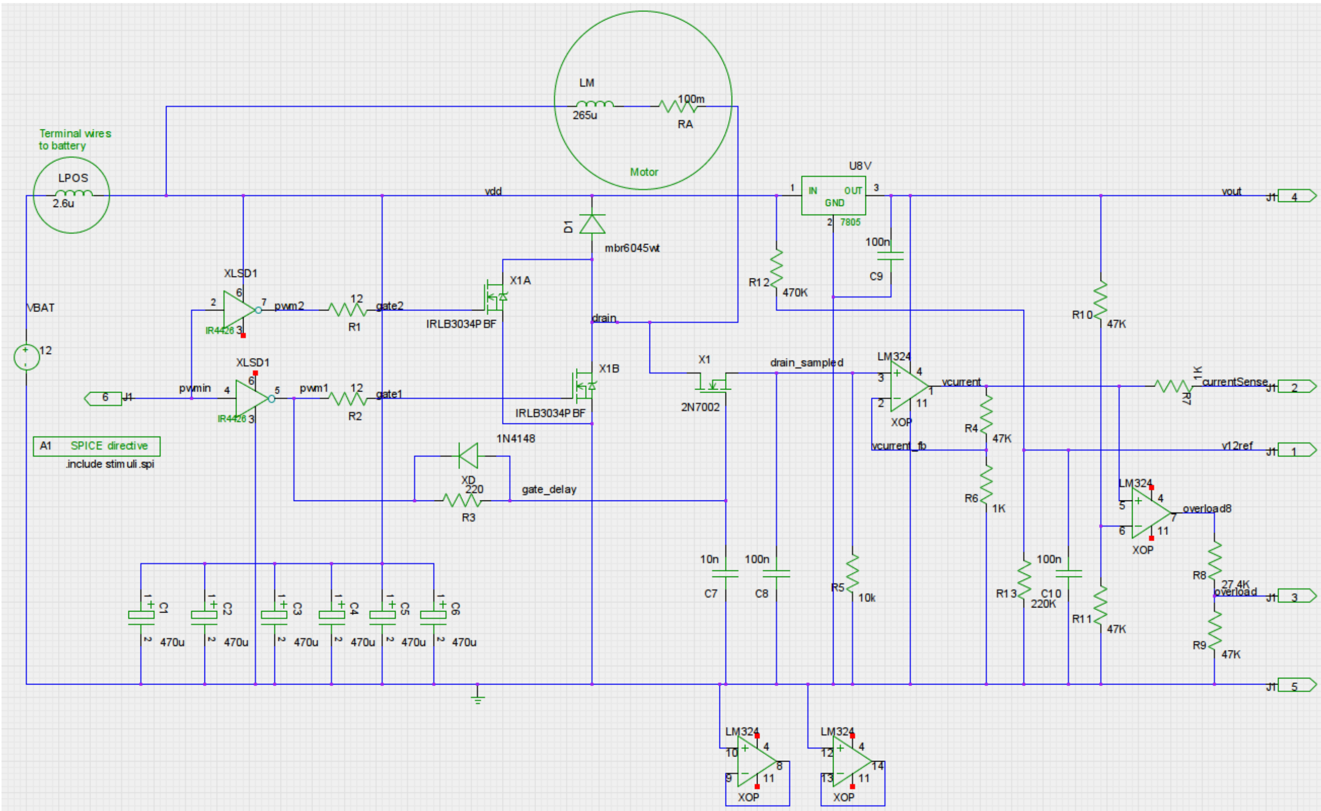


Illustration 1: Schematic diagram of motor controller

The Arduino connects to J6.1 to drive a mos low side driver XLSD1 (IR4427). The mos driver is current limited by a resistor R1 before it drives the gate of the NMOS X1A (IRLB3034PBF). To further lower the on resistance, the driver and NMOS have been doubled.

The motor is directly connected to the positive terminal of the battery. The negative lead of the motor is connected to the negative battery terminal via the NMOS switch. NMOS is chosen as they are better suited for driving high currents.

When the NMOS switches off, the current will continue to flow due to the large self inductance of the motor. The Schottky diode D1 (MBR6045WT) makes sure this is possible.

The final very important component is the capacitor placed over the 12V battery supply, C1-C6. These capacitors will deliver the switching current (peak current 50A). These caps need to be of very good quality. A single cap cannot handle the current, so 6 have been placed in parallel. The Equivalent Series Resistances (ESR) of these caps is the parameter to take into account as ESR creates heat and determines the RMS ripple current they can handle (I used 6 Samwha 470uF/25V WB1E477M10016PA).

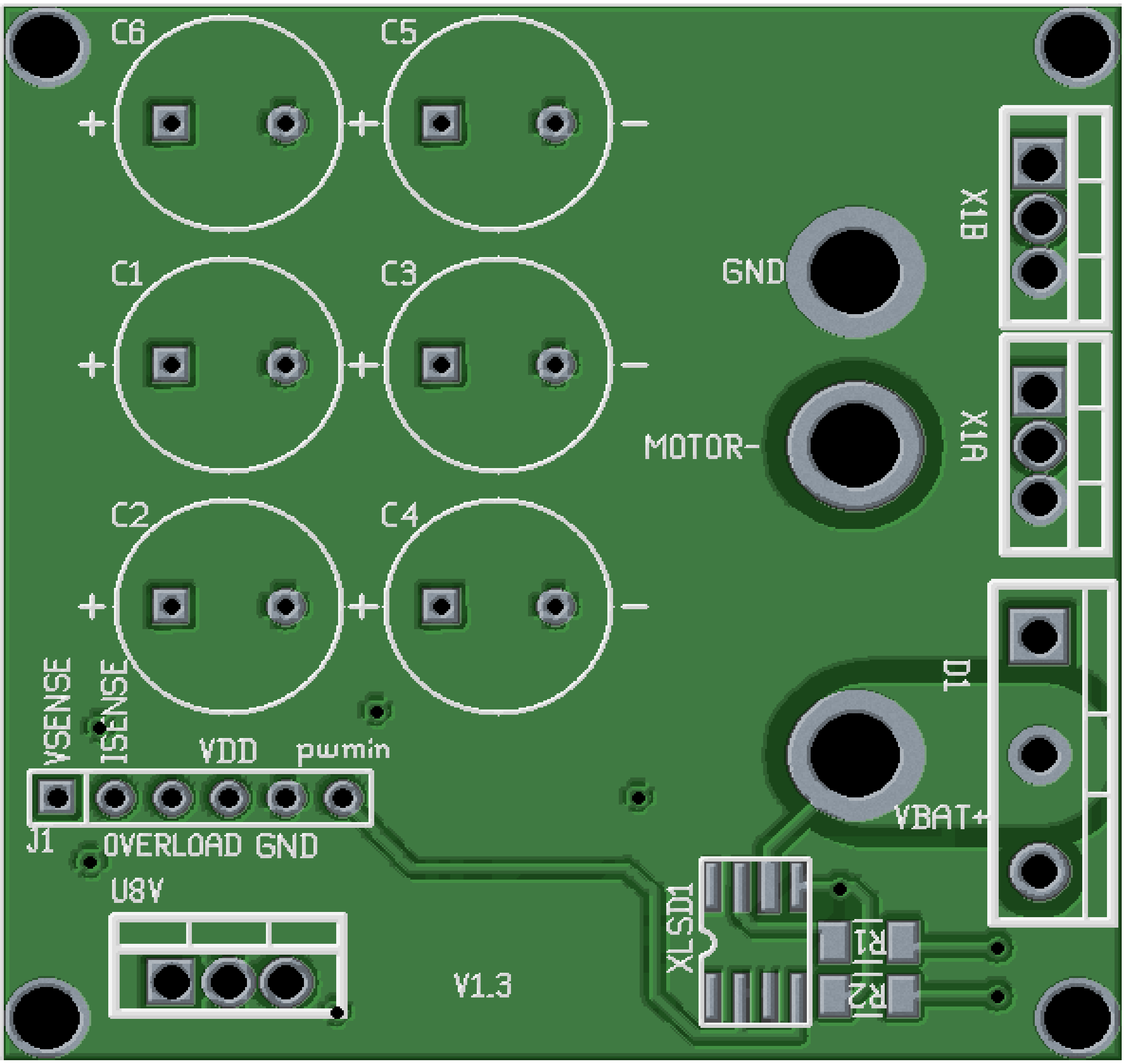
U8V (7808) makes the 8V supply voltage for the Arduino board.

The rest of the components is used to monitor the voltage and current. This information is fed to the Arduino board on terminal J1.1, J1.2 and J1.3. The Arduino software monitors these values to calculate battery state of charge (voltage) or used capacity (current).

X1, R3 and C8 form a sample an hold circuitry. The drain_sampled node contains the DC voltage over the drain which is a measure for the current drawn. The voltage is first multiplied by 48 by an opamp XOP (LM324). The ADC in the Arduino now gets a voltage of $I_{motor}*R_{ds}*48$~100mV/A. This value is very inaccurate and needs to be calibrated using a known load/current. Calibration and offset correction is done in the controller software, as that's much easier than adding a variable resistor.

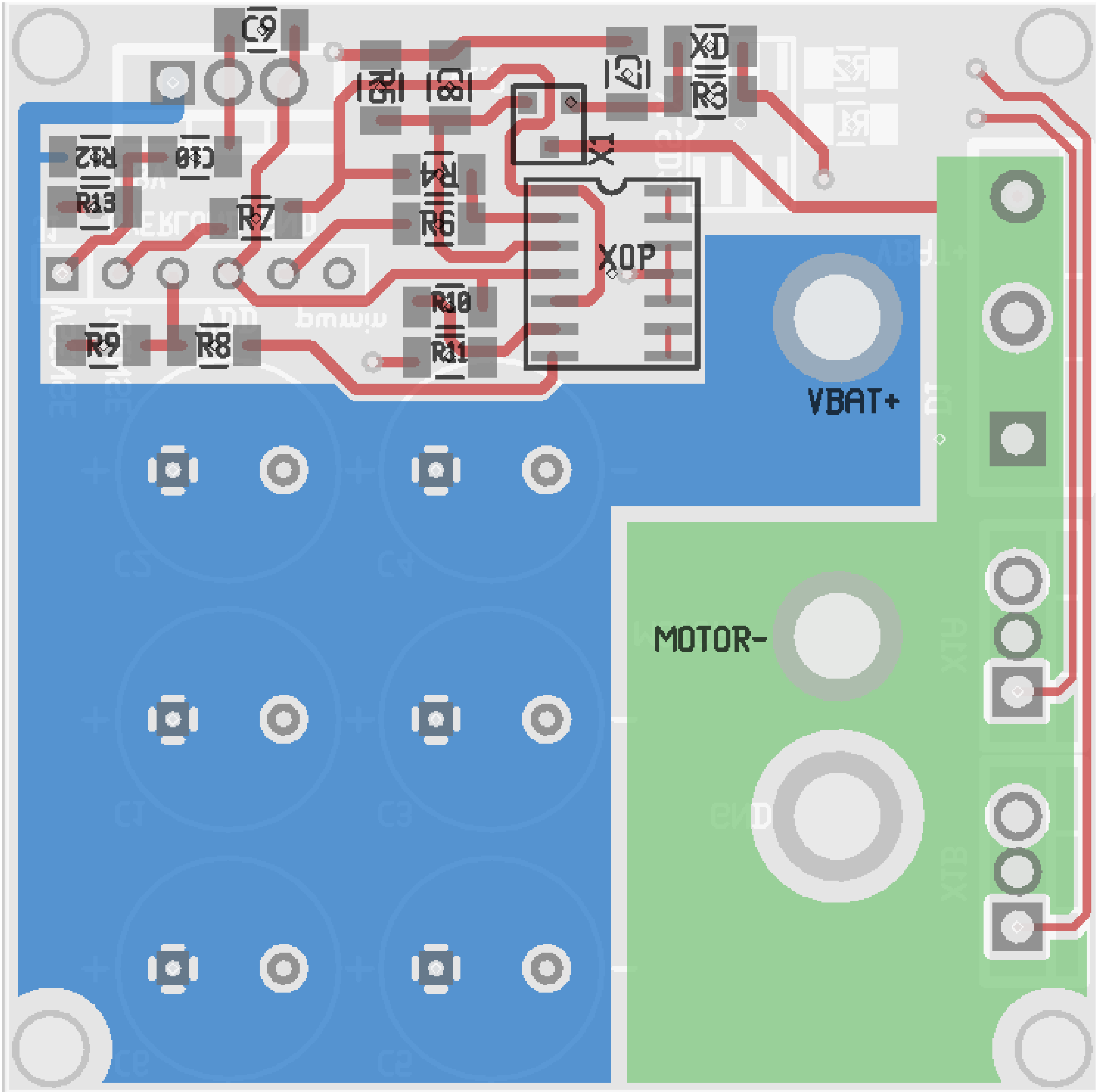
3.2 Printed Circuit Board (PCB)

PCB has been made using “PCB”



Drawing 1: Top side PCB

And the bottom:



Drawing 2: Bottom side PCB

Size of PCB is only 5x5cm, so manufacturing is cheap.

3.3 Parts list

REFDES	device	value	footprint
C1-6	POLARIZED_CAPACITOR	470u WB1E477M10	RCY200P
C7	CAPACITOR	10n	1206
C8	CAPACITOR	100n	1206
C9	CAPACITOR	100n	1206
C10	CAPACITOR	100n	1206
D1	DIODE	MBR6045WT	TO247
J1	OUTPUT		JUMPER6
R1	RESISTOR	12	1206
R2	RESISTOR	12	1206
R3	RESISTOR	220	1206
R4	RESISTOR	47K	1206
R5	RESISTOR	10k	1206
R6	RESISTOR	1K	1206
R7	RESISTOR	1K	1206
R8	RESISTOR	27.4K	1206
R9	RESISTOR	47K	1206
R10	RESISTOR	47K	1206
R11	RESISTOR	47K	1206
R12	RESISTOR	470K	1206
R13	RESISTOR	220K	1206
U8V	7808	8V Link	TO220W
X1	MOSFET	2N7002	SOT23
X1A	MOSFET	IRLB3034PBF	TO220W
X1B	MOSFET	IRLB3034PBF	TO220W
XD	DIODE	1N4148	1206
XLSD1	DUAL low side driver	IR4227	SO8
XOP	QUAD_OPAMP	LM324	SO14

3.4 Arduino

For this project I choose to use an Arduino nano.

Arduino Nano Front   Arduino Nano Rear

Drawing 3: Smallest Arduino available

The display is a standard 1602 LCD. Just search eBay for 1602 lcd. You can choose a green or blue backlight version.

For user input I only added a single potmeter to control the speed and switch the motor off. Simply because it's easy to make. Just drill a hole in the cover plate and fix the potmeter with the nut. Switches (for up/down/on/off to implement a menu etc.) are more difficult to mount, especially if they also need to be waterproof. The potmeter on it's minimum value means off and displays the battery status and the remaining battery capacity. When you turn the potmeter to the right, the motor start to run and the display shows the current and voltage levels. The battery voltage cannot be used to estimate the capacity when the motor is running so those values are only shown when the motor is not running.

Programming the board is easy using a standard USB cable and freely available Arduino programmer software.

The port connections are not that important and can be configured in the software. I choose these ports:

```
const int vsensePin = A0;
const int isensePin = A1;
const int backlightPin = 3;
const int overloadPin = 11;
const int pwmoutPin = 10;
const int PotpPin = A2; // A2 Potmeter plus
const int PotcPin = A3; // A3 Potmeter center
const int PotnPin = A4; // A4 Potmeter min
```

Some features built-in:

- max current regulation. If the current exceeds the maximum (IMAX) the voltage is lowered.
- max current protection. If the output is shorted (or motor blocked) the motor is disabled within a ms (for one second). It's still wise to add a properly sized fuse.
- If the battery voltage drops below 10.5V, the current is lowered to prevent too deep discharge. This means maximum speed drops gradually when battery becomes empty. But efficiency goes up, so you will still get home!
- Display with battery icon, voltages and currents of battery and motor when motor is running,

Illustration 2: Display when motor is running

runtime, battery percentage left and used capacity in AH when idle.

Illustration 3: Display when idle

3.5 Housing

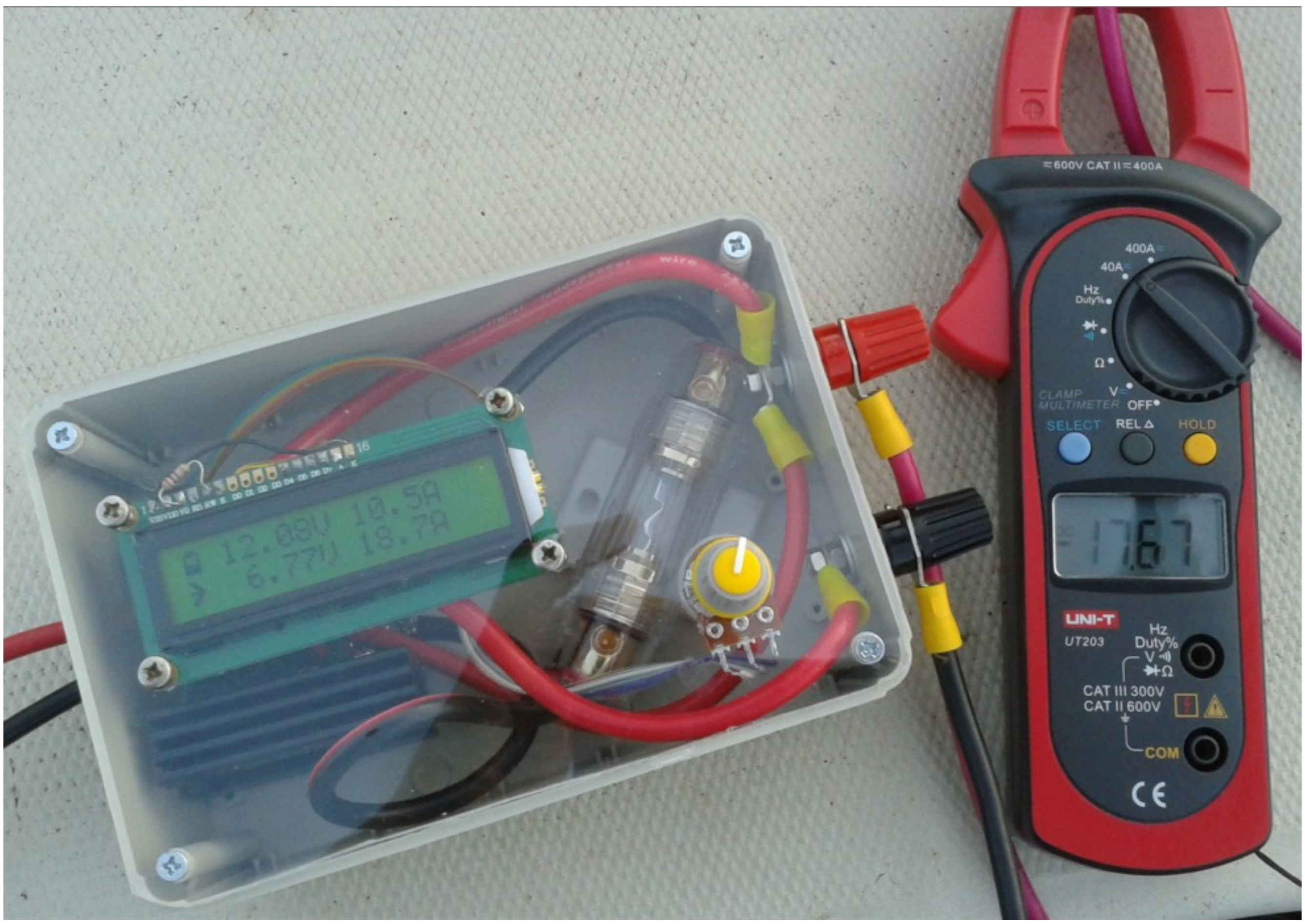


Illustration 4: Plastic box with policarbonate cover

Cables should be capable of carrying 50A. I choose $6mm^2$.

The fuse on the photo was rated 50A and I got it from ebay. The fuse housing burned within one season so I replaced it with a more expensive part from my local car parts dealer. Stuff that can carry these currents is just not cheap. The screw terminals are from Hirschmann and rated 35A. Not enough, I know and indeed they become hot. I bought them as they only cost 3 euro each. I wanted to connect the existing motor wires with a screw connection to the controller box.

The original cover of the box is replaced by a polycarbonate plate cut in the correct shape. The original cover was not transparent and I did not want to make a big hole in it for the display. The controller should withstand some water (rain?)

Calibration is not needed, but current accuracy will be low. Best is to use a DC current clamp and change these software parameters:

```
const float VCAL = 15.47;  // vbat=vsenseVal/1023*VCAL; VCAL=
5V/220K*(220K+470K)=~15.7
const float ICAL = 0.0369; // 37mA/bit
const float IOFFSET = 3;   // Measured offset after curve fitting
```

The Arduino has a 10 bit ADC. VCAL is the value for full scale. Use it to calibrate. ICAL is the current for one ADC step. Again, use it to adjust the full scale range. IOFFSET is the measured offset current. I measured several low and high current values and fitted the displayed values to the measured values. But it also depends on the temperature of the MOS. The channel resistance increases when the mos becomes hot. So I calibrated after half an hour of normal usage.

4 References

The zip file contains schematic diagrams, gerber files and an Arduino sketch.

Youtube:

Trolling motor with homemade speed control (Maximizer)

Elektrische buitenboordmotor test