Illuminance sensor, part 2: construction and calibration

Oskar Skog

October 20, 2021

Contents

List of Figures

List of Tables

1 Introduction

This is the second part of the project which is covering construction and testing of the prototype.

The execution is divided into three sections

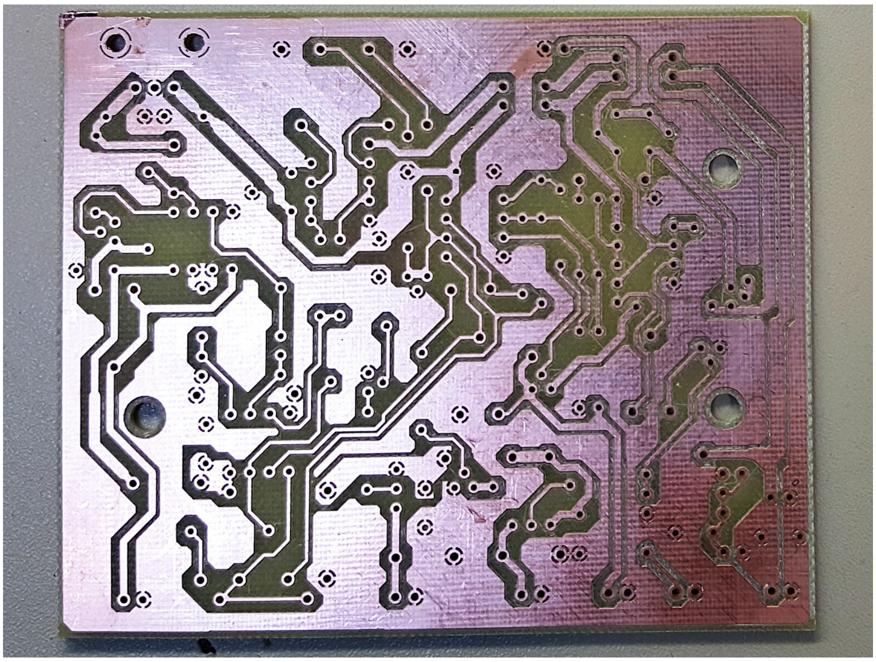
1. Construction of the prototype
2. Testing that it actually works and fixing what needs fixing
3. Calibrating it and testing how well it performs
4. Summary and evaluation

2 Construction

2.1 PCB manufacturing

The PCB was manufactured successfully with the exception of the corner near the heatsink. All other problems are design related

- There is a milling failure on the upper left corner (bottom side), it does not cause any issues however.
- A 0.6 mm drill was substituted for a 0.7 mm drill.
- The holes for the carbon trimmer potentiometers (P1 and P4-P9) were later found to be too tight, but this is probably because of a design error.
- The heatsink had a really tight fit. The excess radius of the holes was apparently insufficient to cover the rounding errors.



(Actual size)

Figure 1: Newly milled PCB

2.2 Cherry picked resistances for fixed dividers

Resistors for the 1:1 voltage dividers were cherry picked even though the tolerances were well within 1% (far better than the expected 5%)

R7	21.98 $k\Omega$	R9	468.8 $k\Omega$
R8	22.01 $k\Omega$	R10	468.4 $k\Omega$
R16	21.98 $k\Omega$	R14	468.3 $k\Omega$
R17	21.93 $k\Omega$	R15	469.5 $k\Omega$
R20	22.00 $k\Omega$	R18	468.5 $k\Omega$
R21	22.02 $k\Omega$	R19	468.7 $k\Omega$

Table 1: Resistance values of all 22 $k\Omega$ and 470 $k\Omega$ resistors

The equations later in this section were not used for cherry picking the resistors, the goal was simply to get R_1/R_2 ratios as close to 1 as possible but doesn't necessarily correspond to what will actually give the lowest possible error.

However, all of this is likely moot as many minor details such as input bias current and input offset voltage etc haven't been considered.

2.2.1 Error in differential amplifier (IC3a)

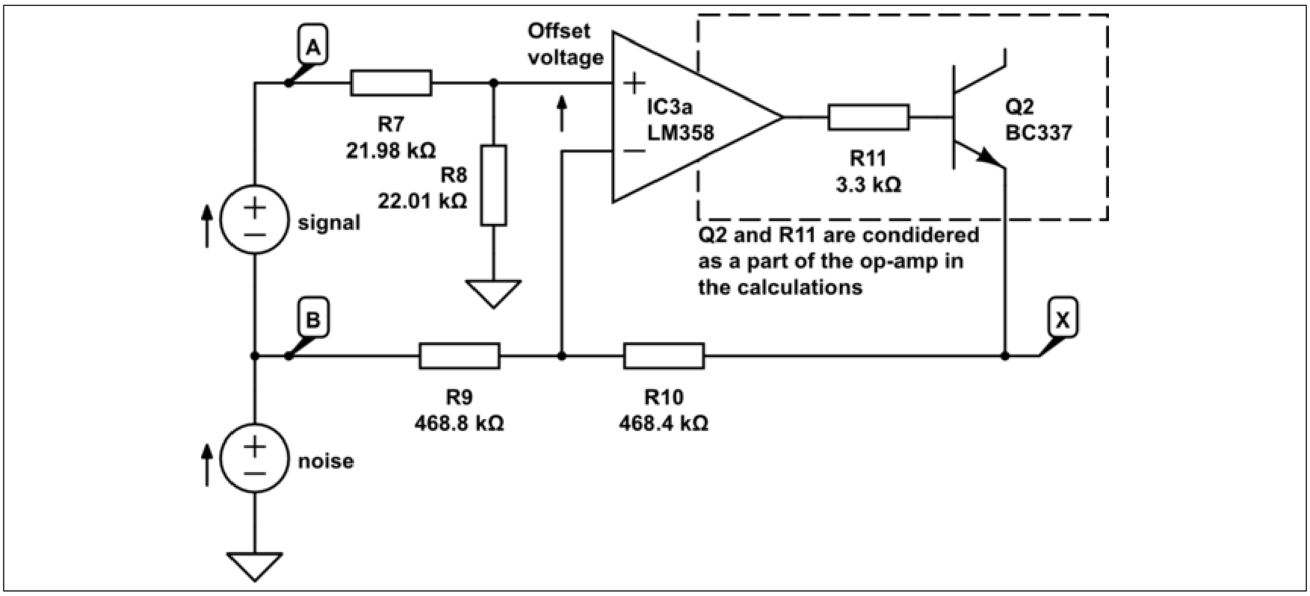


Figure 2: Schematic for the differential amplifier

The circuit is described by the following equation:

$$A\frac{R8}{R7+R8} - \left[(X-B)\frac{R9}{R9+R10} + B\right] = U_{offset}$$

U_{offset} is not just the purely offset voltage but also includes X divided by the gain of the op-amp. This is at worst just a few millivolts.

The equation turned into a function X of A and B:

$$X = \frac{A \cdot R8 \cdot (R9+R10) - B \cdot (R7+R8) \cdot R10 - U_{offset} \cdot (R7+R8) \cdot (R9+R10)}{(R7+R8) \cdot R9}$$

Calculating using the resistance values gives:

$$X = 1.00026 \cdot A - 0.999147 \cdot B - 1.99915 \cdot U_{offset}$$

What is actually wanted is a function X of $A - B$ and B, not the common mode.

$$X = 1.00026(A - B) + 0.001108 \cdot B$$

0.11% of the LDR voltage gets added to the output. This is at worst 2.8 mV and such a tiny error is likely swamped by something else that has been overlooked.

2.2.2 Error in voltage to current conversion

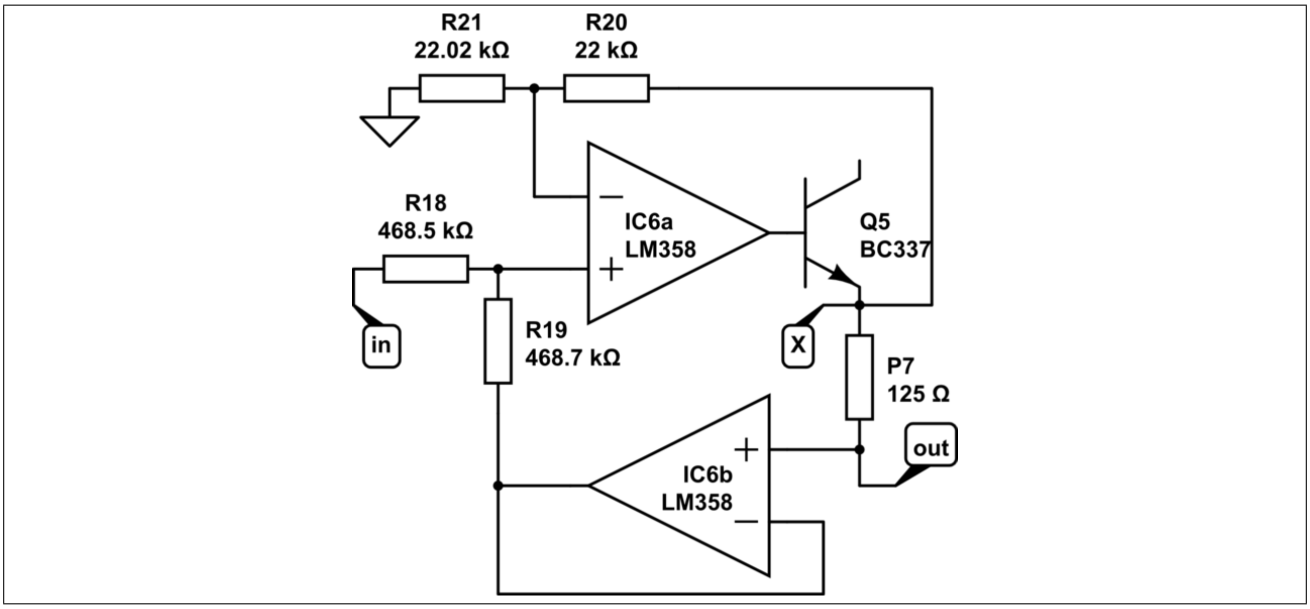


Figure 3: Schematic for the transconductance amplifier

The circuit is described by the following equations:

$$X = \frac{U_{in}R18^{-1} + U_{out}R19^{-1}}{R18^{-1} + R19^{-1}} \cdot \frac{R20 + R21}{R21}$$
$$I_{out} = \frac{X - U_{out}}{P7}$$

With ideal components, the equations would be equivalent to:

$$X = \frac{U_{in} + U_{out}}{2} \cdot 2 = U_{in} + U_{out}$$
$$I_{out} = \frac{U_{in} + U_{out} - U_{out}}{P7} = \frac{U_{in}}{P7}$$

But in reality, the output current is ever so slightly affected by the burden voltage. I as a function of U_{in} and U_{out} is:

$$I_{out} = \frac{U_{in} \cdot R19 \cdot (R20 + R21) + U_{out} \cdot (R18 \cdot R20 - R19 \cdot R21)}{P7 \cdot (R18 + R19) \cdot R21}$$

The error becomes (assuming P7 is 125 Ω)

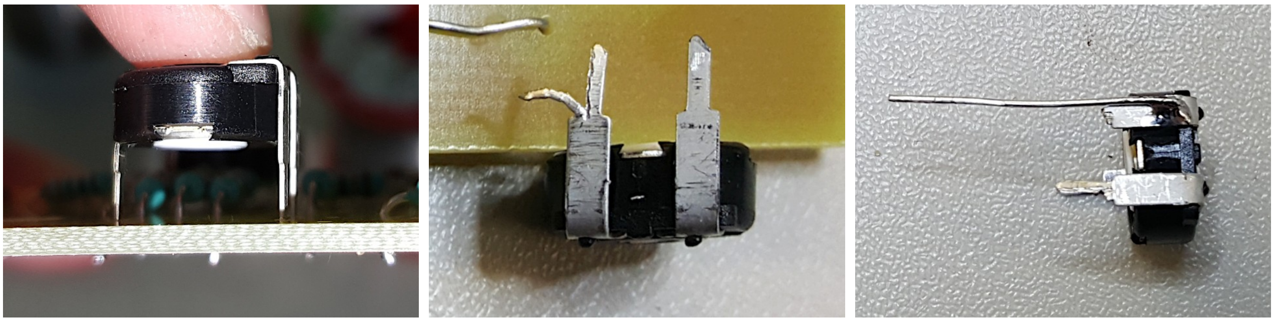
$$\Delta I_{out} = U_{out}\frac{R18 \cdot R20 - R19 \cdot R21}{P7(R18 + R19)R21} = -0.00534\ mA/V$$

2.3 Soldering

2.3.1 Potentiometers not fitting

P1 and P4 to P7 (V10-* and P10-*) did not fit in the holes. The drill diameter in the PCB layout is set to 40 mils (1.0 mm) but the holes should be 1.3 mm.

This has been solved by carefully cutting of one side of each lead. This process has been mostly successful.



On the left: maximum insertion depth. In the middle: modification of the leads. On the right: repair of accidentally snapped of lead

Figure 4: Potentiometer fitment issues

2.3.2 Heatsink

The holes in the PCB for the heatsink were a bit on the small side but the heatsink could be inserted without using unreasonable force. It could be soldered normally using an elevated temperature (near 400 °C) and some patience.

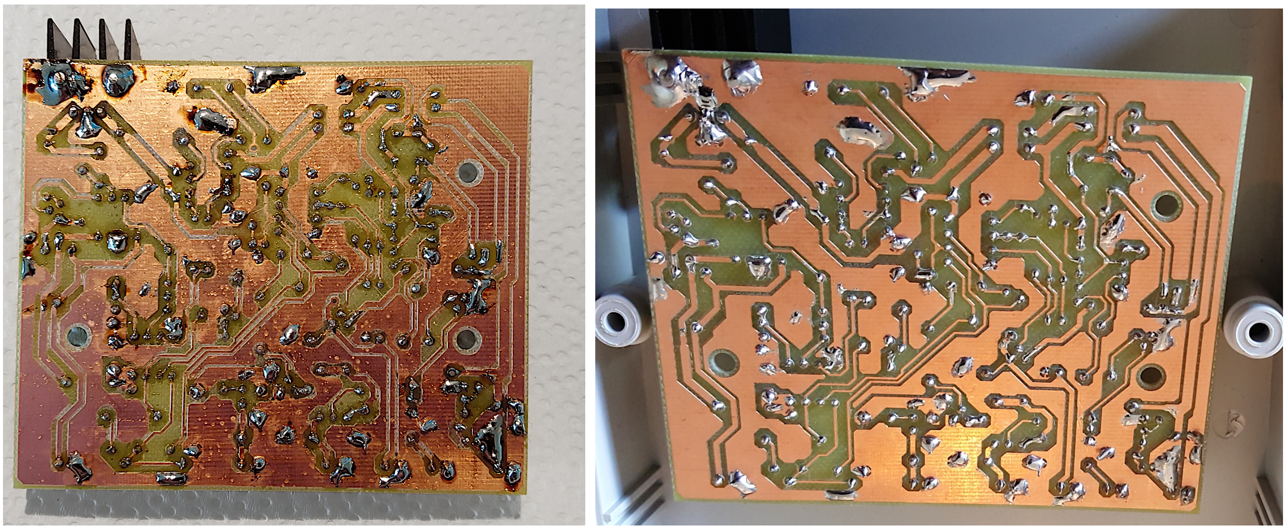
With the heatsink in place, the 7815 voltage regulator was inserted into its place and fastened to the heatsink using a computer case screw (6-32 UNC).

Figure 5: Soldered heatsink and thermal paste pre-applied to TO-220 device

2.3.3 Cleaning

After soldering the board was cleaned with alcohol. Cleaning removes flux residue which may be slightly conductive which maybe could interfere with the well balanced large resistors. But mostly this was done just because it looks nicer.

The soldering happened over two days and the PCB has been cleaned at home in between.



To the left: PCB before cleaning (after first day of soldering)
To the right: PCB after cleaning (after second day of soldering)

Figure 6: PCB before and after cleaning

2.4 Mounting

No 6.4 mm long self-tapping screws were available. The closest match were M3.5 by 9.5 mm. Some plastic washers were used as spacers to compensate for the longer screws.

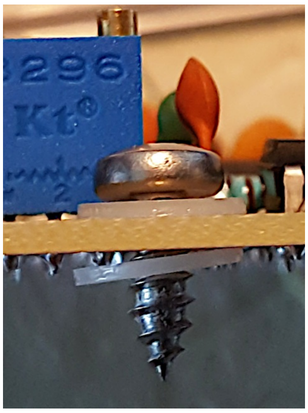
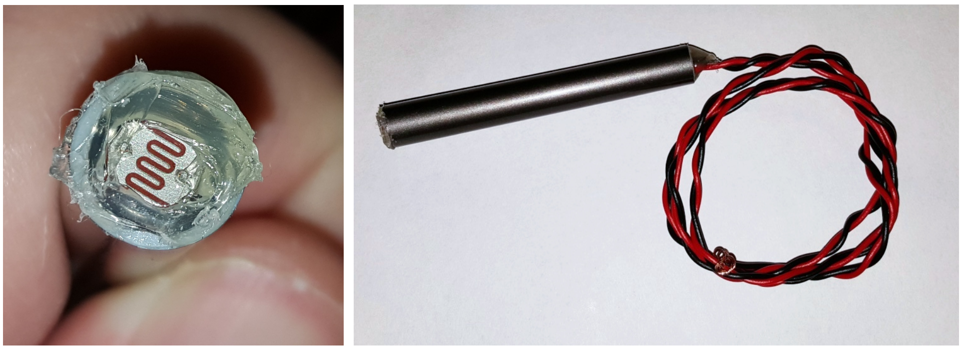


Figure 7: Adding washers to attempt to compensate for longer screws

The washer below the board lifts it up slightly which helps around the missing fourth screw hole and also gives more space for the LDR lead.

Two washers are approximately as thick as the PCB which is 1.6 mm. In hindsight, four of these washers should have been used for each screw instead of just two.

2.5 Test cables (and LDR)



N.B. The wires have later been terminated with a two pin Molex connector.

Figure 8: LDR mounted in plastic tube from ballpoint pen

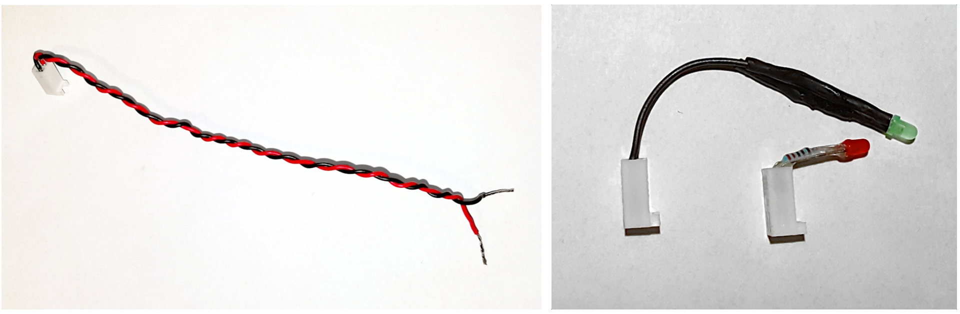


Figure 9: Two-pin Molex to bare wires for tests and LEDs for digital outputs

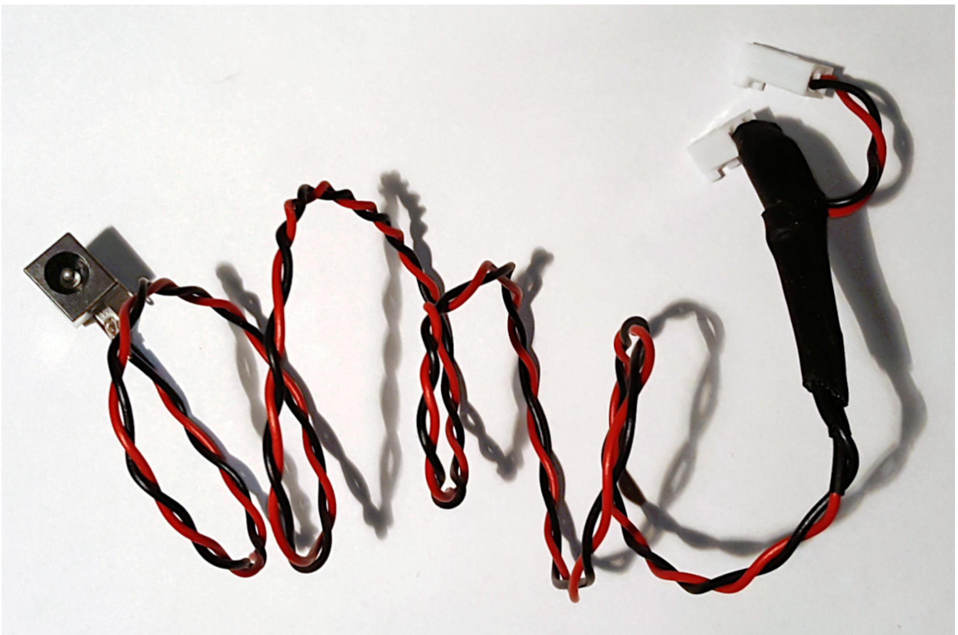


Figure 10: Power cable, with extra connector for `range_select`

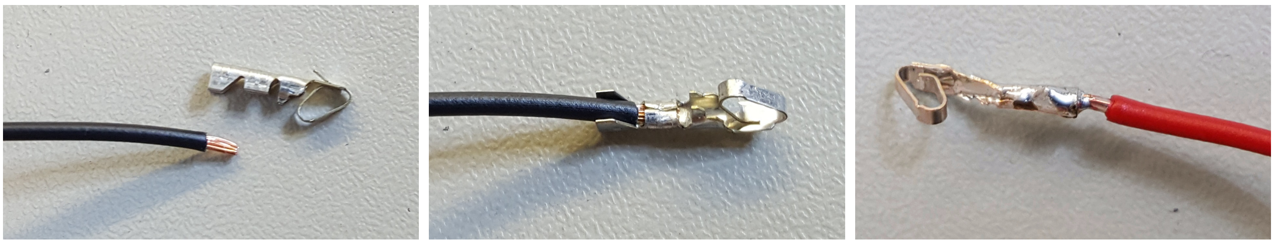


Figure 11: Crimping process

3 Functional testing

3.1 Oscillation in the range limiter

During a test some absolutely horrific oscillation was observed in the range limiting circuitry, this was with P4 set way wrong so the circuit needs to be tested more closely.

3.1.1 More accurate simulations

Improving the op-amp model in LTspice makes the simulation fail in the same way despite having P4 properly set, indicating that there may actually be a problem.

In order to more accurately simulate an LM358 the two `UniversalOpamp2`s had their parameters adjusted to more closely match the actual component. [4]

- Open-loop gain: 100 000 (100 dB), down from 1 000 000 (120 dB)
- Slew rate: 0.4 $V/\mu s$, down from 10 $V/\mu s$
- Gain bandwidth product: 1 MHz, down from 10 MHz.

Crossover distortion has been added by adding a pair of antiparallel diodes in series with the output, the distortion of this is reportedly double that of the actual component. [5]

3.1.2 Physical tests

Even with P4 correctly set, the circuit oscillates badly when the input signal is very stong. For small overloads (eg 1100 Lux on the 1000 scale) the circuit works correctly, but for stronger inputs it starts oscillating.

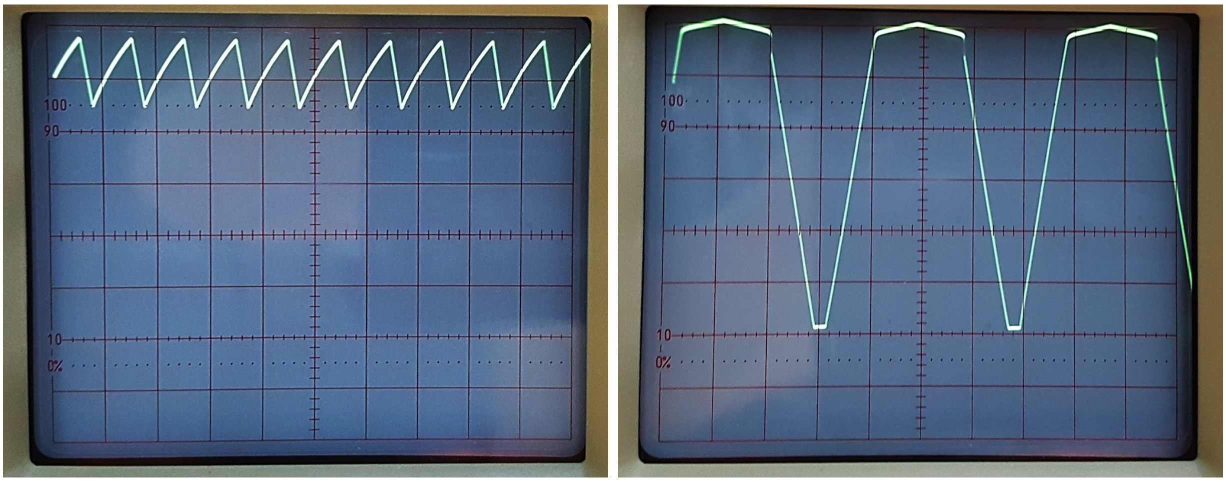
The sensor has been sloppily calibrated in order to make these tests realistic. It's not calibrated well enough for use.

Variables that have been considered:

- Supply voltage: tests have been done at both 19 and 29 volts.
- Input: tests have been done by aiming the LDR at a bright light but also by shorting it out entirely to simulate extreme overload conditions.
- Load on `analog_voltage`: Both open circuit and a 499 Ω resistor have been used on the analog (voltage) output.
- C14: Including this capacitor to attempt to slow down the opamp may have been a mistake. Tests have been made with the 100pF capacitor and with it removed.

All $2^4 = 16$ tests resulted in oscillation, with one test that was spectacularly bad.

Different capacitance values (10 pF, 1 nF, 10 nF, 100 nF and 1 µF) have also been tested but all of them seem to be about equally bad. Removing the capacitor caused the worst observed oscillation.



The 10% line is 0 V, the 110% line is 10 V. 2 volts/div and 20 µs/div. Both images are at 29 V supply voltage, LDR shorted and 499 Ω load on `analog_voltage`. With C14 (100 pF) on the left and without on the right.

Figure 12: With vs without C14

3.1.3 Fixing the crossover distortion in the op-amps

The LM358 is known to have crossover distortion, this was speculated to perhaps cause issues when switching from sinking to sourcing current and vice versa.

The crossover can be mitigated by adding a pull up or pull down resistor to either power supply rail to force one of the transistors to always conduct. [3, 5, 1]

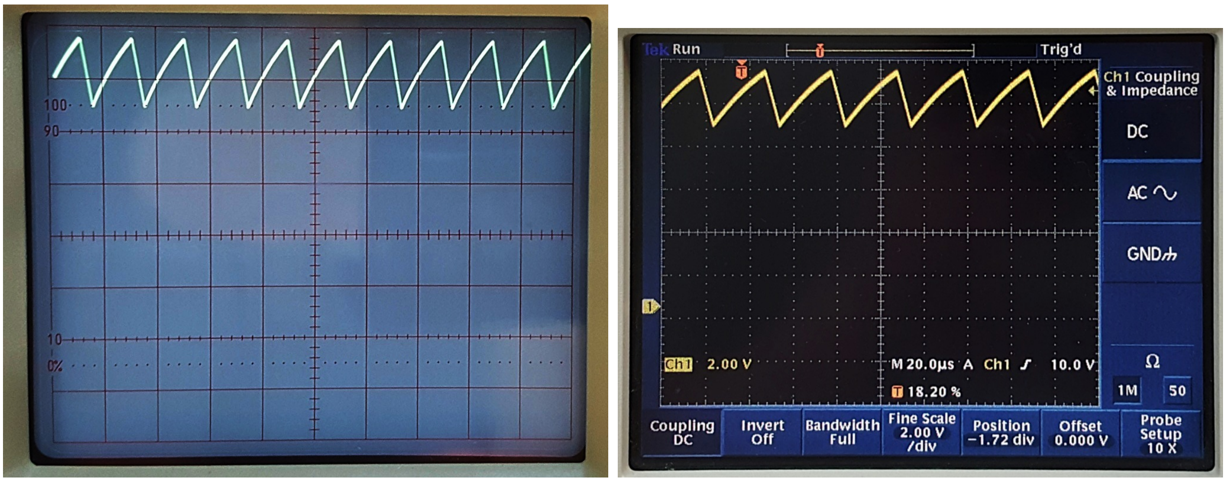
Test conditions:

- LDR shorted
- Supply voltage is 29 volts
- Load on `analog_voltage` is 499 Ω

Various resistance values were tested on the output of IC3a: 4.7, 3.3 and 2.2 $k\Omega$. The lower values seemed to have a larger impact, but there was still significant oscillation. At some point the amplitude of the oscillations appeared to have been cut in half.

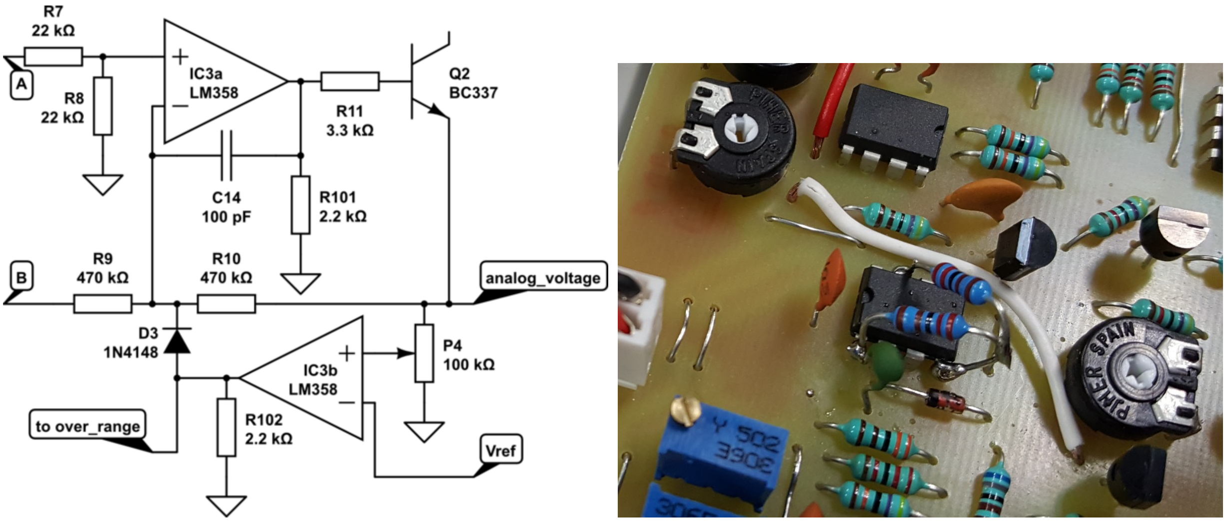
A 2.2 $k\Omega$ resistor was added between output and ground on both IC3a and IC3b. This caused no observable improvement in amplitude, but the frequency is now lower.

In the end, the amplitude of the oscillations is just as large with the added resistors as it was without.



2 volts/div and 20 µs/div on each oscilloscope. Peak to peak signal is approximately 2.5 volts on both. The frequency is lower after adding the one sided load.

Figure 13: Before vs after adding resistors to mitigate oscillations



R101 and R102 were added to push the op-amps from class B operation to class A operation. (Probably unnecessary for IC3b)

Figure 14: Modification to attempt to reduce cross-over distortion in op-amp

4 Measurements

The original intent was to create a test instrument in National Instrument's LabView and get voltage and current readings from a DAQ and the illuminance from a proper light meter and some method of subjecting both light meters to the same light for proper calibration. But due to COVID-19 lockdowns, access to proper lab equipment was delayed. Then the lockdowns ended, further complicating this mess.

Calibration and a lot of measurements was done at home using improvised equipment. At a later point an attempt was made to make a LabView test instrument but due to various issues a custom program using a Modbus TCP DAQ was used instead.

4.1 At home calibration and measurements

4.1.1 Equipment

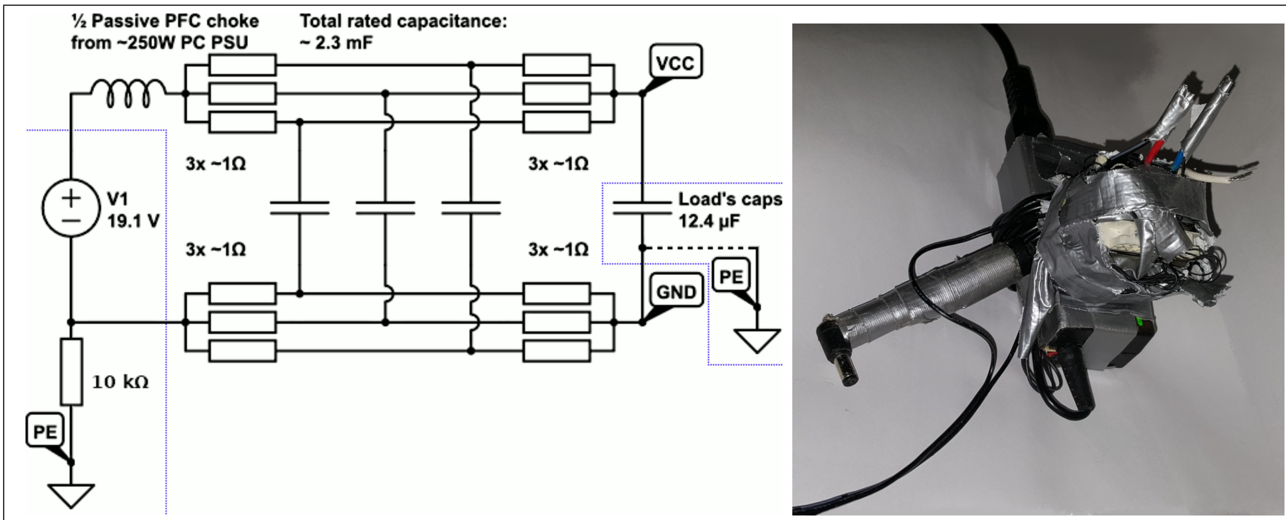
Most of the test leads use bare wires for connecting to the test equipment. Electrical contact was achieved through twisting wires, the connections appear to have hold very well. The only somewhat problematic connection was between the tinned test leads for analog signal output and the multimeter probes as it required twisting a stiff wire around an even stiffer probe.

Copper-clad steel wires with very soft insulation from an old ATA ribbon cable were used as wires for connecting things and as low value resistors.

Light sources	Modified desk lamp, 525 nm LED, natural light through window
Light meter	phyphox on a Samsung SM-G920F
Oscilloscope	Single channel, audio input 44.1 kSamples/s
Strobe light	LED powered from built-in pull-up on a Raspberry Pi (BCM2837B0 SoC)
Power supply	19.1 V only, modified laptop PSU
Voltage/current meter	Axiomet AX-572, V_{ref} measures 2.50 V

Table 2: List of test equipment

The only suitable power supply lying around was a 19 V laptop “charger” that conveniently had DC output on bare wires. A suitable barrel jack plug was taken from a 12 V adapter. It turned out to be quite noisy so a large LC filter was added using a passive PFC choke (unknown inductance) and a bunch of old electrolytic capacitors.

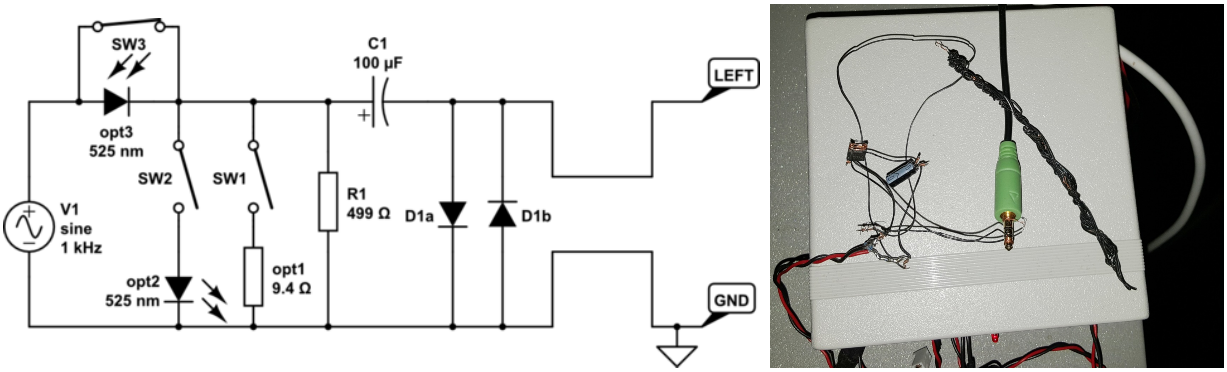


The total capacitance on the output of the power supply is 3.7 mF (measured with the AX-572).

Figure 15: Power supply with LC filter

The desklamp was modified to run from a 12 V DC adapter instead of 12 V AC from the internal transformer. The bulb is a LED equivalent to a 10 W halogen bulb.

An oscilloscope was improvised using the computers audio input and a large capacitor for DC decoupling and antiparallel diodes for protection. The test lead terminated with a 499 Ω resistor was used so it can measure any of the outputs.

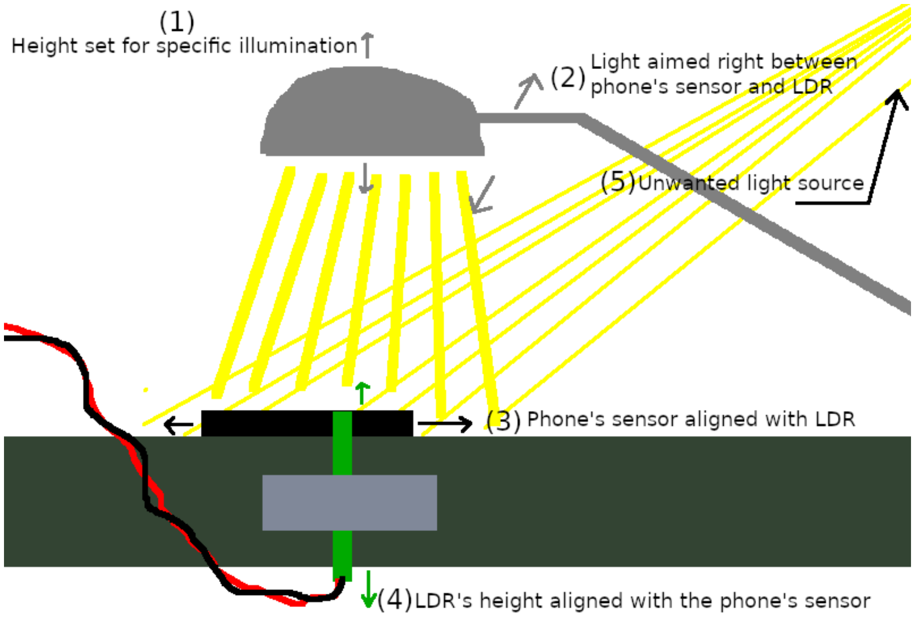


Option 1 connected. A 9.4 ohm wire resistor was added in parellel to be able to measure the entire range of the current output.

Figure 16: "Oscilloscope"

4.1.2 Usage of desk lamp & calibration

The ultra low flicker desk lamp was used as the light source for both calibrations and many if not most of the illuminance/voltage curve measurements.



The tube with LDR duct-taped to the desk and the phone placed on top of the desk with the desklamp shining on them by a close to equal amount.

Figure 17: Usage of the modified desk lamp for controlled illumination

This is very far from perfect. See figure 17: For dark settings (lamp further away) the other light sources in the room (5) become more and more significant and those are not in any way controlled. For bright settings (lamp closer), (2) and (3) will have a larger and larger impact on the angle to the LED bulb and (4) will have a relatively large impact.

The sensor on the phone is much more sensitive to the angle of the light source than the LDR.

This method was used for calibrating both the 1 000 Lux range and the 100 000 Lux range.

Range	Samples	Voltage at 0 Lx
0 - 1 000 Lx	250	0.1298 V
0 - 100 000 Lx	215	0.0157 V

Table 3: Illuminance/voltage data with phone

4.1.3 Illuminance vs voltage

The 1 000 Lux range was tested mostly with the desklamp but some samples are from other sources such as ambient light and a single LED for lower values. The 100 000 Lux range was tested by pointing the LDR and the light sensor of the phone (somewhat) toward the sun and quickly writing down the illuminance and voltage.

The accuracy of measurements is not represented in the plots, nor is it considered in any calculations relying instead on the sheer number of samples.

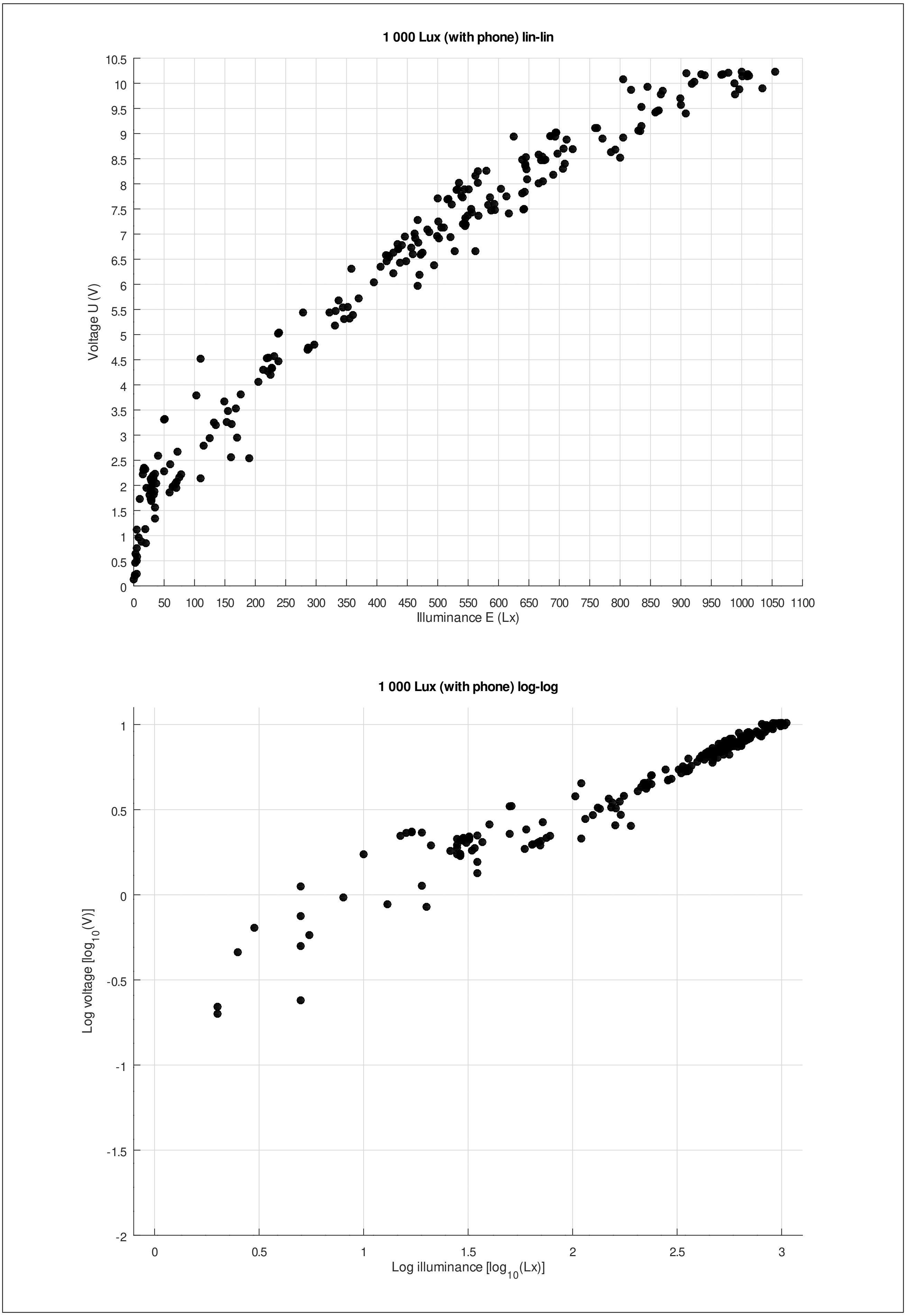


Figure 18: Scatter plots for 1 000 Lux range tested with phone

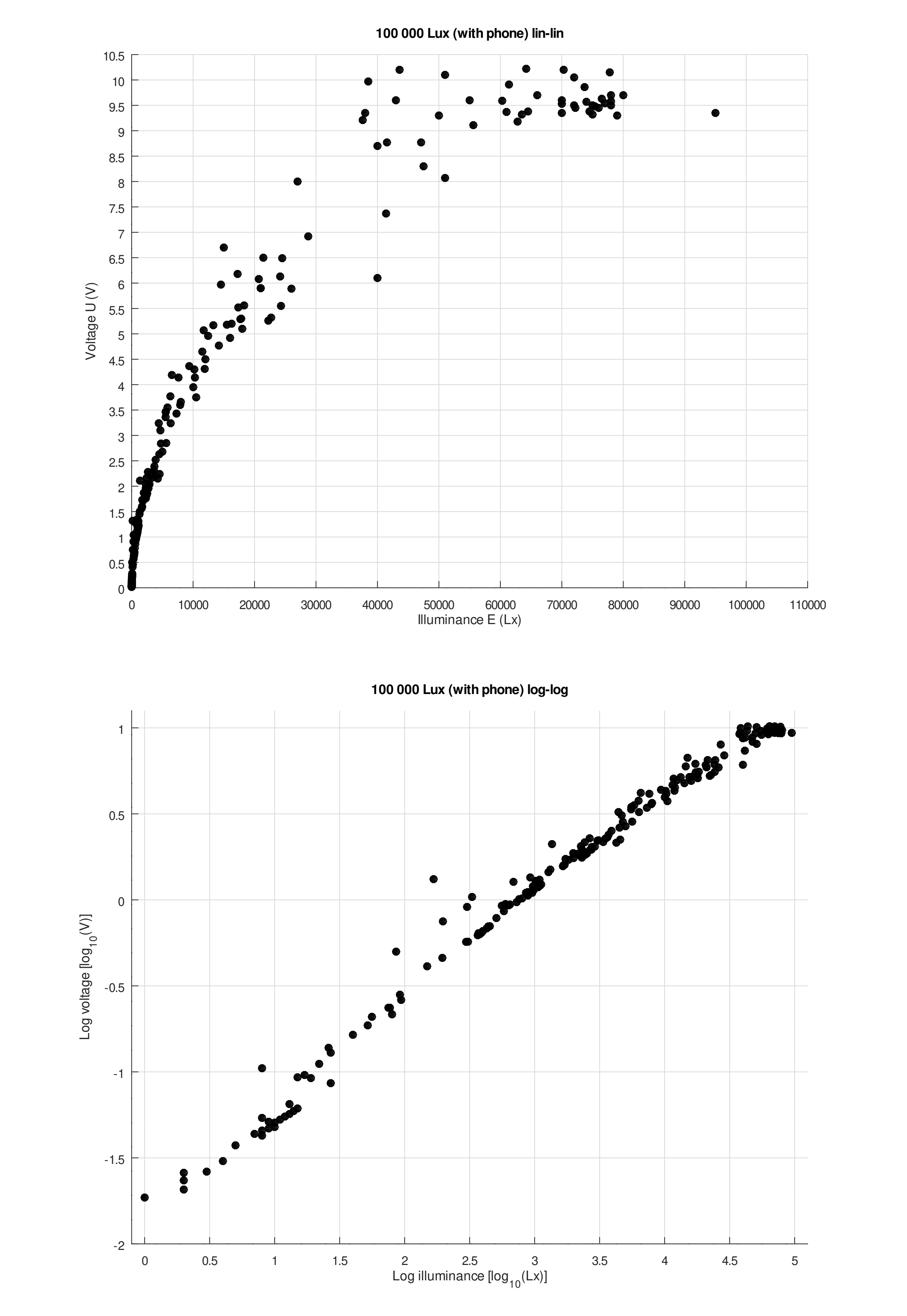


Figure 19: Scatter plots for 100 000 Lux range tested with phone

4.1.4 Time constant

The reader is expected to be familiar with the concept of time constants. If you are not I would recommend this tutorial. This is not an RC circuit, but the same theory applies.

$$\Delta y = y_h - y_l \tag{1}$$

$$y = y_l + \Delta y \cdot \left(1 - e^{-t/\tau}\right) \tag{2}$$

$$y = y_h - \Delta y \cdot \left(1 - e^{-t/\tau}\right) \tag{3}$$

Equation 2 is the rising edge and equation 3 is the falling edge where y is the instantaneous signal level, Δy is the difference between the steady state low (y_l) and high (y_h) levels, t is the time and τ is the time constant.

The exact steady state levels are not possible to determine due to the AC-coupling on the audio input and noise on the low level. The low point has been chosen quite early and may be inaccurate, see figure 22.

Normally the time constant is taken as the time between the signal starts diverging and the signal being at 63.2 % $(1-e^{-1})$, this is the point where $t = \tau$. τ should be identical independent of where it is measured. The time constant for this circuit has been measured over a larger range to see if something strange may be going on:

- τ_1 Between the start and $1 - e^{-1}$ (63.2 %)
- τ_2 Between $1 - e^{-1}$ and $1 - e^{-2}$ (86.5 %)
- τ_3 Between $1 - e^{-2}$ and $1 - e^{-3}$ (95.0 %)
- τ_4 Between $1 - e^{-3}$ and $1 - e^{-4}$ (98.2 %)

See figure 21 or 22 for an example.

The test was performed in the dark with an LED fed with a low-frequency (5 Hz) square wave with sharp edges. The signal measured is the current output through a 9.4 Ω resistor. The voltage levels are unknown.

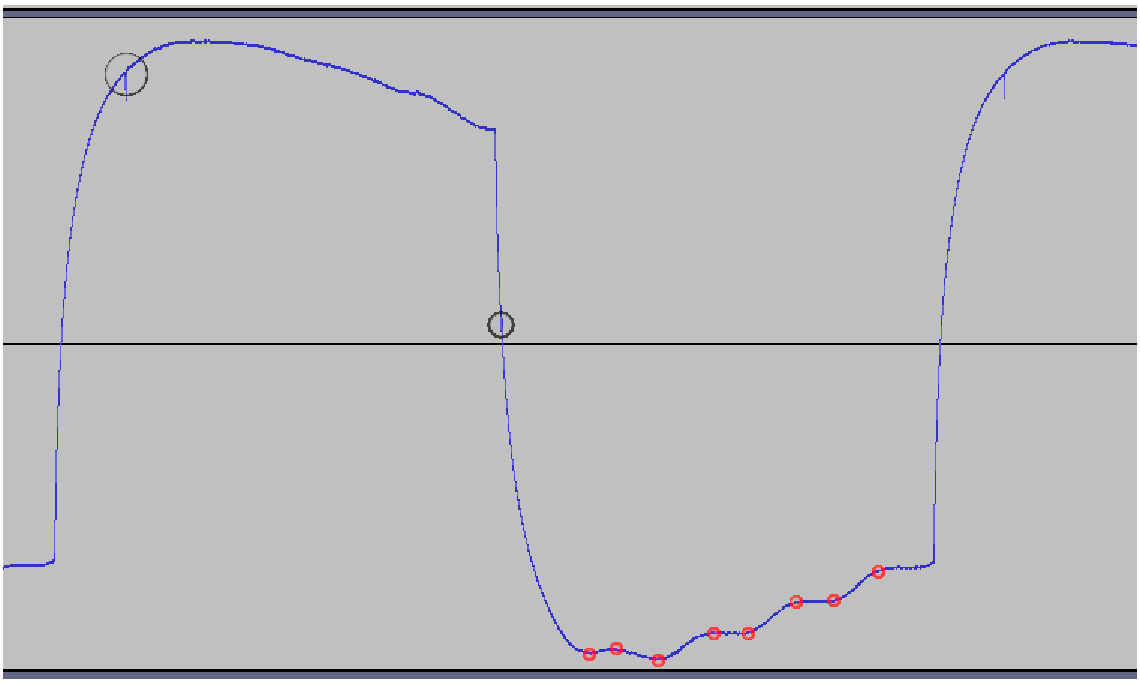
The test was repeated a few times for both rising and falling edge. The results can be seen in table 4 and table 5.

τ in milliseconds

Sample	τ_1	τ_2	τ_3	τ_4	τ_{avg}
1	3.87	6.49	6.96	5.10	5.605
2	3.89	6.70	6.75	5.50	5.710
3	3.79	6.59	6.70	5.87	5.738
4	3.74	6.54	7.06	5.55	5.723
5	3.79	6.59	7.06	5.14	5.645
avg	3.816	6.582	6.906	5.432	5.684
stddev	0.056	0.070	0.153	0.285	0.051

Total: average = 5.684 ms, standard deviation = 1.221 ms

Table 4: Rising time constant



Some unexpected switching? artifacts are circled in black. Some bumps on the lower half, presumably caused by mains hum, are circled in red. The first red point is being used to get y_l for the falling edge.

Figure 20: Measured signal of 5Hz flashes

τ in milliseconds

Sample	τ_1	τ_2	τ_3	τ_4	τ_{avg}
1	4.00	4.88	3.64	2.55	3.768
2	4.00	4.93	3.63	2.44	3.750
3	3.84	4.41	3.43	2.44	3.530
4	3.58	3.63	2.70	1.61	2.880
avg	3.855	4.462	3.350	2.260	3.482
stddev	0.172	0.522	0.385	0.378	0.360

Total: average = 3.482 ms, standard deviation = 0.895 ms

Table 5: Falling timeconstant

As can be seen in the tables, τ does not seem to be constant. Potential sources of distortion:

- Noise pick-up (mains hum)
- The high-pass action of the audio input
- Unkown properties of the LDR or circuit
- Capacitance on the GPIO pin

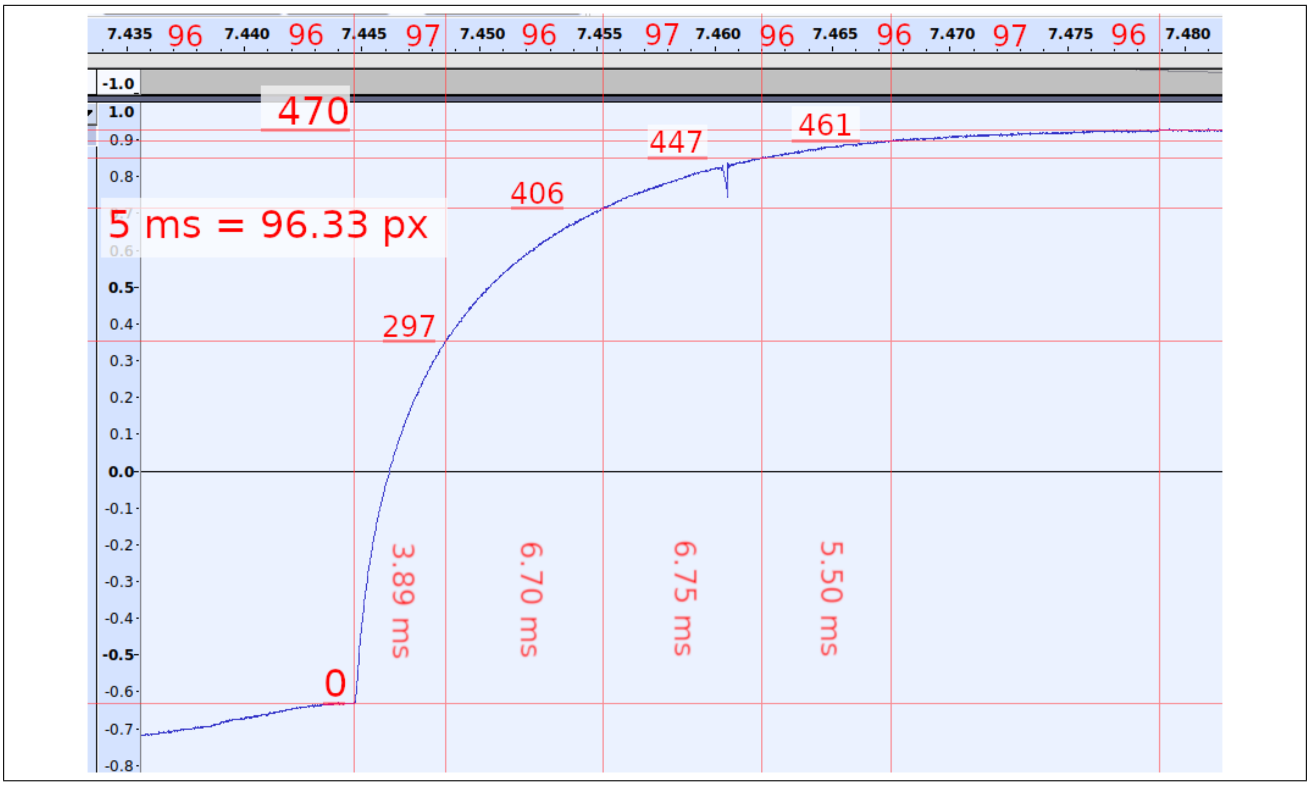


Figure 21: Measuring τ on rising edge

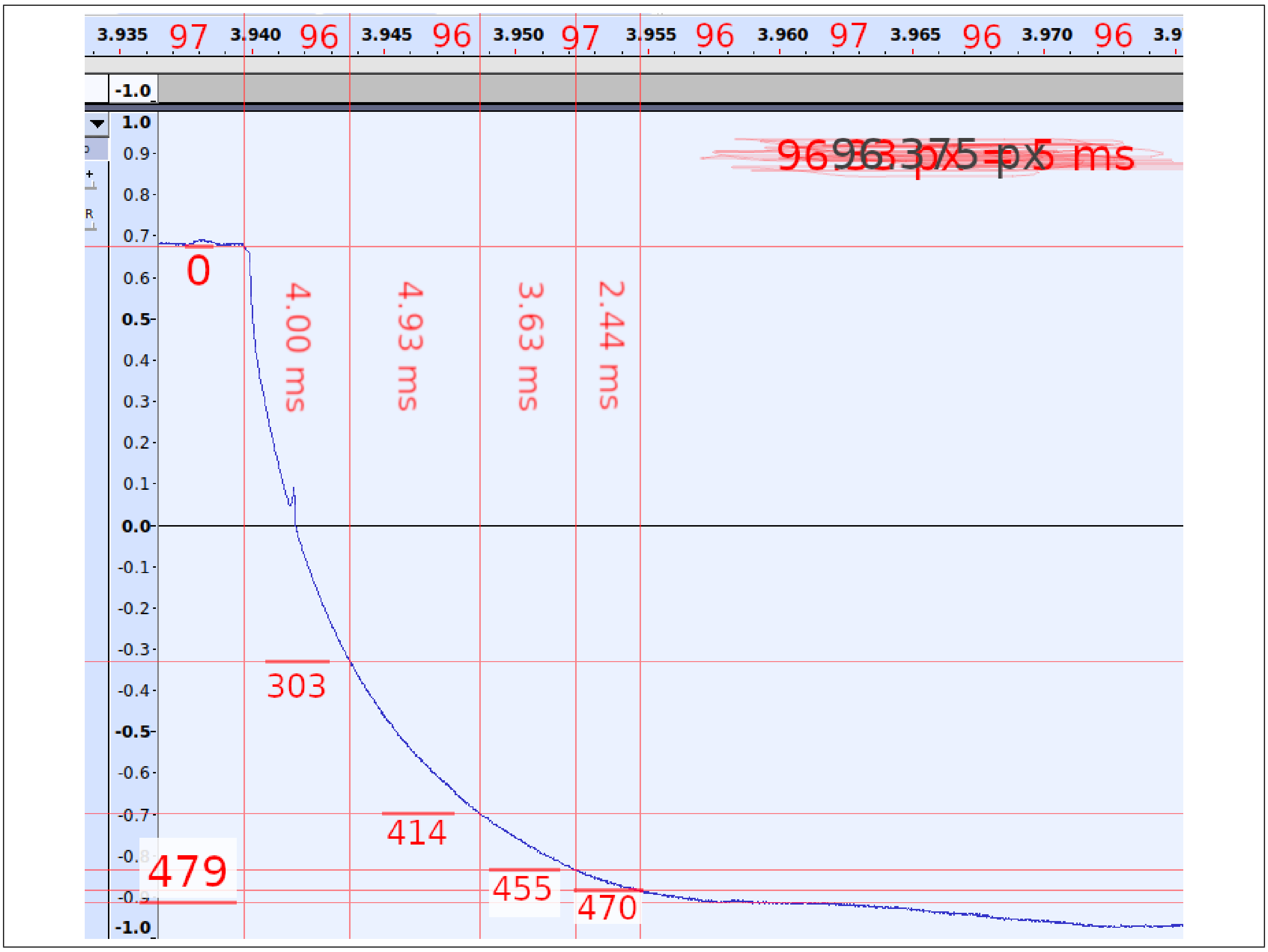


Figure 22: Measuring τ on falling edge

4.2 Testing with proper light meter and Modbus TCP DAQ

More accurate measurements have been taken using a Delta OHM HD2102.1 light meter [6] and a Papouch AD4ETH U [7] to measure the outputs and the supply voltage.

A custom program was written to communicate with the AD4ETH with the following features:

- Several illuminance values are used for each sample, both the mean and standard deviation of the values are logged.
- The AD4ETH measures the analog voltage output, analog current output and the supply voltage. Several samples are used to reduce and measure any noise.
- Measure the outputs through the entire illuminance range at a fixed supply voltage.
- Measure the outputs through the entire supply voltage range at a fixed illuminance.

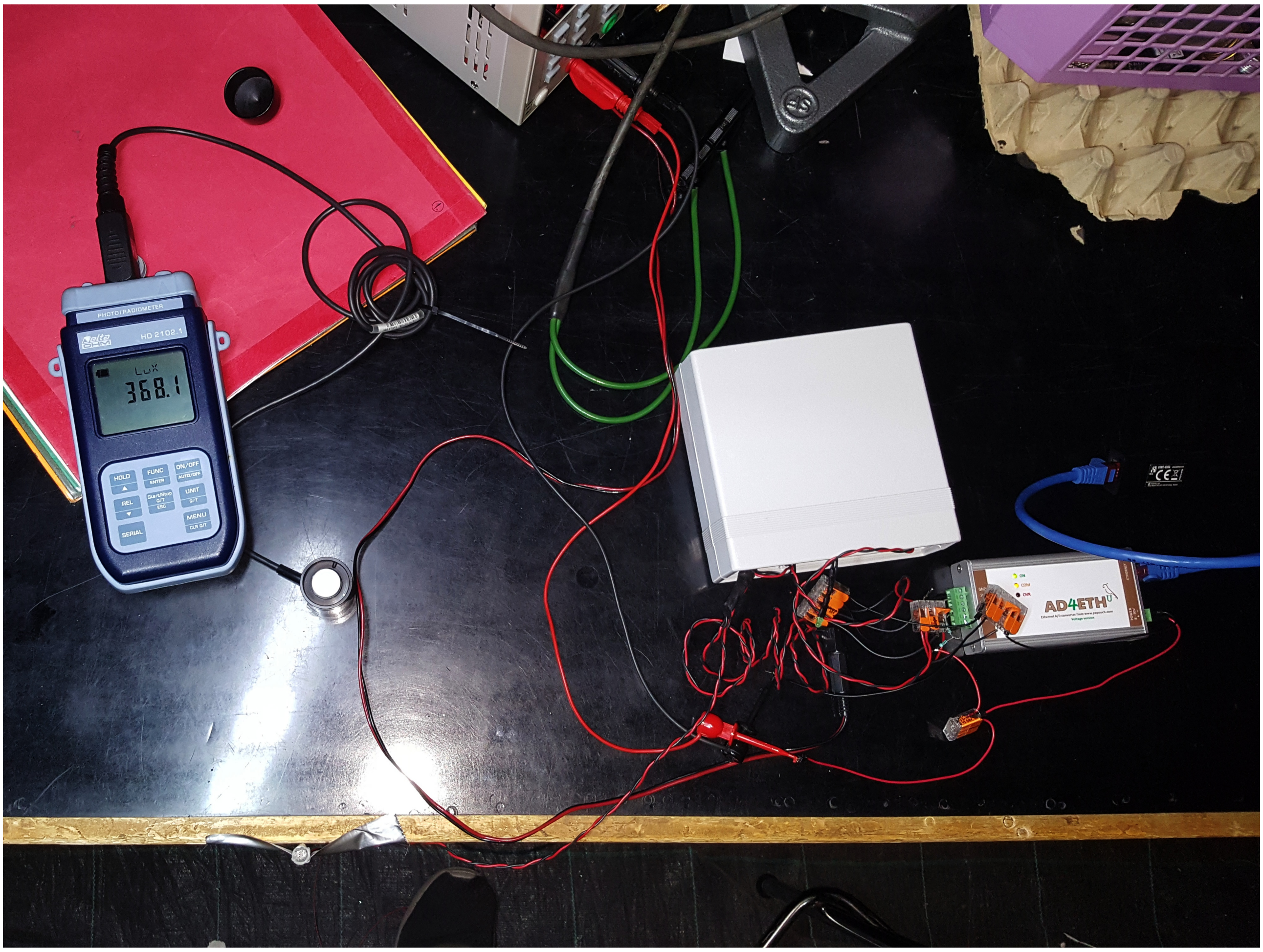
AD4ETH inputs:

1. V_{out} directly measured
2. I_{out} measured with a 500 Ω resistor (actually 498)
3. V_S measured through a 3:1 voltage divider

Some things are as expected and are of no interest:

- The output signals are independent of the supply voltage as long as it is within a tolerable interval.
- The current output is linearly proportional to the voltage output.
- Not much noise. The DAQ is too limited to get any useful graphs of noise.

Source code of the program is in the section "Modbus TCP program".



In the bottom left is the LDR duct taped to the side of the table, above it is the HD2102. To the right are the main electronics, a Papouch AD4ETH U and a rats nest of wires. The HD2102 can be moved around the LDR to measure and compensate for measurement errors. The light source can be adjusted.

Figure 23: Measuring U/E curve with proper equipment

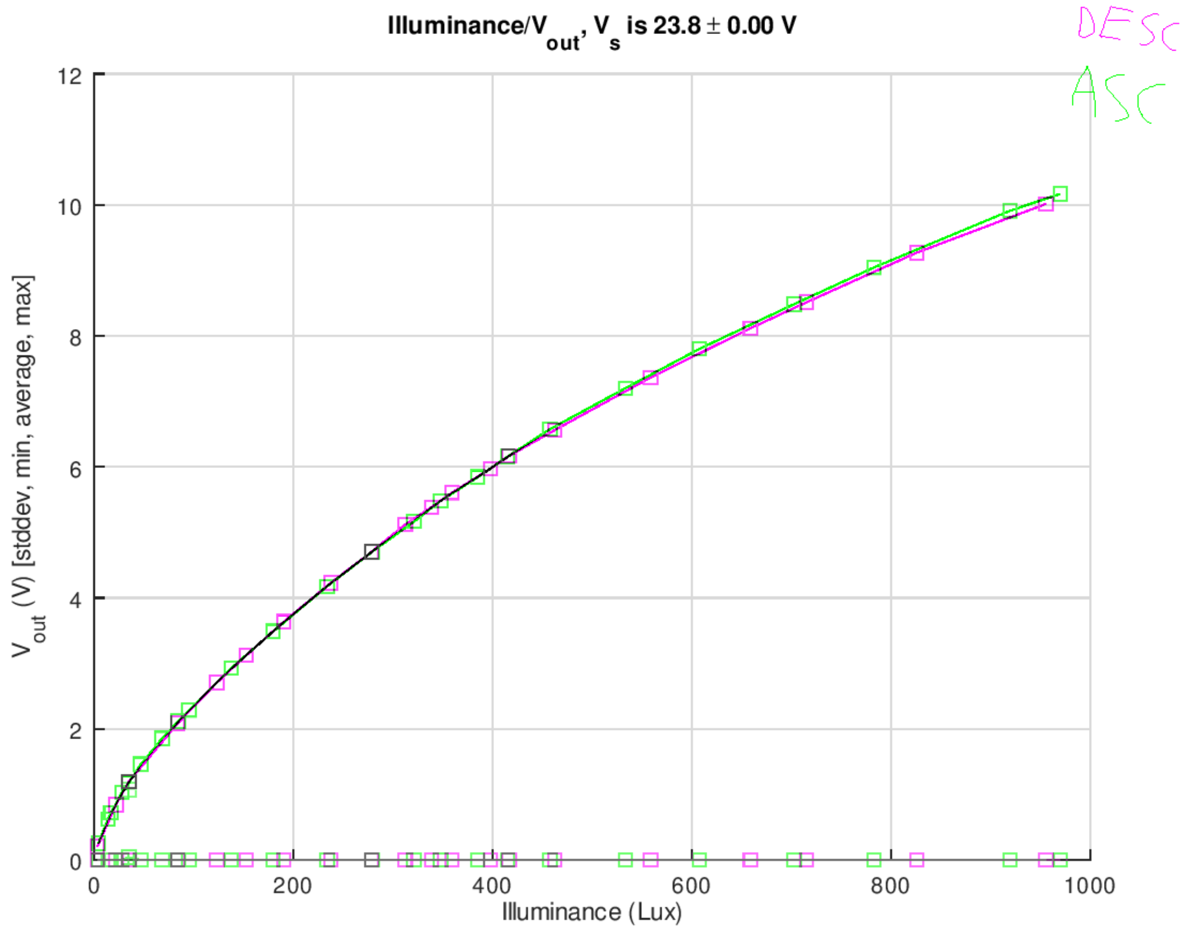
The following has been measured:

- `day1-1k-illum-19.1V`
- `day1-1k-illum-24V-ascending`
- `day1-1k-illum-24V-descending`
- `day1-1k-illum-29V`
- `day1-1k-vsupply-bright-widerange`
- `day1-1k-vsupply-dark`
- `day1-1k-vsupply-medium`
- `day2-100k-illum-24V`
- `day3-1k-illum-24V-ascending`
- `day3-1k-illum-24V-descending`

- `day3-scope-adjustable-light`
- `day3-scope-halogen-unused`

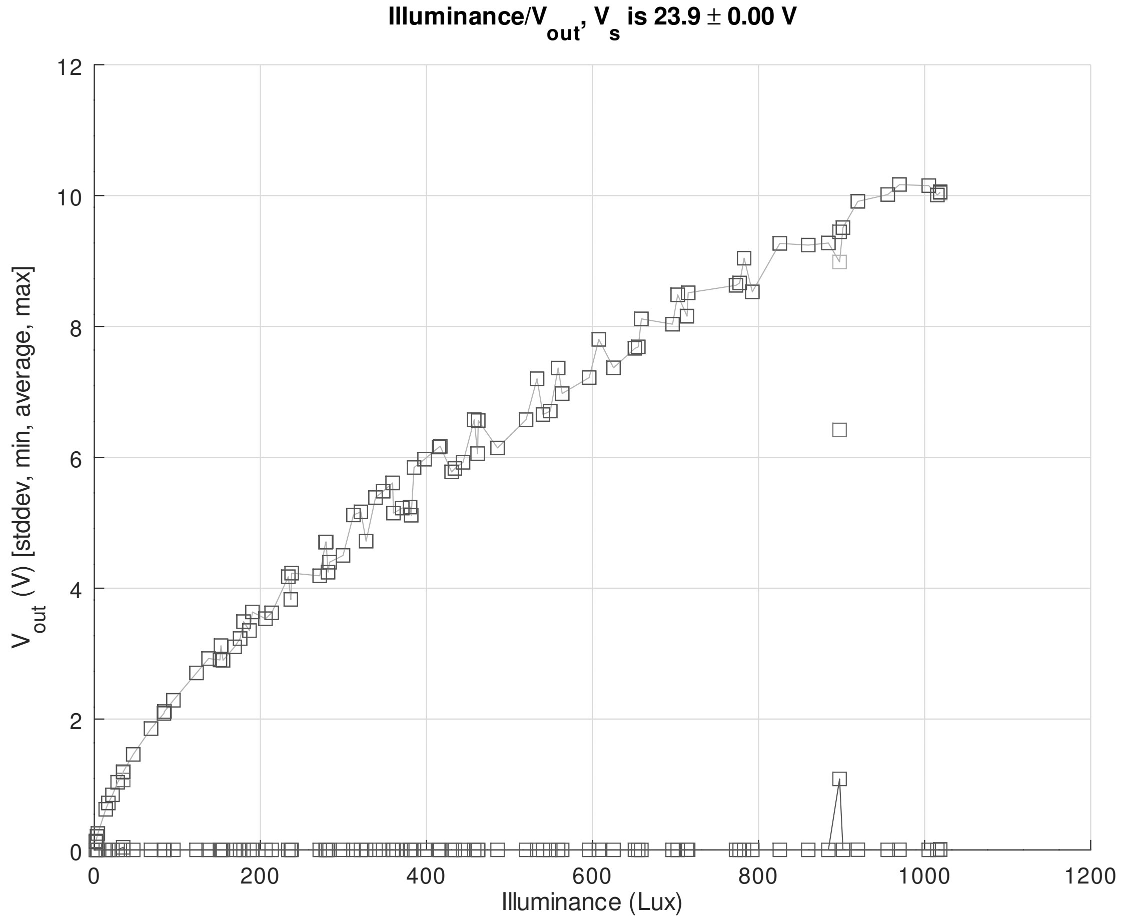
The program can be used, either on platforms where it works or if modified, to show the graphs using the `plot` *command, a compatible DAQ is* ***not*** *required. The datafile (* `2020-12-18` *) is located in the same directory as the program (* `part2/tests/modbus-tcp` *). It has only been tested on Debian with Octave installed, but it should work™ on any unix-like operating system with Python 2.7 and Octave installed.*

Move into the directory, start `./measure_main.py` *and run the command* `load 2020-12-18`*. Then use the* `list`*,* `plot` *and* `help` *commands as needed.*



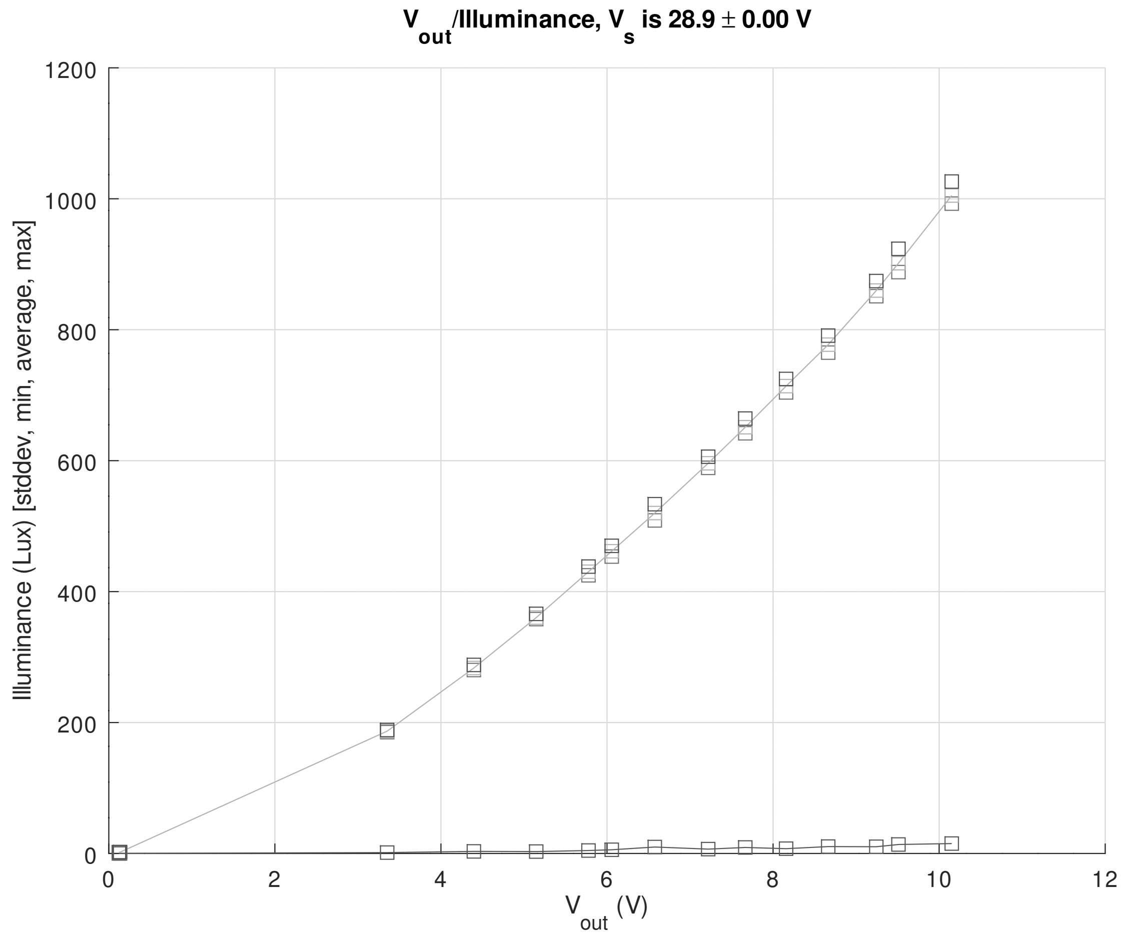
Lines are for ascending and descending averages only, the max and min values only have boxes.

Figure 24: Hysteresis, measured on the third day



All 1k range varied-illuminance measurements combined. The standard deviation only measures short term noise, not long term drift. The jumps in the average value is due to "long" term drift of just a few days.

Figure 25: Repeatability



Maximum, average and minimum recorded illuminance for each sample. Standard deviation is at the bottom. Each point is based on several measurements with the HD2102. There is some variation as the HD2102 is moved around.

Figure 26: Measurement accuracy

4.3 Regression analysis

Source code of the program is in the section "Regression analysis program".

4.3.1 Repeated theory

$$R \propto E^{-\gamma}$$

R is the resistance, E is the illuminance and γ (gamma) is a "constant" equivalent to the sensitivity of the LDR.

Typical values appear to be around 0.5 which means that the conductance of an LDR is approximately proportional to the square root of the illuminance.

Gamma is usually defined in the datasheet as

$$\log\left(\frac{R_{10}}{R_{100}}\right)$$

where R_{10} is the resistance at 10 lux and R_{100} is the resistance at 100 lux.

The light sensor made in the project converts the conductance of the LDR linearly to a voltage, hence:

$$U \propto E^{\gamma}$$

On a logarithmic scale, gamma will be the slope of the curve and all the gamma plots below are made by taking the derivative function of the model an plotting in a logarithmic scale.

4.3.2 Data

- 1k range measured with multimeter and phone
- 1k range measured with AD4ETH and HD2102
- 100k range measured with multimeter and phone
- 100k range measured with AD4ETH and HD2102

Some models will use the logarithms of the voltage and illuminance as input data.

4.3.3 Models

I have been unable to find a more complex model of the resistance of an LDR, but I suspect γ is not constant over a larger range.

The data will be fitted to six different models:

$$U = k_0 \cdot E^{k_1} + U_{offset} \tag{1}$$
$$U = k_0 \cdot E^{k_1} \tag{2}$$
$$\log_{10}(U) = k_1 \cdot \log_{10}(E) + k_0 \tag{3}$$
$$\log_{10}(U) = k_2 \cdot \log_{10}(E)^2 + k_1 \cdot \log_{10}(E) + k_0 \tag{4}$$
$$\log_{10}(U) = k_3 \cdot \log_{10}(E)^3 + k_2 \cdot \log_{10}(E)^2 + k_1 \cdot \log_{10}(E) + k_0 \tag{5}$$
$$\log_{10}(U) = k_4 \cdot \log_{10}(E)^4 + k_3 \cdot \log_{10}(E)^3 + k_2 \cdot \log_{10}(E)^2 + k_1 \cdot \log_{10}(E) + k_0 \tag{6}$$

where U is the voltage in volts and E the illuminance in lux.

Logarithms are performed before least squares fit for equations three to six.

The offset voltage is hard coded to 0.1298 V for the 1k range and 0.0157 V for the 100k range. These values are measured with the multimeter and may be off for data from the AD4ETH.

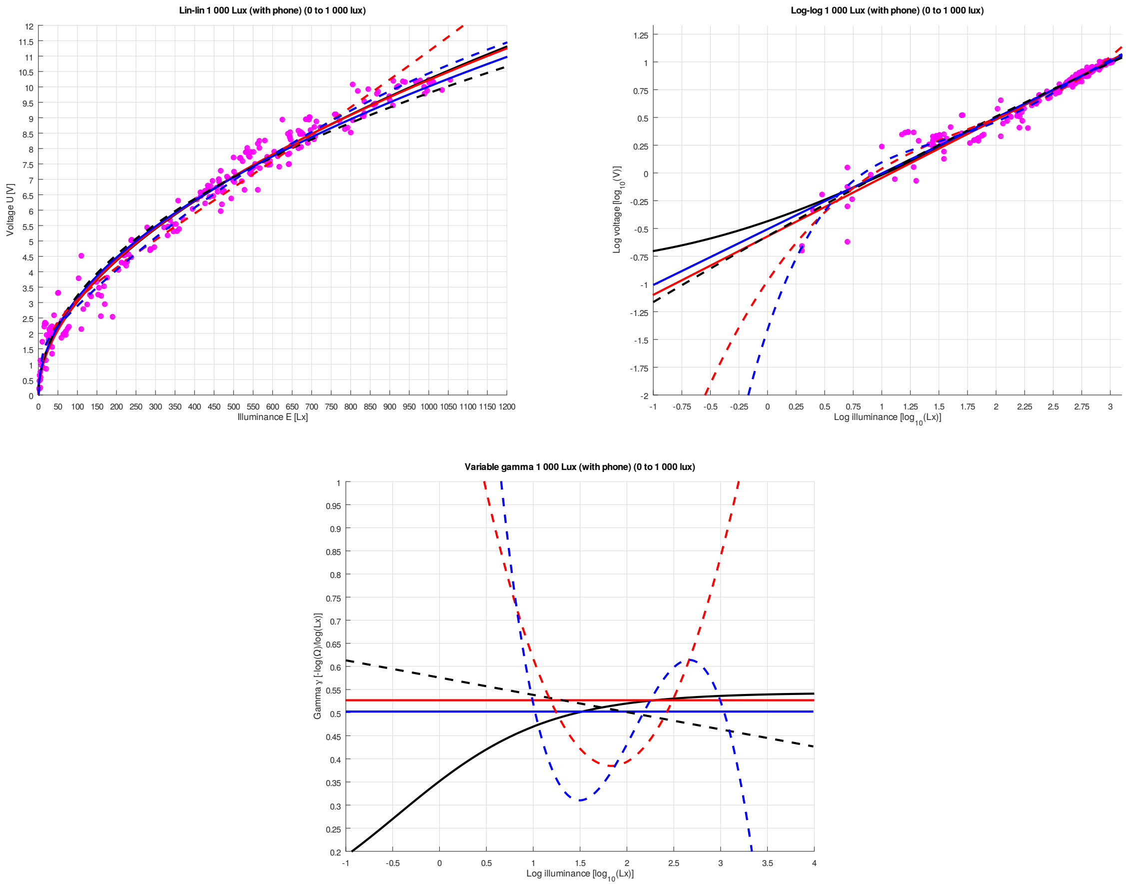
4.3.4 Results

Line	Equation
Solid black	1
Solid red	2
Solid blue	3
Dashed black	4
Dashed red	5
Dashed blue	6

γ is observed to be variable, lower at higher illuminance levels.

The observed gamma value is different between tests made with the phone and with the HD2102 hinting that the light sensor in the phone has a noticeable non-linearity.

Phone – 1k



Linear-linear illuminance to voltage, logarithmic-logarithmic illuminance to voltage, γ (sensitivity) as a function of logarithmic illuminance

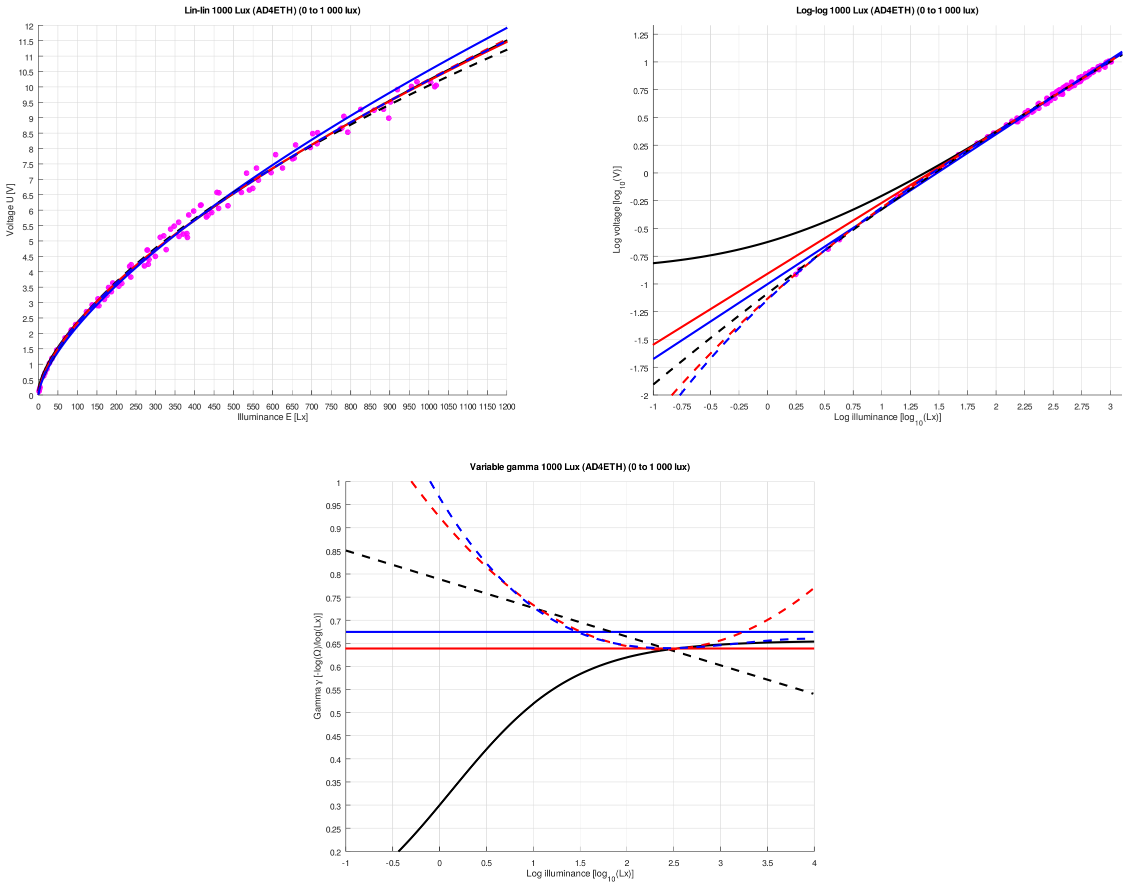
Model	r^2	γ at 10 and 100 Lux	Observations
1 (Solid black)	0.9775	(0.5431)	Bad fit on log plot
2 (Solid red)	0.9768	0.5268	Acceptable
3 (Solid blue)	0.9406	0.5024	Acceptable
4 (Dashed black)	0.9412	0.540/0.504	Acceptable
5 (Dashed red)	0.9534	0.620/0.393	Bad fit on lin plot, doesn't extrapolate
6 (Dashed blue)	0.9572	0.543/0.432	Doesn't extrapolate

Parameters:

Model	k_0	k_1	k_2	k_3	k_4 or U_{offset}
1	0.237766	0.543088			0.129800
2	0.268535	0.526831			
3	-0.506641	0.502405			
4	-0.568727	0.575993	-0.018668		
5	-0.972417	1.513753	-0.613678	0.111182	
6	-1.412182	3.093947	-2.284064	0.792977	-0.095120

Figure 27: Regression analysis for 1k range with phone

HD2102 − 1k



Linear-linear illuminance to voltage, logarithmic-logarithmic illuminance to voltage, γ (sensitivity) as a function of logarithmic illuminance

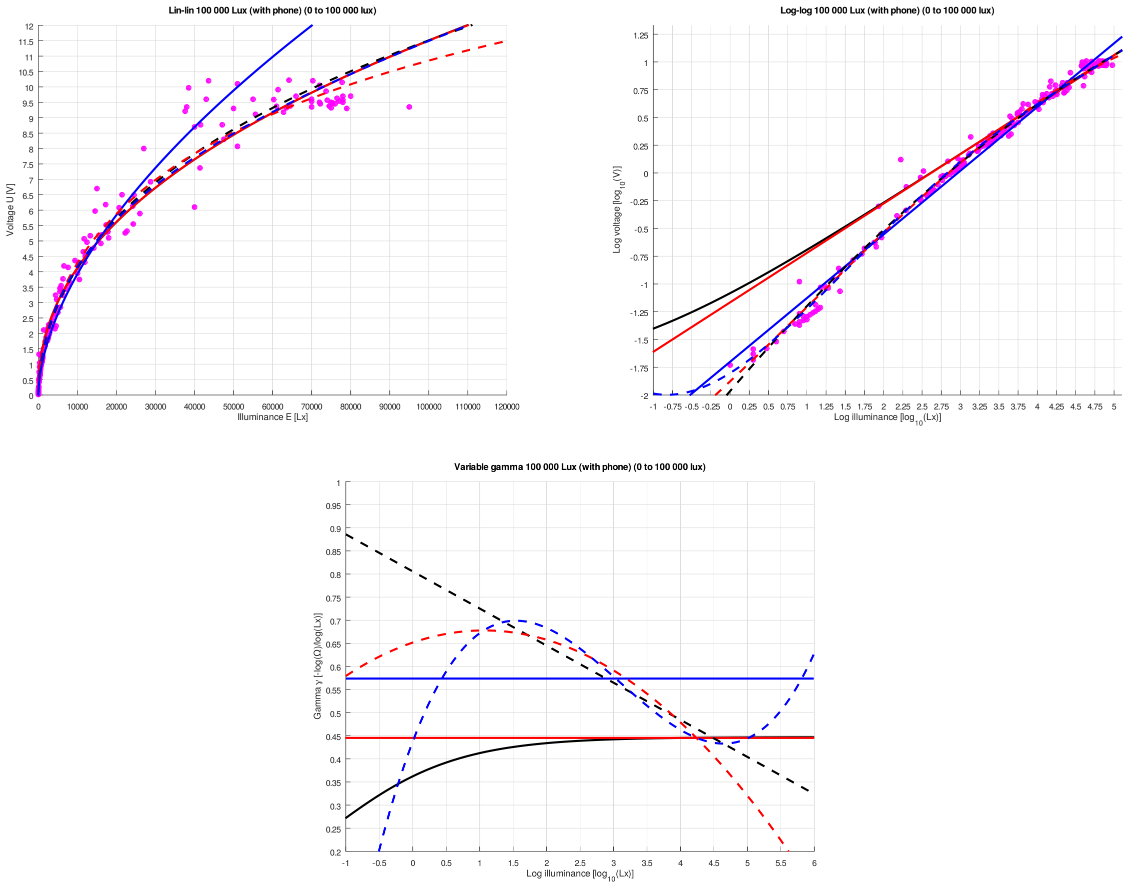
Model	r^2	γ at 10 and 100 Lux	Observations
1 (Solid black)	0.9940	(0.6556)	Bad fit on log plot
2 (Solid red)	0.9943	0.6388	Acceptable
3 (Solid blue)	0.9967	0.6747	Acceptable
4 (Dashed black)	0.9982	0.727/0.665	Acceptable
5 (Dashed red)	0.9984	0.734/0.644	Acceptable
6 (Dashed blue)	0.9984	0.729/0.693	Acceptable

Parameters:

Model	k_0	k_1	k_2	k_3	k_4 or U_{offset}
1	0.108977	0.655589			0.129800
2	0.123673	0.638774			
3	-1.001327	0.674670			
4	-1.085690	0.789041	-0.031058		
5	-1.132386	0.924567	-0.120805	0.016901	
6	-1.142212	0.967297	-0.169340	0.037457	-0.002932

Figure 28: Regression analysis for 1k range with HD2102

Phone – 100k



Linear-linear illuminance to voltage, logarithmic-logarithmic illuminance to voltage, γ (sensitivity) as a function of logarithmic illuminance

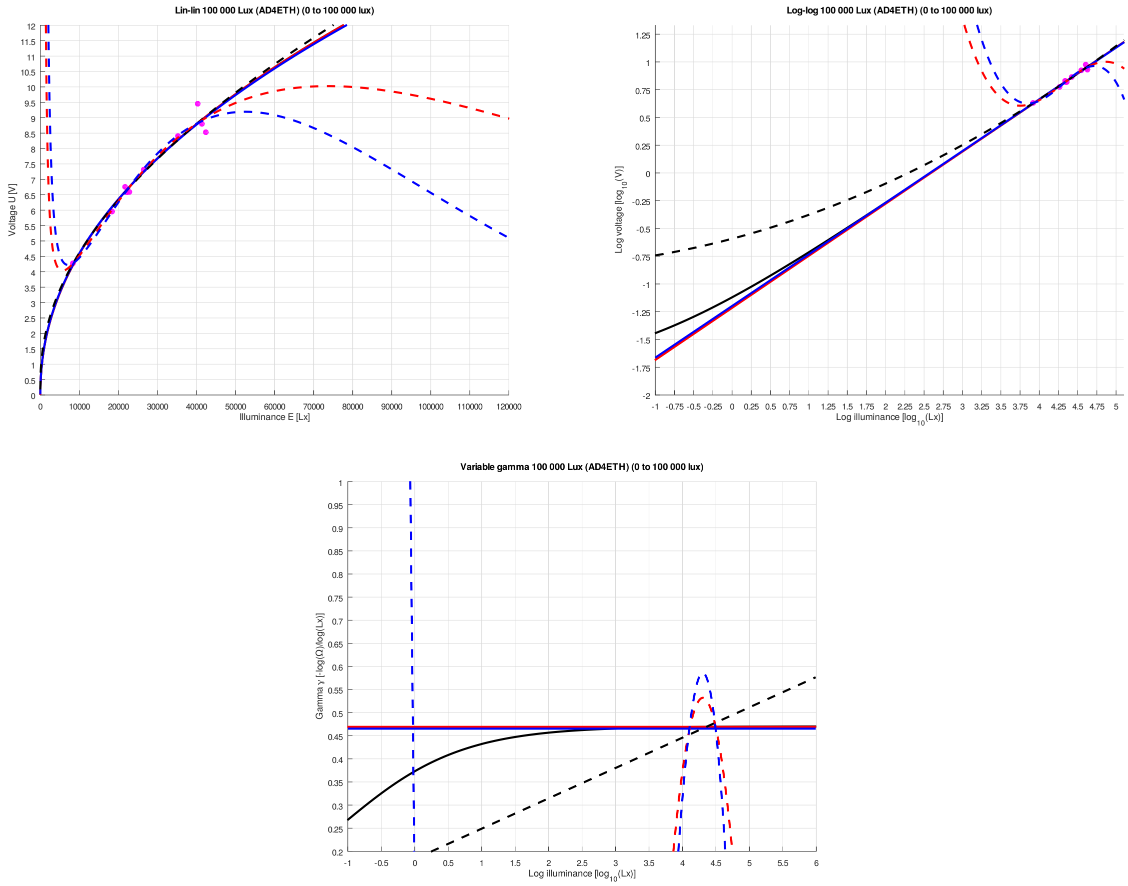
Model	r^2	γ at 10 and 100 Lux	Observations
1 (Solid black)	0.9755	(0.4470)	Bad fit on log plot
2 (Solid red)	0.9757	0.4453	Bad fit on log plot
3 (Solid blue)	0.9820	0.5737	Bad fit on lin plot
4 (Dashed black)	0.9905	0.725/0.644	Acceptable
5 (Dashed red)	0.9910	0.678/0.658	Acceptable
6 (Dashed blue)	0.9912	0.671/0.683	Gamma curve seems off

Parameters:

Model	k_0	k_1	k_2	k_3	k_4 or U_{offset}
1	0.066916	0.446975			0.015700
2	0.068261	0.445319			
3	-1.700996	0.573735			
4	-1.965399	0.805982	-0.040138		
5	-1.877128	0.651369	0.024729	-0.007699	
6	-1.802422	0.435425	0.191642	-0.055050	0.004442

Figure 29: Regression analysis for 100k range with phone

HD2102 – 100k



Linear-linear illuminance to voltage, logarithmic-logarithmic illuminance to voltage, γ (sensitivity) as a function of logarithmic illuminance

Model	r^2	γ at 10 and 100 Lux	Observations
1 (Solid black)	0.9647	(0.4703)	
2 (Solid red)	0.9647	0.4692	
3 (Solid blue)	0.9784	0.4654	
4 (Dashed black)	0.9787	0.249/0.314	Gamma low and increasing
5 (Dashed red)	0.9814	-18.2/-8.61	Doesn't extrapolate
6 (Dashed blue)	0.9825	-7.62/-7.28	Doesn't extrapolate

Parameters:

Model	k_0	k_1	k_2	k_3	k_4 or U_{offset}
1	0.060115	0.470314			0.015700
2	0.060926	0.469216			
3	-1.198418	0.465375			
4	-0.592551	0.182959	0.032818		
5	44.211228	-31.279125	7.380965	-0.570854	
6	19.756969	0.163843	-6.648575	2.080151	-0.181702

Figure 30: Regression analysis for 100k range with HD2102

Dataset	γ at 10 % max illuminance for model n					
	1	2	3	4	5	6
Phone 1k	0.52	0.527	0.502	0.504	-	-
HD2102 1k	0.62	0.639	0.675	0.665	0.644	0.693
Phone 100k	0.447	0.445	0.574	0.485	0.480	-
HD2102 100k	0.470	0.469	(!) 0.465	(!) 0.446	-	-

The data points from HD2102 100k only cover a narrow range on the logarithmic scale.

Table 6: Regression analysis summary: typical sensitivity

Model	Observations
2 (linear scale, $U \propto E^\gamma$, without offset)	Baseline; observe that 1 and 2 are swapped in this table
1 (linear scale, $U \propto E^\gamma$, with voltage offset)	Lower r^2 than without the offset, higher γ for the 1k range. There's not much use in compensating for a voltage offset.
3 (logarithmic scale, straight line)	If there were no errors this would be identical to #2. Errors appear smaller in the upper end and larger in the lower end compared to #2.
4 (logarithmic scale, quadratic polynomial)	The most suited for future work of all the tested models. It usually yields reasonable looking results, has a non-constant γ that decreses with increasing light intensity but does not reach zero before 10^{20} lux.
5 (logarithmic scale, cubic polynomial)	Fits data better than #4 as expected. k_3 appears negative for the 100k range and positive for the 1k range, it's likely an artifact of over-fitting.
6 (logarithmic scale, 4th order polynomial)	Fits data better than #5 as expected but appears to have even more issues in the gamma plot and when extrapolating. It behaves surprisingly well with the data from the 1k range tested with the HD2102.

Table 7: Regression analysis summary: Evaluation of models

Dataset	Evaluation
Phone 1k	γ is low, probably due to non-linear sensor in the phone. Also lots of noise due to method of measurement.
HD2102 1k	Less noise and even high order polynomials behave nicely with the data. γ matches what the datasheet for PGM5526 [2] claims. This will be used for the 1k range.
Phone 100k	γ is low, probably due to non-linear sensor in the phone. Also lots of noise due to method of measurement.
HD2102 100k	Insufficient data to generate a useful regression.

There is no dataset suitable for the 100k range.

Table 8: Regression analysis summary: Evaluation of datasets

4.3.5 Reversing the formula and adding tolerance

Solve quadratic equation:

$$\log_{10}(U) = k_2 \log_{10}(E)^2 + k_1 \log_{10}(E) + k_0 \rightarrow \log_{10}(E) = \frac{-k_1 \pm \sqrt{k_1^2 - 4k_2k_0 + 4k_2 \log_{10}(U)}}{2k_2}$$

$$E = 10^{\left(\frac{-k_1 \pm \sqrt{k_1^2 - 4k_2k_0 + 4k_2 \log_{10}(U)}}{2k_2}\right)}$$

Data for the 1k range (HD2102):

k_0	-1.085690
k_1	0.789041
k_2	-0.031058

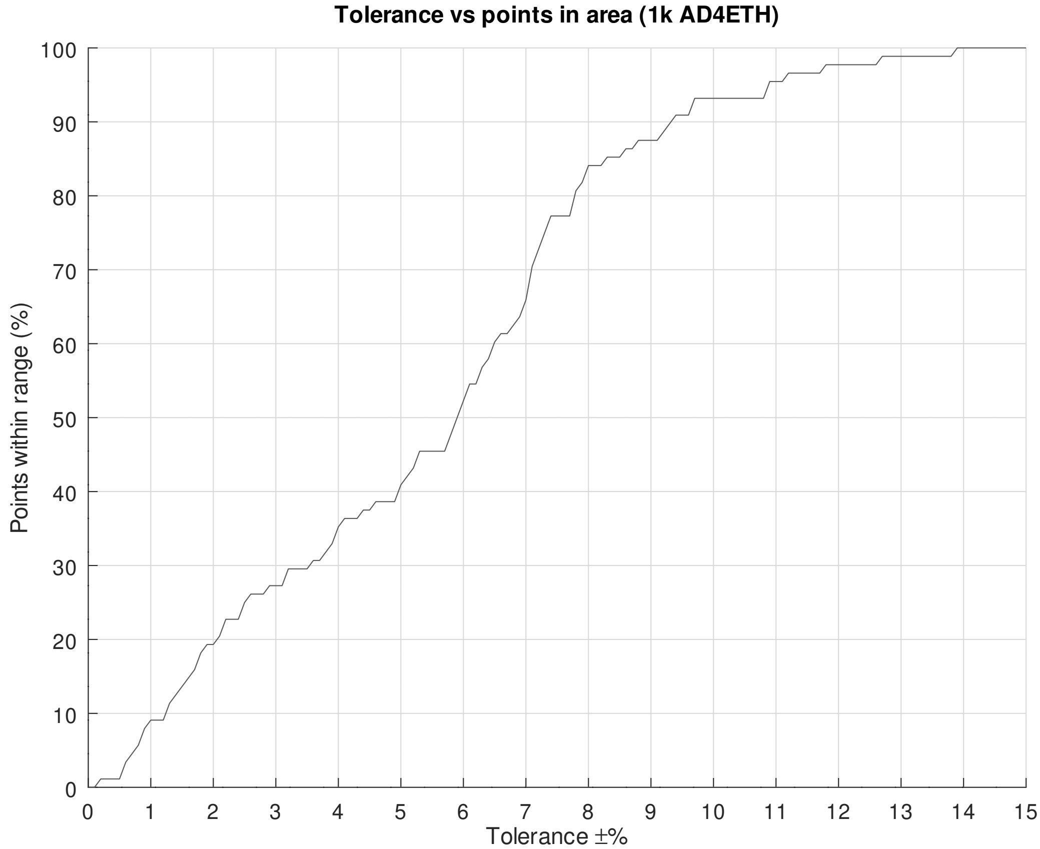
Apply numbers:

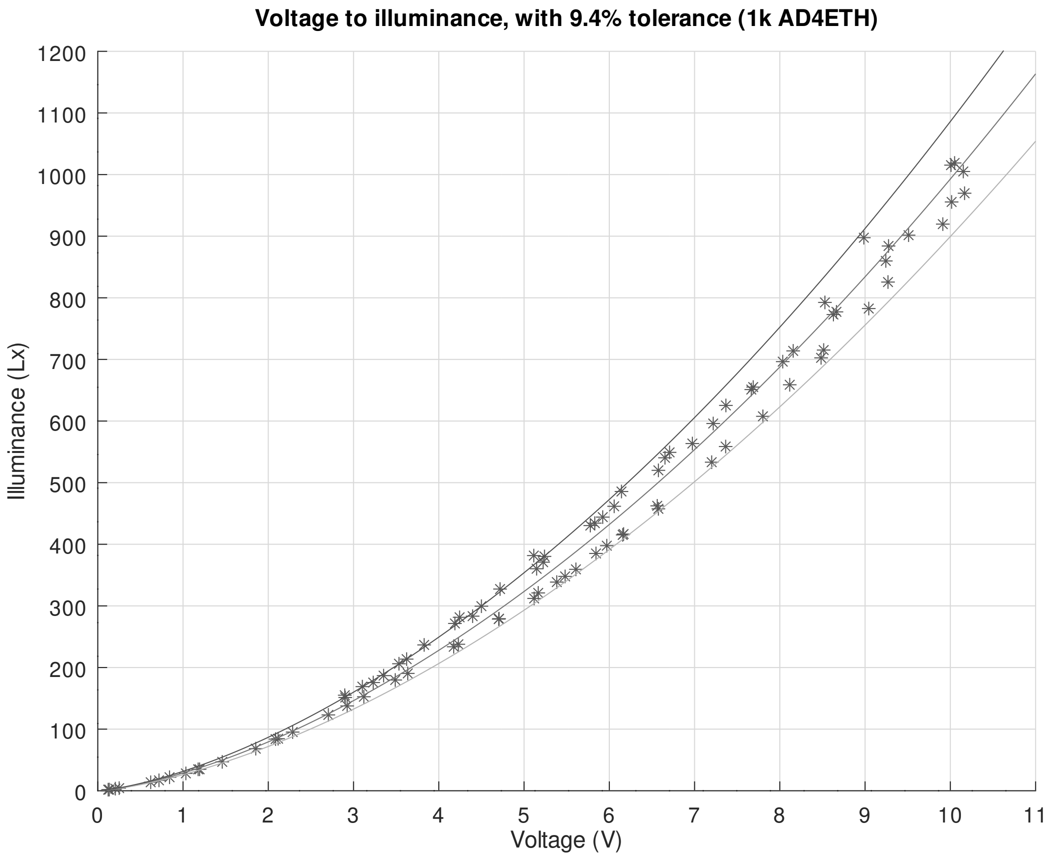
$$E = 10^{\left(\frac{-0.789041 \pm \sqrt{0.487708 - 0.124232 \log_{10}(U)}}{-0.062116}\right)}$$

Deduce that plus minus must be plus (which becomes minus) and simplify further:

$$E = 10^{\left(\frac{0.789041 - \sqrt{0.487708 - 0.124232 \log_{10}(U)}}{0.062116}\right)}$$

To get the accuracy, a percentual tolerance was added large enough to cover a made up number (90%) of data points. The program to add 0.1% at a time until the 90% target is met is in the section "Accuracy measuring program".





For 90% coverage, a tolerance of 9.4% is sufficient.

Figure 31: Accuracy (AD4ETH & HD2102, 1k range)

5 Summary and evaluation

5.1 Project evaluation

Mostly fine, but some issues:

- "Unevenly engineered", some parts are over engineered while other parts could have used more attention.
- The limiter that is supposed to limit the output voltage to 10 V is oscillating with peaks up to 12 V. And it has been left that way, the root cause has not been found.
- The report structure has been made up as I went along.
- The axes should have been swapped in "Regression analysis", the inverse function of the chosen model looks ridiculous and the cubic and 4th order models would be absolute PITAs to invert.
- Everything is extremely overdue, and yet some parts seem half done.

5.2 Evaluation of the prototype

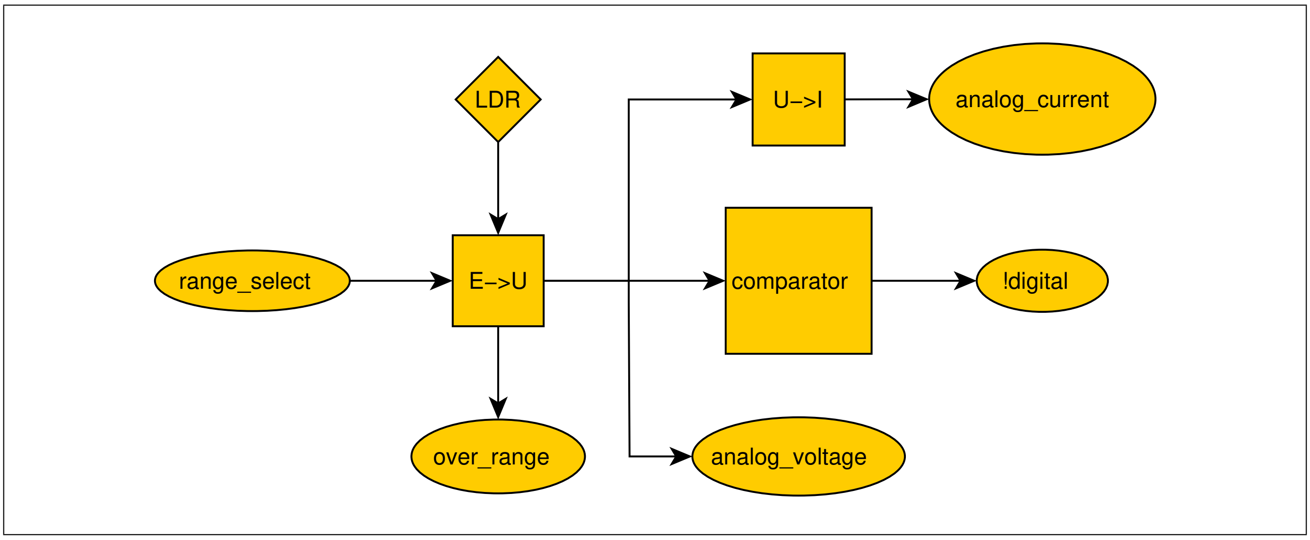


Figure 32: Topology

The topology of the circuit affects how the data has been summarised in the tables below. The E->U block handles both ranges, see table 9 (1k range) and table 10 (100k range). The U->I block is described only by table 11 (current). Table 12 (digital) covers the signals `range_select` and `over_range` which are part of the E->U block as well as the comparator block and the signal `digital`.

1k range, voltage output	
Parameter	**Value**
Accuracy	10 %-ish. See figure 31
Drift	Observed. Not measured. Primary cause for poor accuracy. See figure 25. Included factors: Short amount of time (around a week), small temperature difference (around 5 K) and exposure to bright light.
Repeatability	Short term good, see figure 24. Long term poor, see figure 25. This is not due to poor measurements as can be seen in figure 26.
Hysteresis	None or very little. See figure 24.
Linearity	Very non-linear. Voltage output is approximately a fractional power of the illuminance
LDR γ	0.665 at 100 Lx. See dashed black traces in figure 28.
Rise time (time constant)	4.4 to 7.0 ms (quarter time to 98.2%) or 3.7 to 3.9 ms (time to 63.2%). See table 4
Fall time (time constant)	3.1 to 3.9 ms (quarter time to 98.2%) or 3.6 to 4.1 ms (time to 63.2%). See table 5
Output voltage range	0 - 10 V normally. 12 V max if illuminance exceeds 1000 lux, see left trace on figure 12.
Output load dependence	Not thoroughly tested. 500 Ω load has been tested and seemed fine.
Supply voltage dependence	Unaffected by changes in supply voltage.
Temperature dependence	Not tested
Calibrated voltage to illuminance conversion formula	$$E = 10^{\left(\frac{0.789041-\sqrt{0.487708-0.124232\log_{10}(U)}}{0.062116}\right)}$$ E is the illuminance in lux, U is the voltage in volts. $E \pm 9.4\%$ See figure 31

Table 9: Summarised data: 1k range voltage output

100k range, voltage output	
Parameter	**Value**
No adequate measurements	

Table 10: Summarised data: 100k range voltage output

Current output	
Parameter	**Value**
Load burden voltage dependence	Not tested
Calibrated	Only roughly
Current to voltage conversion formula	$$U = (I - 4\,mA) \cdot \frac{10\,V}{16\,mA}$$
Linearity	No imperfections observed.
Output current range	Approximately 4 - 20 mA normally. Max 25 mA at 12 V voltage output.
Supply voltage dependence	Unaffected by changes in supply voltage
Temperature dependence	Not tested

Table 11: Summarised data: current output

Digital signals (24V)	
Parameter	**Value**
Input: `range_select`	Works. Connect to supply voltage for logic 1, leave unconnected or connect to ground for logic 0. Intermediary voltages not tested.
Output: `over_range`	May oscillate at high audible frequencies (1-20 kHz). Can be mitigated with a capacitor to ground as the digital outputs can only source current but not sink. Only tested with “light” loads.
Output: $\overline{\texttt{digital}}$	Works. Hysteresis has not been formally measured, max hysteresis is subjectively low. Only tested with “light” loads.

Table 12: Summarised data: digital signals

6 References

References

[1] *Adding a resistor to reduce crossover distortion in an LM324/LM358*. Electronics StackExchange. URL: https://electronics.stackexchange.com/questions/341843/adding-a-resistor-to-reduce-crossover-distortion-in-an-lm324-lm358.

[2] Token Electronics. *PGM CdS Photoresistors*. datasheet. URL: https://www.starelec.fi/UserFiles/File/PDF-liitteet2/PGM-TOKEN.pdf.

[3] Texas Instruments. *Application Design Guidelines for LM324/LM358 Devices*. TI Application Report. URL: https://www.ti.com/lit/an/sloa277/sloa277.pdf#page=17.

[4] Texas Instruments. *LMx58-N Low-Power, Dual-Operational Amplifiers*. TI Datasheet. URL: https://www.ti.com/lit/ds/symlink/lm158-n.pdf.

[5] *LM358 from LT? (crossreference for an LTSpice simulation)*. EEVBlog forum. URL: https://www.eevblog.com/forum/beginners/lm358-from-lt-(crossreference-for-an-ltspice-simulation)/.

[6] Delta OHM. *HD2102.1, HD2102.2 PHOTO-RADIOMETERS*. datasheet. URL: https://www.deltaohm.com/wp-content/uploads/document/DeltaOHM_HD2102.1_2_datasheet_ENG.pdf.

[7] *Papouch AD4ETH U*. Product page. URL: https://en.papouch.com/ad4eth-ethernet-measurement-module-p4607/.

7 Code listings

- Some of the Python code requires Python 2.7.
- Octave code may use some Octave specific features and will likely require minor modification to run on Matlab.
- Tested on Debian but should work™ on any unix-like operating system. Some things will require modification to run on Windows, probably only the `plot` command in the Modbus TCP program.

Code (or rather this entire project) can be browsed on Gitlab or downloaded (as compressed tape archive). This report is part2.

7.1 Modbus TCP program

Note: It consists of several files. The program should be started from its containing folder. The main program file is `measure_main.py`.

See also notes in the beginning of "Code listings".

7.1.1 `measure_main.py`

Path: `part2/tests/modbus-tcp/measure_main.py`

```
#!/usr/bin/env python2

import code
import os
import pprint
import sys
import time

from modbustcp import ModbusTCP
from measure_common import stats, get_analog, get_analog_stats

from measure_timedomain import record_timedomain
from measure_illum import record_illum
from measure_vsupply import record_vsupply

'''
dataset = {
    label: {
        'type': type,
        'scalars': {
            key: value,
            ...
        },
        'plots': [
            {
                'title': title,
                'x-axis': column_name,
                'y-axes': [column_name, ...],
                'xlabel': xlabel,
                'ylabel': ylabel,
                'styles': [style, ...],
            },
            ...
        ],
```

```
        'vectors': {
            column_name: [data, ...],
            ...
        }
    },
    ...
}
'''

def main():
    command_functions = {
        'illum': record_illum,
        'vsupply': record_vsupply,
        'scope': record_timedomain,
        'load': load_dataset,
        'save': save_dataset,
        'exit': prog_exit,
        'rename': data_rename,
        'del': data_del,
        'plot': plot,
        'list': list_labels,
        'scalars': scalars,
        'help': disp_help,
        'realtime': realtime,
        'connect': connect,
        'cat': merge_data,
        'python': python,
        'shell': shell,
    }
    connection = [None]
    dataset = {}
    while True:
        sys.stderr.write('> ')
        sys.stderr.flush()
        try:
            line = sys.stdin.readline()
        except KeyboardInterrupt:
            sys.stderr.write('\n')
            exit(0)
        if line == '':  # EOF
            sys.stderr.write('\n')
            break
        if line == '\n':
            continue
        parts = filter(None, line.rstrip('\n').split(' '))
        command, args = parts[0], parts[1:]
        if command not in command_functions:
            sys.stderr.write('Unknown command\n')
            continue
        #command_functions[command](dataset, connection, *args)
        try:
            command_functions[command](dataset, connection, *args)
        except TypeError:
            sys.stderr.write('Bad number of arguments\n')

def disp_help(dataset, connection):
    print('''
connect <IP> [<port>]           Connect to AD4ETH
realtime                        Continuosly read analog inputs twice a second
load <filename>                 Load data from file
```

```
save <filename>                 Store data to file
illum <label>                   Enter illuminance vs output measuring
vsupply <label> <lux> ...       Enter supply voltage dependence measuring
scope <label> <lux> ...         Measure signals for 1 second
plot <label> [<filename>]       Display or save an Octave plot of <label>
list                            List labels in loaded data
scalars <label>                 Print scalar values belonging to object
rename <old> <new>              Rename label
del <label>                     Delete data by label
cat <label> ... output <label>  Merge data to new label
python                          Interactive Python to deal with stuff manually
shell [command]                 Launch a shell or run a command
help                            This message
exit                            Exit without saving

"illum" measuring
    Constant supply voltage, varying illumination
    Enter illuminance values (*) separated by spaces
    Type "done" to finish
    *) The uncertainty of the illuminance values will be recorded as well
       as the variation in output signals over 1 second

"vsupply" measuring
    Constant illumination, varying supply voltage
    Press enter to take another reading
    Type "done" to finish''')

def merge_data(dataset, connection, *args):
    if len(args) < 3:
        sys.stderr.write('Too few arguments\n')
        return
    if args[-2] != 'output':
        sys.stderr.write('Invalid syntax\n')
        return
    output_label = args[-1]
    input_labels = args[:-2]
    first_input = dataset[input_labels[0]]
    data_type = first_input['type']
    for input_label in input_labels:
        if dataset[input_label]['type'] != data_type:
            sys.stderr.write('Mixed types\n')
            return
    #
    dest = {}
    dest['type'] = data_type
    # Scalars
    dest['scalars'] = {}
    for scalar in first_input['scalars']:
        # Is it stats?
        test = first_input['scalars'][scalar]
        is_stats = False
        if isinstance(test, dict):
            items = {'count', 'min', 'max', 'average', 'stddev'}
            if sorted(items) == sorted(test.keys()):
                is_stats = True
        if not is_stats:
            sys.stderr.write('Warning: Unkown scalar: {}\n'.format(scalar))
            dest['scalars'][scalar] = eval(repr(test))
        else:
            count = 0
```

```
            total = 0
            sum_sqr_err = 0
            global_min = test['min']
            global_max = test['max']
            for input_label in input_labels:
                input_stats = dataset[input_label]['scalars'][scalar]
                this_count = input_stats['count']
                count += this_count
                total += this_count * input_stats['average']
                sum_sqr_err += this_count * input_stats['stddev']**2
                global_min = min(global_min, input_stats['min'])
                global_max = min(global_min, input_stats['min'])
            output_stats = {}
            output_stats['count'] = count
            output_stats['average'] = total/count
            output_stats['stddev'] = (sum_sqr_err/count)**.5
            output_stats['min'] = global_min
            output_stats['max'] = global_max
            dest['scalars'][scalar] = output_stats
    # Plots
    sys.stderr.write('Notice: Using plot data from first data\n')
    dest['plots'] = eval(repr(first_input['plots']))
    # Vectors
    dest['vectors'] = {}
    for vector in first_input['vectors']:
        dest['vectors'][vector] = []
        for input_label in input_labels:
            dest['vectors'][vector] += dataset[input_label]['vectors'][vector]
    # Save
    dataset[output_label] = dest

def python(dataset, connection):
    code.interact(local=locals())

def shell(dataset, connection, *args):
    if args:
        os.system(' '.join(args))
    else:
        os.system('$SHELL || sh')

def connect(dataset, connection, address, port="502"):
    try:
        connection[0] = ModbusTCP(address, int(port))
    except:
        sys.stderr.write('Connection failed\n')

def realtime(dataset, connection):
    try:
        while True:
            Vout, Iout, Vs = get_analog(connection)
            print('Vout: {:.3f} V\tIout: {:.3f} mA\tVs: {:.1f} V'.format(
                Vout, Iout, Vs))
            time.sleep(0.5)
    except KeyboardInterrupt:
        sys.stderr.write('\n')
        return

def list_labels(dataset, connection):
    sys.stderr.write('{:50} Type\n{:50} ----\n'.format('Name', '----'))
    for label in sorted(dataset.keys()):
```

```
        sys.stdout.write('{:50} {}\n'.format(label, dataset[label]['type']))

def scalars(dataset, connection, label):
    try:
        print(pprint.pformat(dataset[label]['scalars']))
    except KeyError:
        sys.stderr.write('No such object: {}.\n'.format(repr(label)))

def plot(dataset, connection, label, *args):
    def matlab(vector):
        '''Convert Python list of floats to Matlab horizontal vector'''
        return '[' + ' '.join(map(str, vector)) + ']'
    try:
        data = dataset[label]
    except KeyError:
        sys.stderr.write('No such object: {}.\n'.format(repr(label)))
        return
    if len(args) > 1:
        sys.stderr.write('Too many arguments')
        return
    elif len(args) == 1:
        filename = args[0]
    else:
        filename = 'tmp-plot'
    try:
        f = open(filename, 'wx')     # Open file only if it doesn't exist
    except:
        sys.stderr.write('Error opening file for writing.\n')
        return
    f.write('#!/usr/bin/env octave\n')
    for fig_index, plot in enumerate(data['plots']):
        f.write('fig{} = figure;\n'.format(fig_index))
        f.write('hold on;\n')
        f.write('[x{0}, x{0}_order] = sort({1});\n'.format(
            fig_index,
            matlab(data['vectors'][plot['x-axis']])
        ))
        for y_index, y_name in enumerate(plot['y-axes']):
            f.write('y{0}_{1} = {2}(x{0}_order);\n'.format(
                fig_index, y_index,
                matlab(data['vectors'][y_name])
            ))
            f.write('plot(x{0}, y{0}_{1}, {2});\n'.format(
                fig_index, y_index, repr(plot['styles'][y_index])
            ))
        f.write('title({});\n'.format(repr(plot['title']).replace(r'\\','\\')))
        f.write('xlabel({});\n'.format(repr(plot['xlabel'])))
        f.write('ylabel({});\n'.format(repr(plot['ylabel'])))
        f.write('grid on;\nhold off;\n')
    for i in range(len(data['plots'])):
        f.write('waitfor(fig{});\n'.format(i))
    f.close()
    os.chmod(filename, 0o755)
    if len(args) == 0:
        os.spawnl(os.P_NOWAIT, filename, filename)
        # It appears that without the delay the script is removed before Octave
        # gets to opening it.
        time.sleep(2)
        os.unlink(filename)
```

```
def load_dataset(dataset, connection, filename):
    try:
        new_dataset = eval(open(filename).read())
    except:
        sys.stderr.write('Load failed\n')
        return
    dataset.clear()
    for key in list(new_dataset):
        dataset[key] = new_dataset[key]
        del new_dataset[key]

def save_dataset(dataset, connection, filename):
    try:
        f = open(filename, 'w')
        f.write(pprint.pformat(dataset))
        f.close()
    except:
        sys.stderr.write('Save failed')

def prog_exit(dataset, connection):
    exit(0)

def data_rename(dataset, connection, old, new):
    if new in dataset:
        sys.stderr.write('Destination label exists\n')
    try:
        dataset[new] = dataset[old]
        del dataset[old]
    except KeyError:
        sys.stderr.write('No such object: {}.\n'.format(repr(old)))

def data_del(dataset, connection, label):
    try:
        del dataset[label]
    except KeyError:
        sys.stderr.write('No such object: {}.\n'.format(repr(label)))


if __name__ == '__main__':
    main()
```

7.1.2 measure_common.py

Path: `part2/tests/modbus-tcp/measure_common.py`

```
import time

def stats(data, old=None):
    data = map(float, data)
    if old is None:
        old = {
            'min': float("inf"),
            'max': float("-inf"),
            'count': 0,
            'average': 0.0,
            'stddev': 0.0,
        }
    result = {}
    result['min'] = min(min(data), old['min'])
    result['max'] = max(max(data), old['max'])
```

```
    count = len(data) + old['count']
    result['count'] = count
    new_sum_data = sum(data)
    old_sum_data = old['average'] * old['count']
    avg = (new_sum_data+old_sum_data) / count
    result['average'] = avg
    new_sum_sqr_err = sum([(x-avg)**2 for x in data])
    old_sum_sqr_err = old['stddev']**2 * old['count']
    result['stddev'] = ((new_sum_sqr_err+old_sum_sqr_err)/count)**.5
    return result

def get_analog(connection):
    '''
    Vout, Iout, Vs
    '''
    # TODO
    #for i in range(1000):
    #    i**i
    #return (0.5*time.time())%10, (0.5*time.time())%16+4, (0.25*time.time())%10+19
    # Real code here
    response = connection[0].request([
        0xff,        # Unit ID
        0x04,        # Read input register(s)
        0x00, 0x00, # Start address 0 for channel 1
        0x00, 0x0c   # 12 registers (3 channels * 4 registers per channel)
    ])
    # response[0]                # Unit ID echo
    assert response[1] == 0x04   # Function code echo, high bit indicates error
    assert response[2] == 0x18   # 24 bytes are to be returned
    data = response[3:]
    Vout = (data[2]*256 + data[3]) / 1000.0
    Iout = (data[10]*256 + data[11]) / 500.0
    Vs = 3 * (data[18]*256 + data[19]) / 1000.0
    return Vout, Iout, Vs

def get_analog_stats(connection):
    Vout_array = []
    Iout_array = []
    Vs_array = []
    start_t = time.time()
    while True:
        Vout, Iout, Vs = get_analog(connection)
        timestamp = time.time()
        Vout_array.append(Vout)
        Iout_array.append(Iout)
        Vs_array.append(Vs)
        if timestamp - start_t > 1:
            break
    return stats(Vout_array), stats(Iout_array), stats(Vs_array)
```

7.1.3 measure_illum.py

Path: `part2/tests/modbus-tcp/measure_illum.py`

```
import pprint
import sys

from measure_common import stats, get_analog, get_analog_stats

default_illum = {
```

```
    'type': 'illum',
    'plots': [
        {
            'title': None,
            'xlabel': 'Illuminance (Lux)',
            'ylabel': 'V_{out} (V) [stddev, min, average, max]',
            'x-axis': 'illuminance-average',
            'y-axes': ['Vout-stddev', 'Vout-min', 'Vout-average', 'Vout-max'],
            'styles': ['-s', 's', '-s', 's']
        },
        {
            'title': None,
            'xlabel': 'Illuminance (Lux)',
            'ylabel': 'I_{out} (mA) [stddev, min, average, max]',
            'x-axis': 'illuminance-average',
            'y-axes': ['Iout-stddev', 'Iout-min', 'Iout-average', 'Iout-max'],
            'styles': ['-s', 's', '-s', 's']
        },
        {
            'title': None,
            'xlabel': 'V_{out} (V)',
            'ylabel': 'Illuminance (Lux) [stddev, min, average, max]',
            'x-axis': 'Vout-average',
            'y-axes': ['illuminance-stddev', 'illuminance-min',
                       'illuminance-average', 'illuminance-max'],
            'styles': ['-s', 's', '-s', 's']
        },
        {
            'title': None,
            'xlabel': 'I_{out} (mA)',
            'ylabel': 'Illuminance (Lux) [stddev, min, average, max]',
            'x-axis': 'Iout-average',
            'y-axes': ['illuminance-stddev', 'illuminance-min',
                       'illuminance-average', 'illuminance-max'],
            'styles': ['-s', 's', '-s', 's']
        },
        {
            'title': None,
            'xlabel': 'U_{out} (V)',
            'ylabel': 'I_{out} (mA)',
            'x-axis': 'Vout-average',
            'y-axes': ['Iout-average'],
            'styles': ['-s']
        }
    ],
    'scalars': {
        'Vs': None
    },
    'vectors': {
        'illuminance-stddev': [],
        'illuminance-min': [],
        'illuminance-average': [],
        'illuminance-max': [],
        'Vout-stddev': [],
        'Vout-min': [],
        'Vout-average': [],
        'Vout-max': [],
        'Iout-stddev': [],
        'Iout-min': [],
        'Iout-average': [],
```

```
        'Iout-max': [],
    }
}

def record_illum(dataset, connection, label):
    '''
    '''
    if label in dataset:
        if dataset[label]['type'] != 'illum':
            sys.stderr.write('Incompatible\n')
            return
    if label not in dataset:
        dataset[label] = default_illum.copy()
    # Load data
    data = eval(pprint.pformat(dataset[label]))
    # Always update plot info
    data['plots'] = default_illum['plots'][:]
    # Update scalars and plot titles
    scalars = data['scalars']
    # TODO: Vs is replaced by the latest run
    #scalars['Vs'] = stats(get_analog_stats(connection)[2], scalars['Vs'])
    scalars['Vs'] = get_analog_stats(connection)[2]
    Vs = 'V_s is {:.1f} \\pm {:.2f} V'.format(
        scalars['Vs']['average'],
        scalars['Vs']['stddev']
    )
    data['plots'][0]['title'] = 'Illuminance/V_{out}, ' + Vs
    data['plots'][1]['title'] = 'Illuminance/I_{out}, ' + Vs
    data['plots'][2]['title'] = 'V_{out}/Illuminance, ' + Vs
    data['plots'][3]['title'] = 'I_{out}/Illuminance, ' + Vs
    data['plots'][4]['title'] = 'V_{out}/I_{out}, ' + Vs
    # Sample new data
    while True:
        sys.stderr.write('>> ')
        sys.stderr.flush()
        line = sys.stdin.readline()
        if line == 'done\n':
            break
        try:
            illuminance = stats(line.rstrip('\n').split(' '))
        except:
            sys.stderr.write('Bad input\n')
            continue
        Vout, Iout, _ = get_analog_stats(connection)
        for key in ('stddev', 'min', 'average', 'max'):
            data['vectors']['illuminance-'+key].append(illuminance[key])
            data['vectors']['Vout-'+key].append(Vout[key])
            data['vectors']['Iout-'+key].append(Iout[key])
    # Store data
    dataset[label] = data
```

7.1.4 measure_vsupply.py

Path: `part2/tests/modbus-tcp/measure_vsupply.py`

```
import pprint
import sys

from measure_common import stats, get_analog, get_analog_stats

```

```python
default_vsupply = {
    'type': 'vsupply',
    'plots': [
        {
            'title': None,
            'xlabel': 'Supply voltage (V)',
            'ylabel': 'V_{out} (V) [stddev, min, average, max]',
            'x-axis': 'Vs',
            'y-axes': ['Vout-stddev', 'Vout-min', 'Vout-average', 'Vout-max'],
            'styles': ['-s', 's', '-s', 's']
        },
        {
            'title': None,
            'xlabel': 'Supply voltage (V)',
            'ylabel': 'I_{out} (mA) [stddev, min, average, max]',
            'x-axis': 'Vs',
            'y-axes': ['Iout-stddev', 'Iout-min', 'Iout-average', 'Iout-max'],
            'styles': ['-s', 's', '-s', 's']
        },
    ],
    'scalars': {
        'illuminance': None
    },
    'vectors': {
        'Vs': [],
        'Vout-stddev': [],
        'Vout-min': [],
        'Vout-average': [],
        'Vout-max': [],
        'Iout-stddev': [],
        'Iout-min': [],
        'Iout-average': [],
        'Iout-max': [],
    }
}

def record_vsupply(dataset, connection, label, *illuminance):
    '''
    '''
    if label in dataset:
        if dataset[label]['type'] != 'vsupply':
            sys.stderr.write('Incompatible\n')
            return
    if label not in dataset:
        dataset[label] = default_vsupply.copy()
    # Load data
    data = eval(pprint.pformat(dataset[label]))
    # Always update plot info
    data['plots'] = default_vsupply['plots'][:]
    # Update scalars and plot titles
    scalars = data['scalars']
    scalars['illuminance'] = stats(illuminance, scalars['illuminance'])
    illum = 'at {:.0f} \\pm {:.0f} Lux'.format(
        scalars['illuminance']['average'],
        scalars['illuminance']['stddev']
    )
    data['plots'][0]['title'] = 'Vout {}'.format(illum)
    data['plots'][1]['title'] = 'Iout {}'.format(illum)
    # Sample new data
    while True:
```

```
        sys.stderr.write('>> ')
        sys.stderr.flush()
        if sys.stdin.readline() == 'done\n':
            break
        Vout, Iout, Vs = get_analog_stats(connection)
        data['vectors']['Vs'].append(Vs['average'])
        for key in ('stddev', 'min', 'average', 'max'):
            data['vectors']['Vout-'+key].append(Vout[key])
            data['vectors']['Iout-'+key].append(Iout[key])
    # Store data
    dataset[label] = data
```

7.1.5 measure_timedomain.py

Path: `part2/tests/modbus-tcp/measure_timedomain.py`

```
import pprint
import time

from measure_common import stats, get_analog, get_analog_stats

default_timedomain = {
    'type': 'timedomain',
    'plots': [
        {
            'title': None,
            'xlabel': 'time (ms)',
            'ylabel': 'V_{out} (V)',
            'x-axis': 'time',
            'y-axes': ['Vout'],
            'styles': ['-s']
        },
        {
            'title': None,
            'xlabel': 'time (ms)',
            'ylabel': 'I_{out} (mA)',
            'x-axis': 'time',
            'y-axes': ['Iout'],
            'styles': ['-s']
        },
        {
            'title': None,
            'xlabel': 'time (ms)',
            'ylabel': 'V_s (V)',
            'x-axis': 'time',
            'y-axes': ['Vs'],
            'styles': ['-s']
        },
    ],
    'scalars': {
        'illuminance': None
    },
    'vectors': {
        'time': [],
        'Vout': [],
        'Iout': [],
        'Vs': [],
    }
}

```

```
def record_timedomain(dataset, connection, label, *illuminance):
    '''
    '''
    if label in dataset:
        if dataset[label]['type'] != 'timedomain':
            sys.stderr.write('Incompatible\n')
            return
    if label not in dataset:
        dataset[label] = default_timedomain.copy()
    # Load data
    data = eval(pprint.pformat(dataset[label]))
    # Always update plot info
    data['plots'] = default_timedomain['plots'][:]
    # Update scalars and plot titles
    scalars = data['scalars']
    scalars['illuminance'] = stats(illuminance, scalars['illuminance'])
    illum = 'at {:.0f} \\pm {:.0f} Lux'.format(
        scalars['illuminance']['average'],
        scalars['illuminance']['stddev']
    )
    data['plots'][0]['title'] = 'Vout {}'.format(illum)
    data['plots'][1]['title'] = 'Iout {}'.format(illum)
    data['plots'][2]['title'] = 'Vs {}'.format(illum)
    # Sample new data
    start_t = time.time()
    t = 0
    while t < 1:
        Vout, Iout, Vs = get_analog(connection)
        t = time.time() - start_t
        data['vectors']['Vout'].append(Vout)
        data['vectors']['Iout'].append(Iout)
        data['vectors']['Vs'].append(Vs)
        data['vectors']['time'].append(t*1000)
    # Store data
    dataset[label] = data
```

7.1.6 modbustcp.py

Path: `part2/tests/modbus-tcp/modbustcp.py`

```
# Python 2 only (at the moment)

import socket

class ModbusTCP():
    '''
    This is a synchronous Modbus TCP client.

    foo = ModbusTCP(address, port=502)
    response = foo.request([unit_id, function, data_bytes ...])
    '''
    def __init__(self, address, port=502):
        self.conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.conn.connect((address, port))
        self.sequence_id = 0

    def request(self, modbus_request):
        '''
        modbus_request = [
            unit_identifier,
```

```
            function_code,
            data_bytes ...
        ]
        '''
        datalen = len(modbus_request)
        modbus_tcp_request_packet = [
            (self.sequence_id >> 8) & 0xff, # Transaction identifier, high
            self.sequence_id & 0xff,        # Transaction identifier, low
            0, 0,                           # Protocol identifier, high & low
            (datalen >> 8) & 0xff,          # Length field, high
            datalen & 0xff,                 # Length field, low
        ]
        modbus_tcp_request_packet += modbus_request
        # send of the request
        self.conn.sendall(''.join(map(chr, modbus_tcp_request_packet)))
        # Response
        response_head = map(ord, self.conn.recv(6))
        response_transaction = response_head[0]*256 + response_head[1]
        response_protocol = response_head[2]*256 + response_head[3]
        response_length = response_head[4]*256 + response_head[5]
        assert response_transaction == self.sequence_id
        assert response_protocol == 0
        # Get the Modbus part of the response
        self.sequence_id += 1
        return map(ord, self.conn.recv(response_length))

    def __del__(self):
        self.conn.close()
```

7.2 Regression analysis program

Note: The program should be started from its containing folder. It needs the `*.m` files which contain the measurement data.

See also notes in the beginning of "Code listings".

Path: `part2/tests/regression-analysis/regression`

```
#!/usr/bin/env octave

# Needs package "optim"
pkg load optim

% Make regressions and plots for E-U data
% DATA: 0-1000 and 0-100000
% PLOTS: lin-lin, log-log, gamma
% LINEAR LEAST SQUARES: E^gamma + zero_offset, E^gamma
% LOGARITHMIC LEAST SQUARES: 1st-4th power polynomials


% Font size:          22      8
% Scatter ball size: 49      25

function [out_E, out_U] = logsafe(E, U)
    out_E = [];
    out_U = [];
    for i = 1:size(E)(2)
        if E(i) > 0 && U(i) > 0
            out_E = [out_E E(i)];
            out_U = [out_U U(i)];
```

```
        end
    end
end


function myplot(E, U, interp_arr_E, interp_arr_U, E_range, plot_title)
    style = {"k" "r" "b" "--k" "--r" "--b"};
    % solid black, solid red, solid blue, dashed black, dashed red, dashed blue

    % LINEAR
    f = figure('DefaultAxesFontSize', 8);
    hold on;
    %plot(E, U, 'dm');
    scatter(E, U, 25, 'm', 'MarkerFaceColor', 'm');
    for i = 1:size(interp_arr_E)(2)
        interp_E = interp_arr_E{1,i};
        interp_U = interp_arr_U{1,i};
        plot(interp_E, interp_U, style{1,i}, "linewidth", 2)
    end
    axis([0 1.2*10^E_range 0 12]);
    if E_range < 4
        xticks([0:5*10^(E_range-2):1.2*10^E_range]);
    else
        xticks([0:10*10^(E_range-2):1.2*10^E_range]);
    end
    yticks([0:0.5:12]);
    xlabel('Illuminance E [Lx]');
    ylabel('Voltage U [V]');
    title(['Lin-lin ' plot_title]);
    grid on;
    hold off;

    % LOGARITHMIC
    f2 = figure('DefaultAxesFontSize', 8);
    hold on;
    scatter(log10(E), log10(U), 25, 'm', 'MarkerFaceColor', 'm');
    %plot(log10(E), log10(U), 'dm');
    for i = 1:size(interp_arr_E)(2)
        interp_E = interp_arr_E{1,i};
        interp_U = interp_arr_U{1,i};
        plot(log10(interp_E), log10(interp_U), style{1,i}, "linewidth", 2)
    end
    axis([-1 E_range+0.1 -2 1.33]);
    xticks([-1:0.25:E_range]);
    yticks([-2:0.25:1.25]);
    %xlabel('Illuminance lg(E) (log_{10}(Lx))');
    %ylabel('Voltage lg(U) (log_{10}(V))');
    xlabel('Log illuminance [log_{10}(Lx)]');
    ylabel('Log voltage [log_{10}(V)]');
    title(['Log-log ' plot_title]);
    grid on;
    hold off;

    % GAMMA
    f3 = figure('DefaultAxesFontSize', 8);
    hold on;
    for i = 1:size(interp_arr_E)(2)
        [interp_E, interp_U] = logsafe(interp_arr_E{1,i}, interp_arr_U{1,i});
        interp_gamma = diff(log10(interp_U))./diff(log10(interp_E));
        plot(log10(interp_E)(1:end-1), interp_gamma, style{1,i}, "linewidth", 2)
```

```
    end
    axis([-1 E_range+1 0.2 1]);
    xticks([-1:0.5:E_range+1]);
    yticks([0.2:0.05:1]);
    %xlabel('Illuminance decade lg(E) (log_{10}(Lx))');
    %ylabel('Gamma \gamma (-log(\Omega)/log(Lx))');
    xlabel('Log illuminance [log_{10}(Lx)]');
    ylabel('Gamma \gamma [-log(\Omega)/log(Lx)]');
    title(['Variable gamma ' plot_title]);
    grid on;
    hold off;

    waitfor(f);
    waitfor(f2);
    waitfor(f3);
end


function interpolate(E, U, data_name, is_100k)
    plot_arr_E = {};
    plot_arr_U = {};

    if is_100k
        plot_name = [data_name ' (0 to 100 000 lux)'];
    else
        plot_name = [data_name ' (0 to 1 000 lux)'];
    end
    printf("%s\n", plot_name);

    % Excess is needed for the gamma plot
    % E range for linear interpolation
    if is_100k
        interp_E = [[0:0.01:9.99] [10:0.1:99.9] [100:1:999] [1000:10:9990] ...
                    [10000:100:99900] [10^5:10^3:10^6]];
    else
        interp_E = [[0:0.01:9.99] [10:0.1:99.9] [100:1:999] [1000:10:10000]];
    end

    if is_100k
        offset = 0.0157;
    else
        offset = 0.1298;
    end

    % TRACE 1
    func = @(E, param) param(1) * E.^param(2) + offset;
    pin = [0.1 0.5];
    [_f1, param,  _kvg1, _iter1, _corp1, _covp1, _covr1, ...
            _stdresid1, _Z1, r2] = leasqr(E, U, pin, func);
    interp_U = func(interp_E, param);
    printf("[SOLID BLACK]\tlin with offset:\tr^2 is %f, Gamma is %f\n",
            r2, param(2))
    printf("\t%f\t%f\t%f\n", param(1), param(2), offset)
    plot_arr_E = {interp_E};
    plot_arr_U = {interp_U};

    % TRACE 2
    func = @(E, param) param(1) * E.^param(2);
    [_f1, param,  _kvg1, _iter1, _corp1, _covp1, _covr1, ...
            _stdresid1, _Z1, r2]  = leasqr(E, U, pin, func);
```

```
    interp_U = func(interp_E, param);
    printf("[SOLID RED]\tlin without offset:\tr^2 is %f, Gamma is %f\n",
            r2, param(2))
    printf("\t%f\t%f\n", param(1), param(2))
    plot_arr_E = {plot_arr_E{1,1:end} interp_E};
    plot_arr_U = {plot_arr_U{1,1:end} interp_U};

    % Excess is needed for the gamma plot
    % E range for logarithmic interpolation
    [E, U] = logsafe(E, U);
    if is_100k
        interp_E = [-1:0.01:6];
    else
        interp_E = [-1:0.01:4];
    end

    % TRACE 3
    func = @(logE, param) param(1) + param(2)*logE;
    pin = [0.1 0.5];
    [_f1, param,  _kvg1, _iter1, _corp1, _covp1, _covr1, ...
            _stdresid1, _Z1, r2]  = leasqr(log10(E), log10(U), pin, func);
    interp_U = func(interp_E, param);
    printf("[SOLID BLUE]\tlog 1st:\t\tr^2 is %f, Gamma is %f\n", r2, param(2))
    printf("\t%f\t%f\n", param(1), param(2))
    plot_arr_E = {plot_arr_E{1,1:end} 10.^interp_E};
    plot_arr_U = {plot_arr_U{1,1:end} 10.^interp_U};

    % TRACE 4
    func = @(logE, param) param(1) + param(2)*logE + param(3)*logE.^2;
    pin = [0.1 0.5 0];
    [_f1, param,  _kvg1, _iter1, _corp1, _covp1, _covr1, ...
            _stdresid1, _Z1, r2]  = leasqr(log10(E), log10(U), pin, func);
    interp_U = func(interp_E, param);
    printf("[DASHED BLACK]\tlog 2nd:\t\t")
    printf("r^2 is %f, Gamma is [%f + (%f * decade)]\n",
            r2, param(2), 2*param(3))
    printf("\t%f\t%f\t%f\n", param(1), param(2), param(3))
    plot_arr_E = {plot_arr_E{1,1:end} 10.^interp_E};
    plot_arr_U = {plot_arr_U{1,1:end} 10.^interp_U};

    % TRACE 5
    func = @(logE, param) param(1) + param(2)*logE + param(3)*logE.^2 ...
            + param(4)*logE.^3;
    pin = [0.1 0.5 0 0];
    [_f1, param,  _kvg1, _iter1, _corp1, _covp1, _covr1, ...
            _stdresid1, _Z1, r2]  = leasqr(log10(E), log10(U), pin, func);
    interp_U = func(interp_E, param);
    printf("[DASHED RED]\tlog 3rd:\t\t")
    printf("r^2 is %f, Gamma is [%f + (%f * decade) + (%f * decade^2)]\n",
            r2, param(2), 2*param(3), 3*param(4))
    printf("\t%f\t%f\t%f\t%f\n", param(1), param(2), param(3), param(4))
    plot_arr_E = {plot_arr_E{1,1:end} 10.^interp_E};
    plot_arr_U = {plot_arr_U{1,1:end} 10.^interp_U};

    % TRACE 6
    func = @(logE, param) param(1) + param(2)*logE + param(3)*logE.^2 ...
            + param(4)*logE.^3 + param(5)*logE.^4;
    pin = [0.1 0.5 0 0 0];
    [_f1, param,  _kvg1, _iter1, _corp1, _covp1, _covr1, ...
            _stdresid1, _Z1, r2]  = leasqr(log10(E), log10(U), pin, func);
```

```
    interp_U = func(interp_E, param);
    printf("[DASHED BLUE]\tlog 4th:\t\tr^2 is %f, Gamma is ", r2)
    printf("[%f + (%f * decade) + (%f * decade^2) + (%f * decade^3)]\n",
        param(2), 2*param(3), 3*param(4), 4*param(5))
    printf("\t%f\t%f\t%f\t%f\t%f\n", param(1), param(2), param(3), param(4),
        param(5))
    plot_arr_E = {plot_arr_E{1,1:end} 10.^interp_E};
    plot_arr_U = {plot_arr_U{1,1:end} 10.^interp_U};

    % Display traces
    if is_100k
        myplot(E, U, plot_arr_E, plot_arr_U, 5, plot_name);
    else
        myplot(E, U, plot_arr_E, plot_arr_U, 3, plot_name);
    end
    printf("\n");
end

function myscatter(E, U, plot_title, is_100k)
    if is_100k
        E_range = 5;
    else
        E_range = 3;
    end

    linfig = figure('DefaultAxesFontSize', 8);
    hold on;
    scatter(E, U, 25, 'k', 'MarkerFaceColor', 'k');
    axis([0 1.1*10^E_range 0 10.5]);
    if E_range < 4
        xticks([0:5*10^(E_range-2):1.1*10^E_range]);
    else
        xticks([0:10*10^(E_range-2):1.1*10^E_range]);
    end
    yticks([0:0.5:10.5]);
    xlabel('Illuminance E (Lx)');
    ylabel('Voltage U (V)');
    title([plot_title ' lin-lin']);
    grid on;
    hold off;

    logfig = figure('DefaultAxesFontSize', 8);
    hold on;
    scatter(log10(E), log10(U), 25, 'k', 'MarkerFaceColor', 'k');
    axis([-0.1 E_range+0.1 -2 1.1]);
    xticks([0:0.5:E_range]);
    yticks([-2:0.5:1]);
    xlabel('Log illuminance [log_{10}(Lx)]');
    ylabel('Log voltage [log_{10}(V)]');
    title([plot_title ' log-log']);
    grid on;
    hold off;

    waitfor(linfig);
    waitfor(logfig);
end


%set(0, "defaultaxesfontname", "Helvetica")
graphics_toolkit("fltk") %("gnuplot")
```

```

[E, U] = data_1k();
%myscatter(E, U, '1 000 Lux (with phone)', false);
interpolate(E, U, '1 000 Lux (with phone)', false);

[E, U] = data_1k_ad4eth();
interpolate(E, U, '1000 Lux (AD4ETH)', false);

[E, U] = data_100k();
%myscatter(E, U, '100 000 Lux (with phone)', true);
interpolate(E, U, '100 000 Lux (with phone)', true);

[E, U] = data_100k_ad4eth();
interpolate(E, U, '100 000 Lux (AD4ETH)', true);
```

7.2.1 Accuracy measuring program

Note: The program should be started from its containing folder. It needs one of the `*.m` files which contains the measurement data.

See also notes in the beginning of "Code listings".

Path: `part2/tests/regression-analysis/ad4eth-1k-accuracy`

```
#!/usr/bin/env octave

function pass_rate = test_tolerance(E_points, U_points, solution, tolerance)
    passes = 0;
    fails = 0;
    for i = 1:size(U_points)(2)
        if U_points(i) <= 0
            continue
        end
        calculated_E = solution(U_points(i));
        calculated_error = E_points(i)/calculated_E;
        if calculated_error > 1+tolerance || calculated_error < 1-tolerance
            fails += 1;
        else
            passes += 1;
        end
    end
    pass_rate = passes/(passes+fails);
end

function passrate_vector = tolerance_to_passrate(E, U, sol, tolerance_vector)
    passrate_vector = [];
    for i = 1:size(tolerance_vector)(2)
        passrate_vector = [passrate_vector test_tolerance(E,U,sol, ...
                                            tolerance_vector(i))];
    end
end

function tolerance = find_tolerance(E,U,sol,target_passrate)
    tolerance = 0;
    while test_tolerance(E,U,sol, tolerance) < target_passrate
        tolerance += 0.001;
    end
end

[E_points, U_points] = data_1k_ad4eth();
```

```
solution = @(U) 10.^((0.789041-sqrt(0.487708-0.124232*log10(U)))/0.062116);

tolerance = find_tolerance(E_points,U_points,solution, 0.90);
printf("Tolerance: %.1f%%\n", tolerance*100)

U_range = [0.01:0.01:11];

tolerance_vector = [0:0.001:0.15];
passrate_vector = tolerance_to_passrate(E_points,U_points,solution, ...
                                        tolerance_vector);

fig = figure;
hold on
plot(U_points, E_points, "*")
plot(U_range, solution(U_range), "-")
plot(U_range, (1-tolerance)*solution(U_range), "-")
plot(U_range, (1+tolerance)*solution(U_range), "-")
xlabel("Voltage (V)")
ylabel("Illuminance (Lx)")
title(sprintf("Voltage to illuminance, with %.1f%% tolerance (1k AD4ETH)", ...
              100*tolerance))
grid on
axis([0 11 0 1200])
xticks([0:1:11])
yticks([0:100:1200])
hold off

fig2 = figure;
hold on
plot(100*tolerance_vector, 100*passrate_vector, "-")
xlabel("Tolerance \\pm%")
ylabel("Points within range (%)")
title("Tolerance vs points in area (1k AD4ETH)")
grid on
axis([0 15 0 100])
xticks([0:1:15])
yticks([0:10:100])
hold off

waitfor(fig)
waitfor(fig2)
```