

Illuminance sensor, part 2: construction and calibration

Oskar Skog

October 20, 2021

Contents

1	Introduction	4
2	Construction	4
2.1	PCB manufacturing	4
2.2	Cherry picked resistances for fixed dividers	5
2.2.1	Error in differential amplifier (IC3a)	5
2.2.2	Error in voltage to current conversion	6
2.3	Soldering	7
2.3.1	Potentiometers not fitting	7
2.3.2	Heatsink	7
2.3.3	Cleaning	8
2.4	Mounting	8
2.5	Test cables (and LDR)	8
3	Functional testing	10
3.1	Oscillation in the range limiter	10
3.1.1	More accurate simulations	10
3.1.2	Physical tests	10
3.1.3	Fixing the crossover distortion in the op-amps	11
4	Measurements	12
4.1	At home calibration and measurements	13
4.1.1	Equipment	13
4.1.2	Usage of desk lamp & calibration	14
4.1.3	Illuminance vs voltage	15
4.1.4	Time constant	18
4.2	Testing with proper light meter and Modbus TCP DAQ	21
4.3	Regression analysis	26
4.3.1	Repeated theory	26
4.3.2	Data	26
4.3.3	Models	26
4.3.4	Results	27
4.3.5	Reversing the formula and adding tolerance	33
5	Summary and evaluation	35

5.1	Project evaluation	35
5.2	Evaluation of the prototype	35
6	References	38
7	Code listings	39
7.1	Modbus TCP program	39
7.1.1	measure_main.py	39
7.1.2	measure_common.py	44
7.1.3	measure_illum.py	45
7.1.4	measure_vsupply.py	47
7.1.5	measure_timedomain.py	49
7.1.6	modbustcp.py	50
7.2	Regression analysis program	51
7.2.1	Accuracy measuring program	56

List of Figures

1	Newly milled PCB	4
2	Schematic for the differential amplifier	5
3	Schematic for the transconductance amplifier	6
4	Potentiometer fitment issues	7
5	Soldered heatsink and thermal paste pre-applied to TO-220 device	7
6	PCB before and after cleaning	8
7	Adding washers to attempt to compensate for longer screws	8
8	LDR mounted in plastic tube from ballpoint pen	9
9	Two-pin Molex to bare wires for tests and LEDs for digital outputs	9
10	Power cable, with extra connector for <code>range_select</code>	9
11	Crimping process	9
12	With vs without C14	11
13	Before vs after adding resistors to mitigate oscillations	12
14	Modification to attempt to reduce cross-over distortion in op-amp	12
15	Power supply with LC filter	13
16	“Oscilloscope”	14
17	Usage of the modified desk lamp for controlled illumination	14
18	Scatter plots for 1 000 Lux range tested with phone	16
19	Scatter plots for 100 000 Lux range tested with phone	17
20	Measured signal of 5Hz flashes	19
21	Measuring τ on rising edge	20
22	Measuring τ on falling edge	20
23	Measuring U/E curve with proper equipment	22
24	Hysteresis, measured on the third day	23
25	Repeatability	24
26	Measurement accuracy	25
27	Regression analysis for 1k range with phone	28
28	Regression analysis for 1k range with HD2102	29
29	Regression analysis for 100k range with phone	30
30	Regression analysis for 100k range with HD2102	31
31	Accuracy (AD4ETH & HD2102, 1k range)	34
32	Topology	35

List of Tables

1	Resistance values of all 22 $k\Omega$ and 470 $k\Omega$ resistors	5
2	List of test equipment	13
3	Illuminance/voltage data with phone	15
4	Rising time constant	18
5	Falling timeconstant	19
6	Regression analysis summary: typical sensitivity	32
7	Regression analysis summary: Evaluation of models	32
8	Regression analysis summary: Evaluation of datasets	33
9	Summarised data: 1k range voltage output	36
10	Summarised data: 100k range voltage output	36
11	Summarised data: current output	37
12	Summarised data: digital signals	37

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

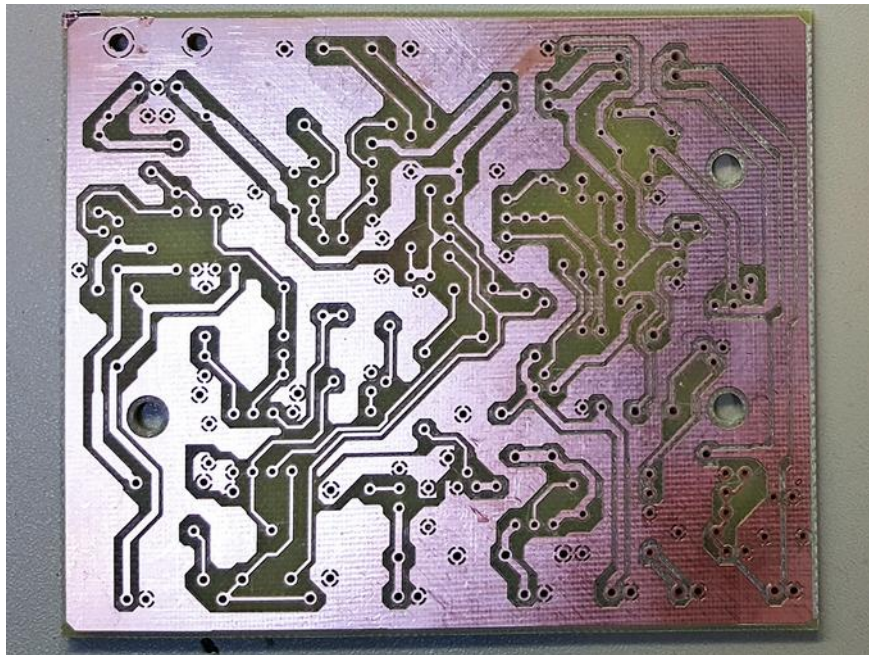
1. Construction of the prototype
2. Testing that it actually works and fixing what needs fixing
3. Calibrating it and testing how well it performs
4. Summary and evaluation

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 kΩ	R9	468.8 kΩ
R8	22.01 kΩ	R10	468.4 kΩ
R16	21.98 kΩ	R14	468.3 kΩ
R17	21.93 kΩ	R15	469.5 kΩ
R20	22.00 kΩ	R18	468.5 kΩ
R21	22.02 kΩ	R19	468.7 kΩ

Table 1: Resistance values of all 22 kΩ and 470 kΩ 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)

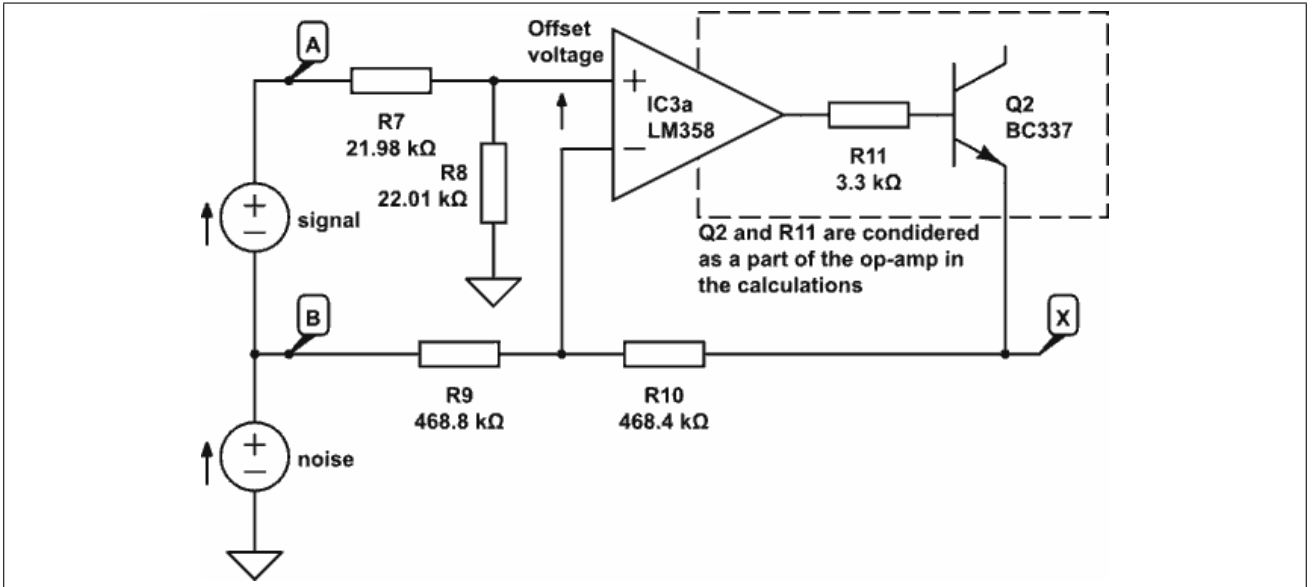


Figure 2: Schematic for the differential amplifier

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

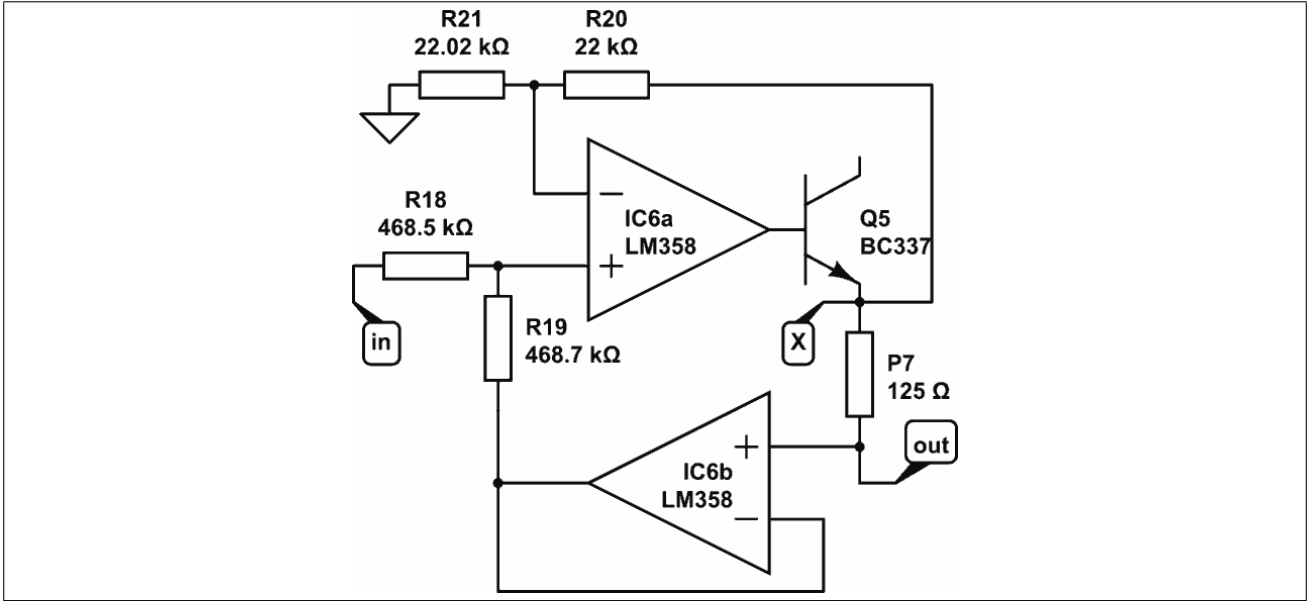


Figure 3: Schematic for the transconductance amplifier

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)

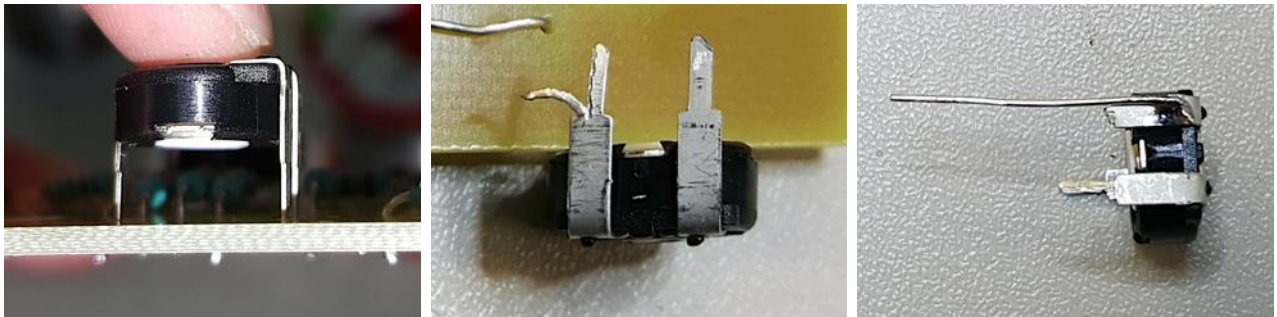
$$\Delta I_{out} = U_{out} \frac{R18 \cdot R20 - R19 \cdot R21}{P7(R18 + R19)R21} = -0.00534 \text{ mA/V}$$

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).

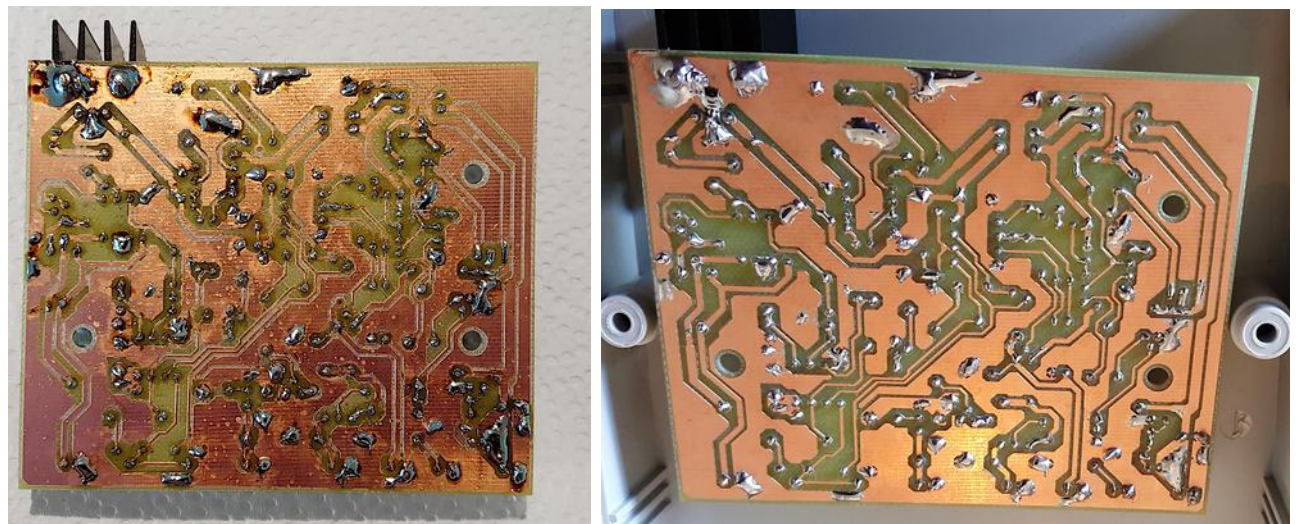


Figure 5: Soldered heatsink and thermal paste pre-applied to TO-220 device

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

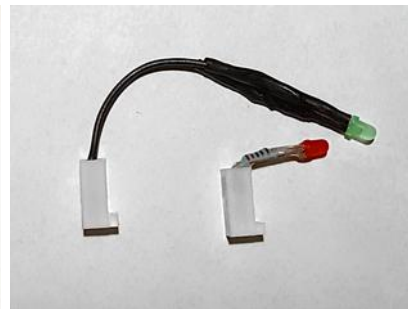


Figure 9: Two-pin Molex to bare wires for tests and LEDs for digital outputs

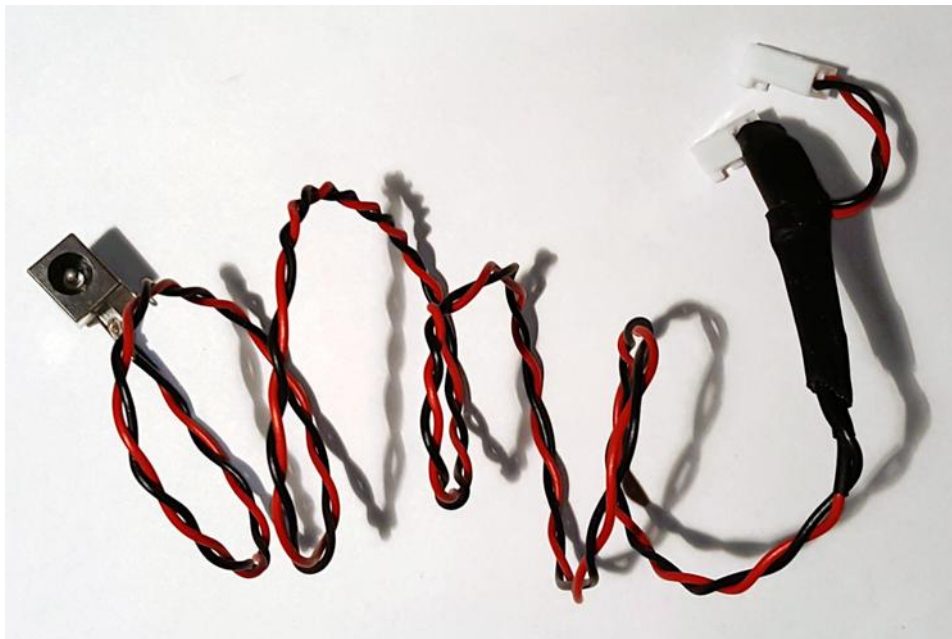


Figure 10: Power cable, with extra connector for range_select

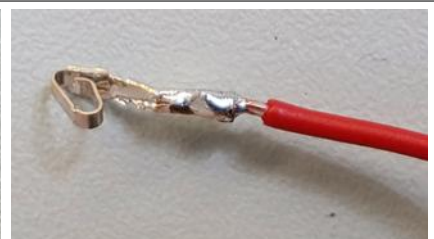
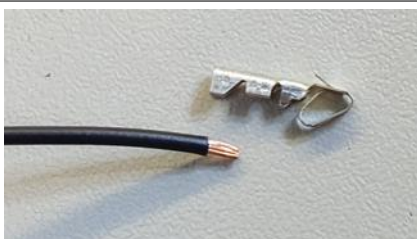


Figure 11: Crimping process

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/ μ s, down from 10 V/ μ 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 strong. For small overloads (eg 1100 Lux on the 1000 scale) the circuit works correctly, but for stronger inputs it starts oscillating.

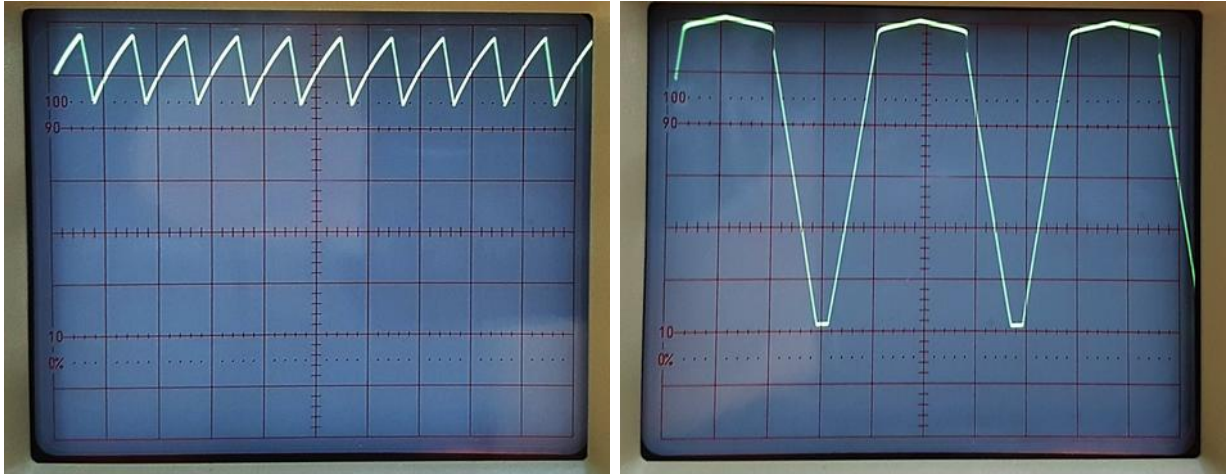
The sensor has been sloppily calibrated in order to make these tests realistic. It's not calibrated well enough for use.

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 μ F) 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 μ s/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]

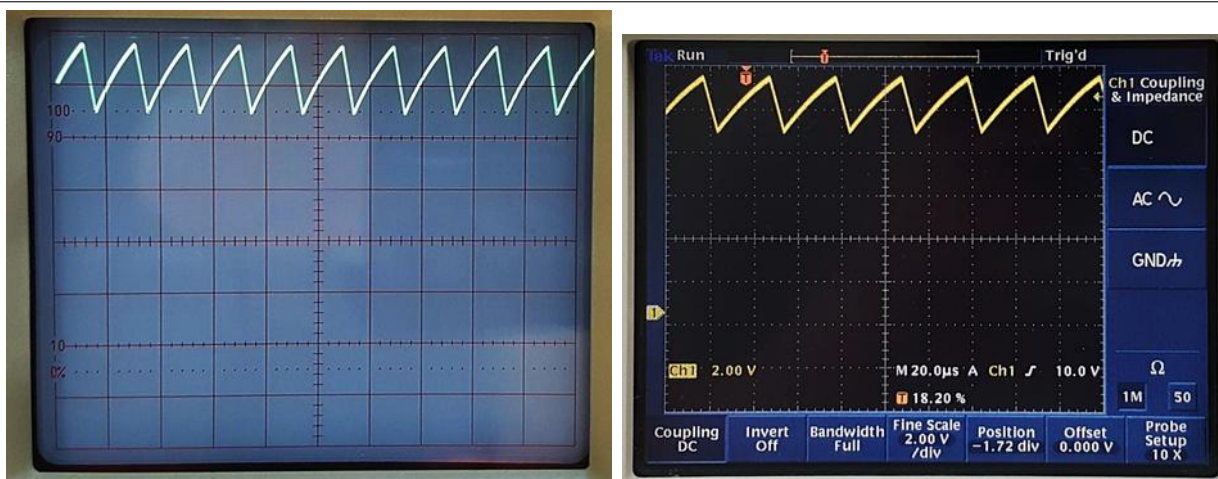
Test conditions:

- 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 μ s/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

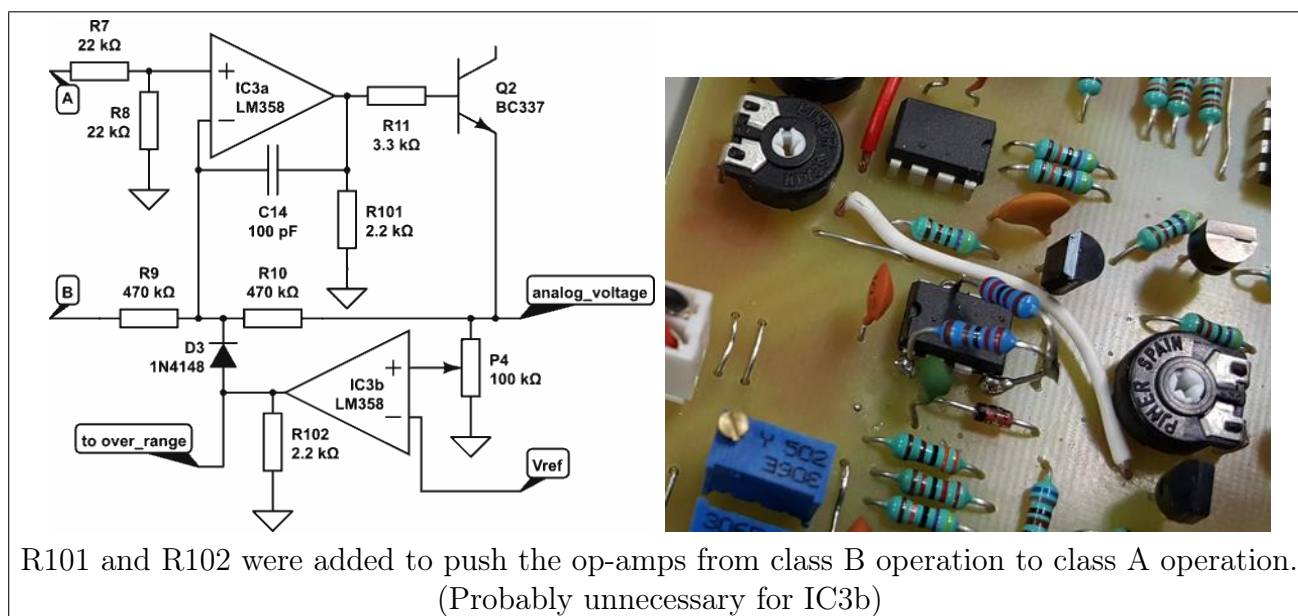


Figure 14: Modification to attempt to reduce cross-over distortion in op-amp

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.

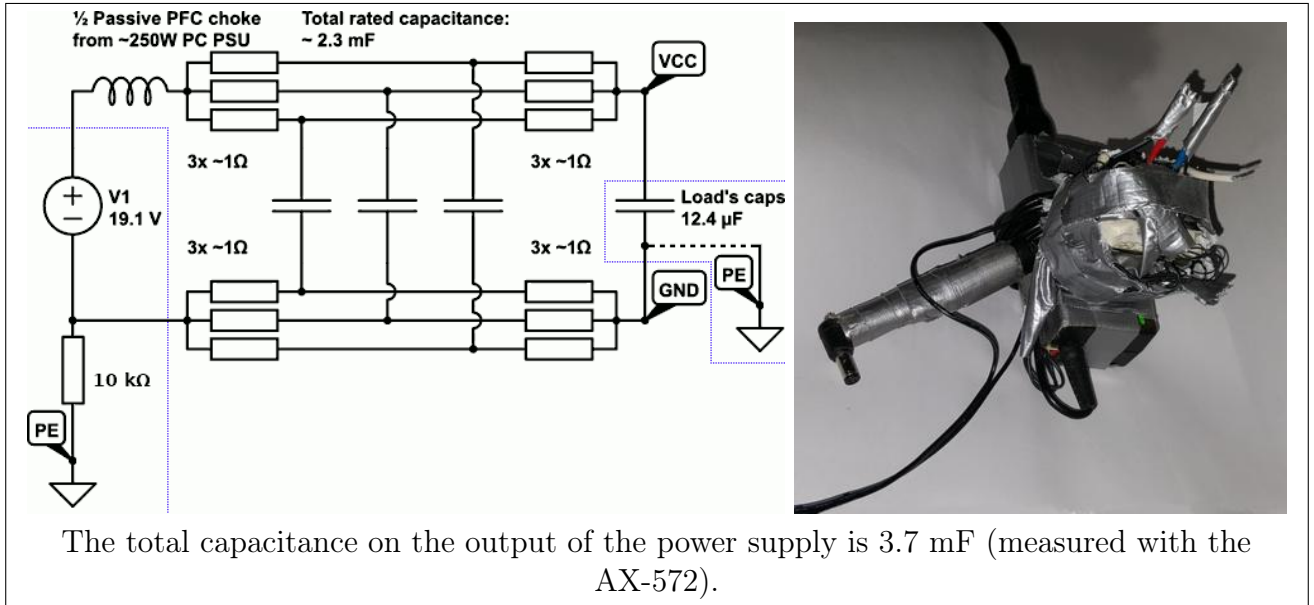


Figure 15: Power supply with LC filter

The desk lamp 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.

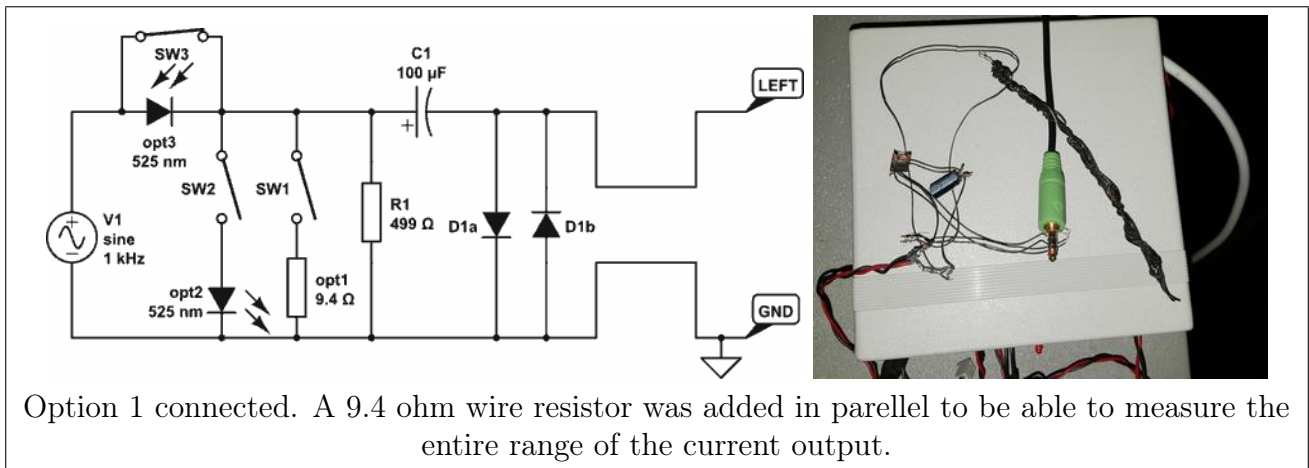


Figure 16: “Oscilloscope”

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.

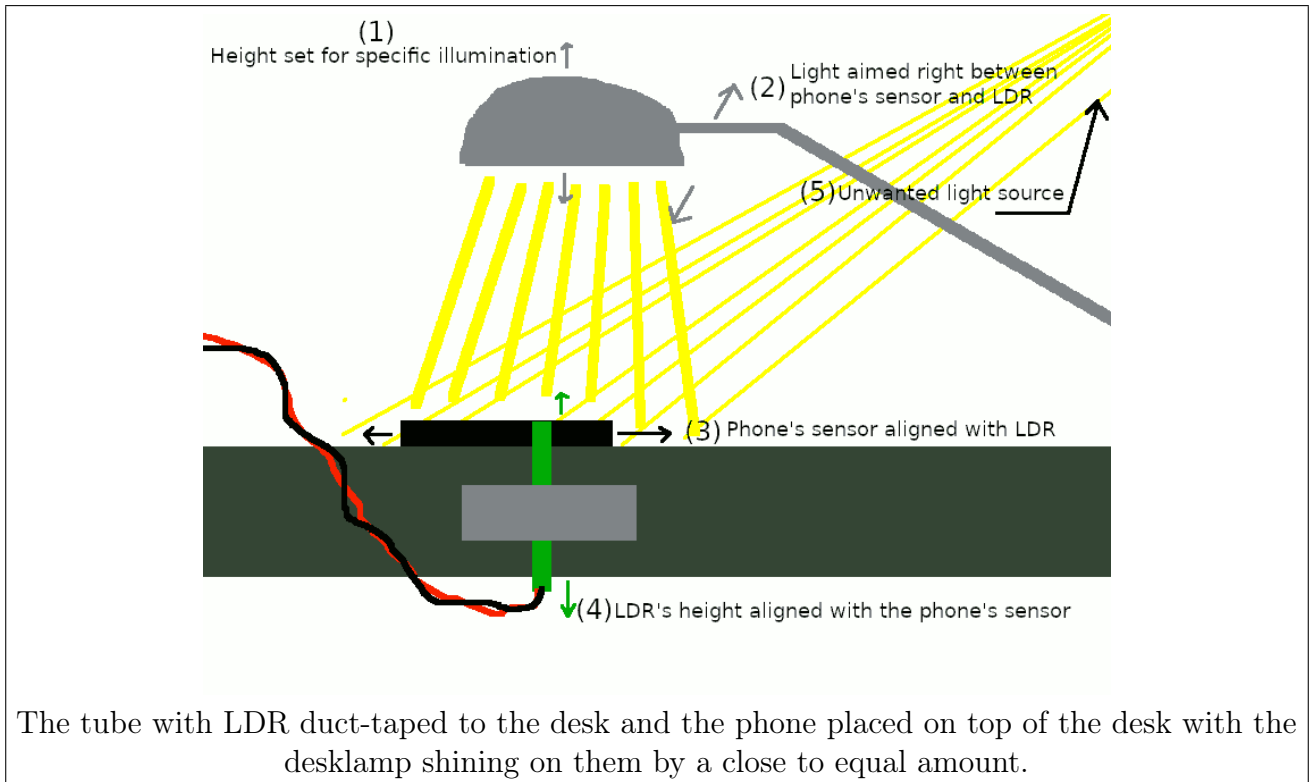


Figure 17: Usage of the modified desk lamp for controlled illumination

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.

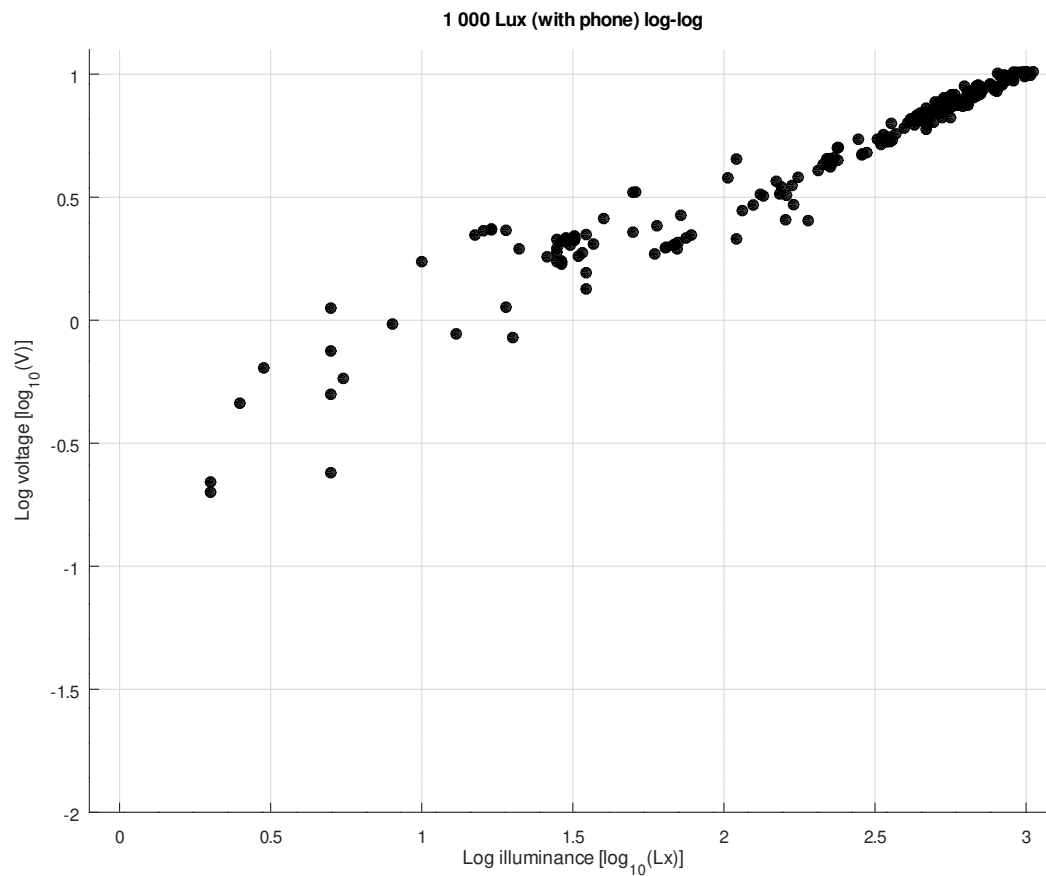
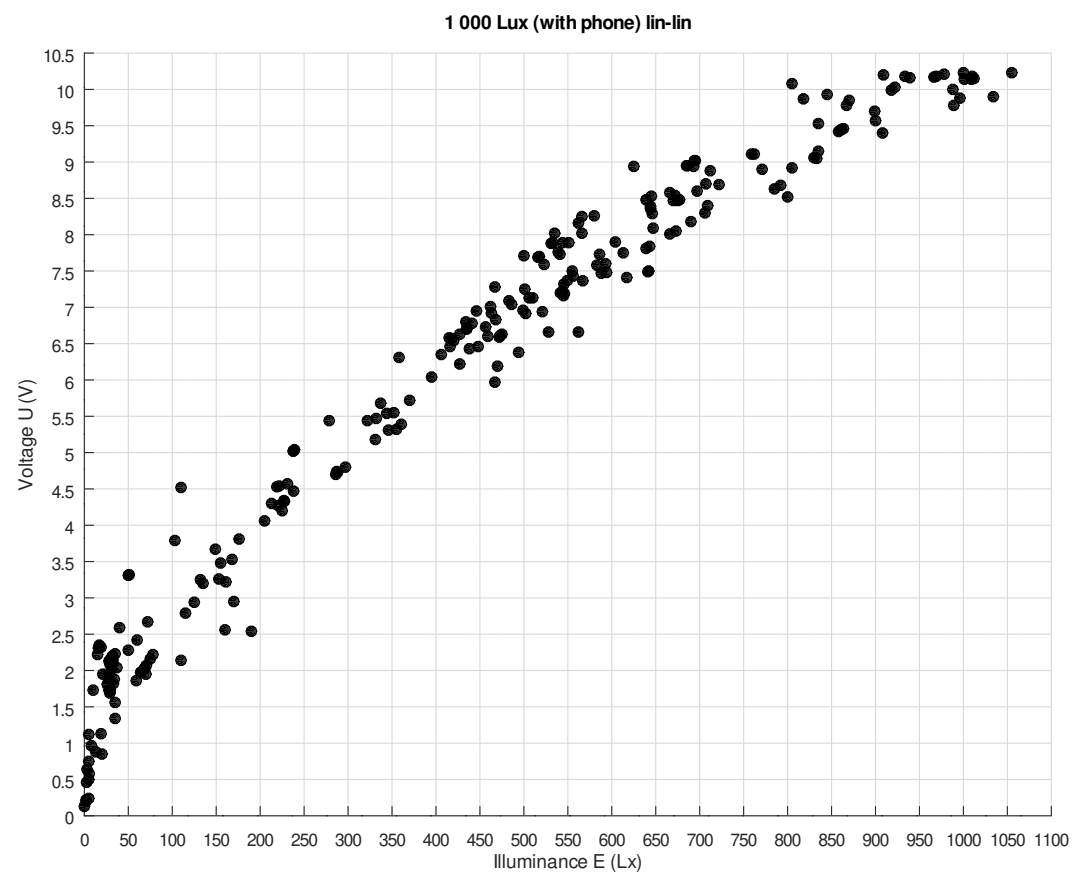


Figure 18: Scatter plots for 1 000 Lux range tested with phone

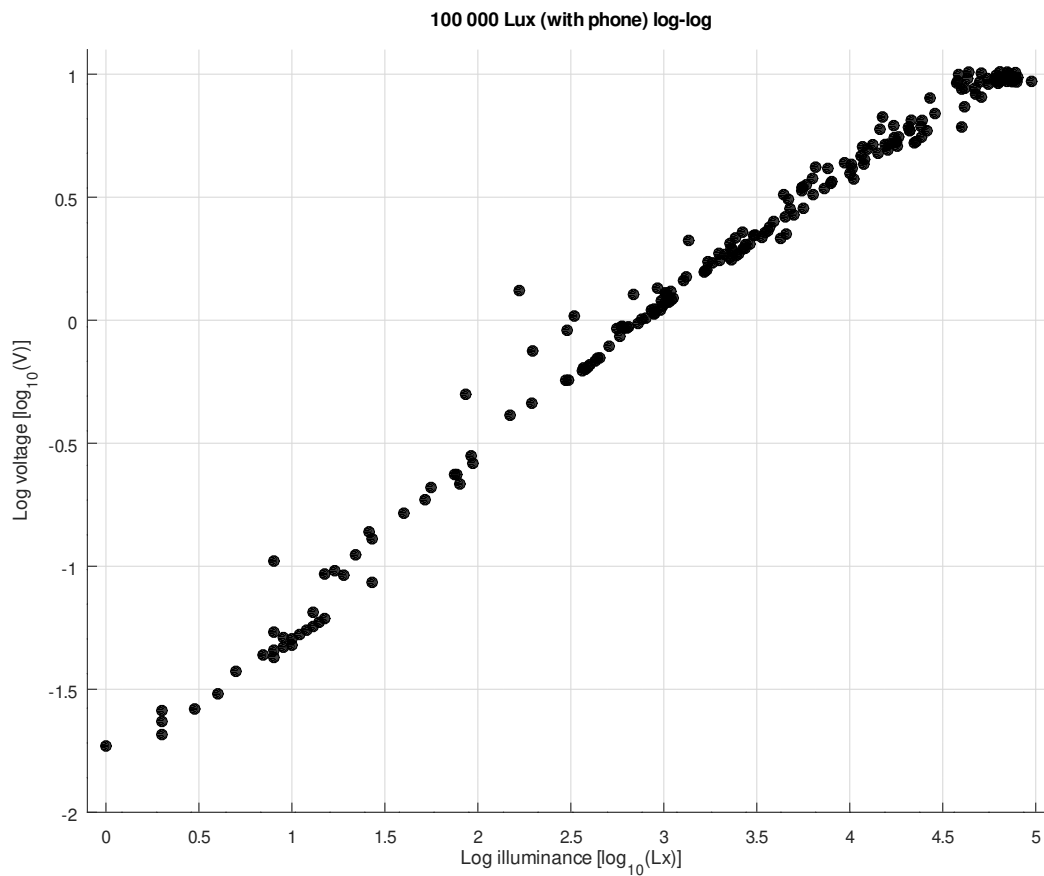
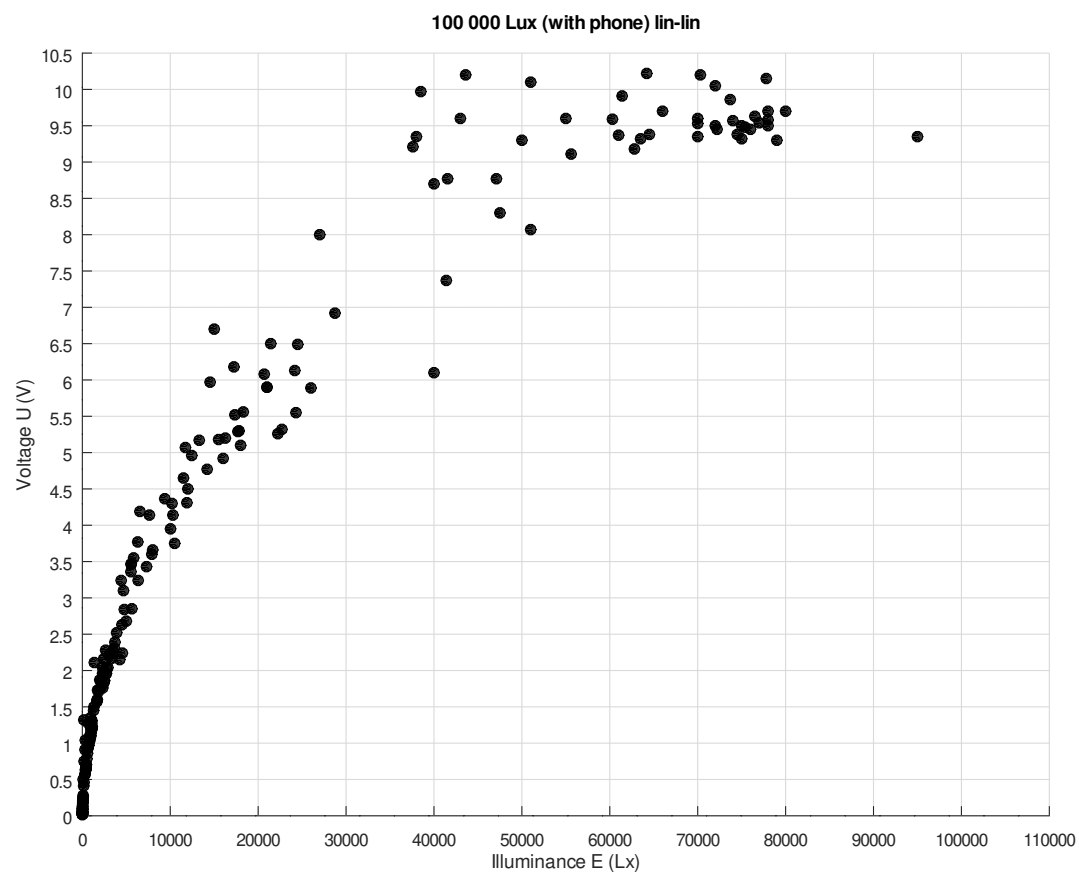


Figure 19: Scatter plots for 100 000 Lux range tested with phone

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.

$$\Delta y = y_h - y_l \quad (1)$$

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

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

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.

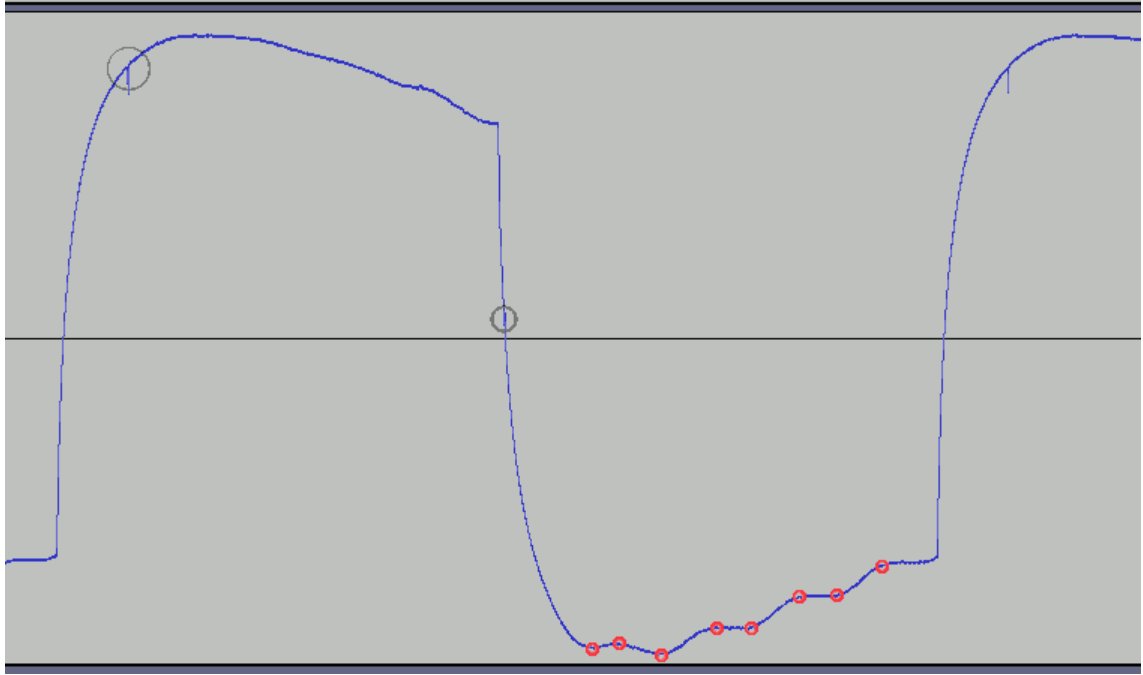
The test was performed in the dark with an LED fed with a low-frequency (5 Hz) square wave with sharp edges. The signal measured is the current output through a 9.4 Ω resistor. The voltage levels are unknown.

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

Sample	τ in milliseconds				
	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
- Unknown properties of the LDR or circuit
- Capacitance on the GPIO pin

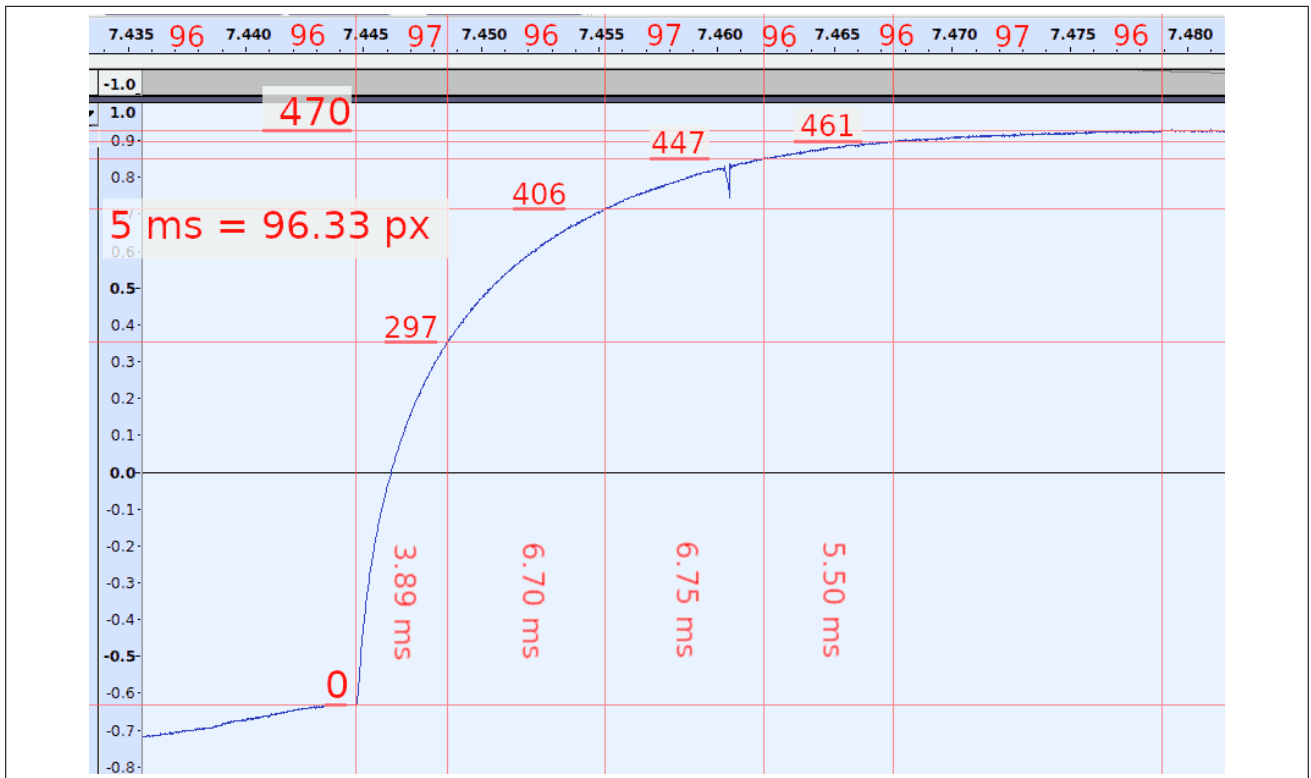


Figure 21: Measuring τ on rising edge

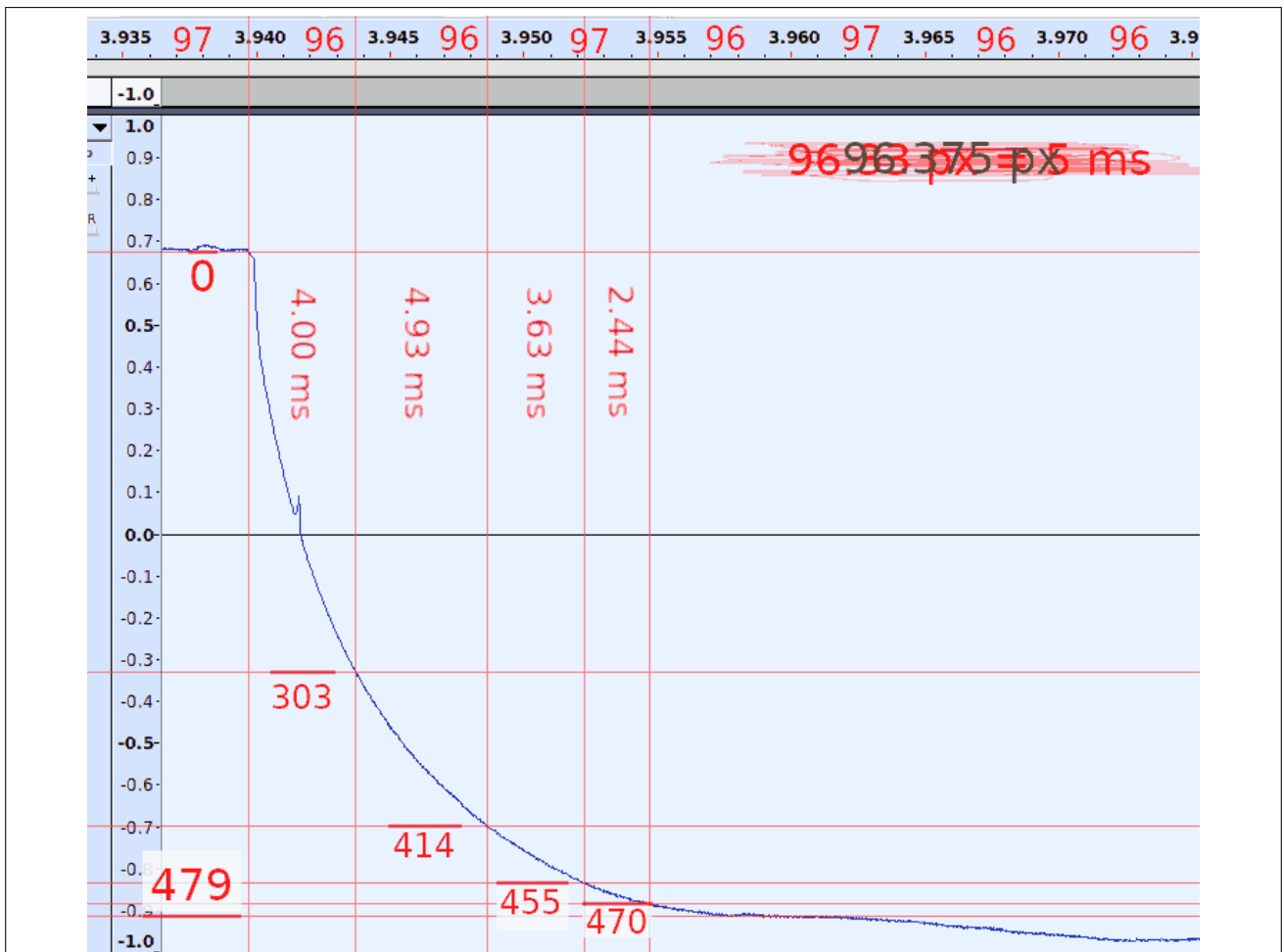


Figure 22: Measuring τ on falling edge

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.

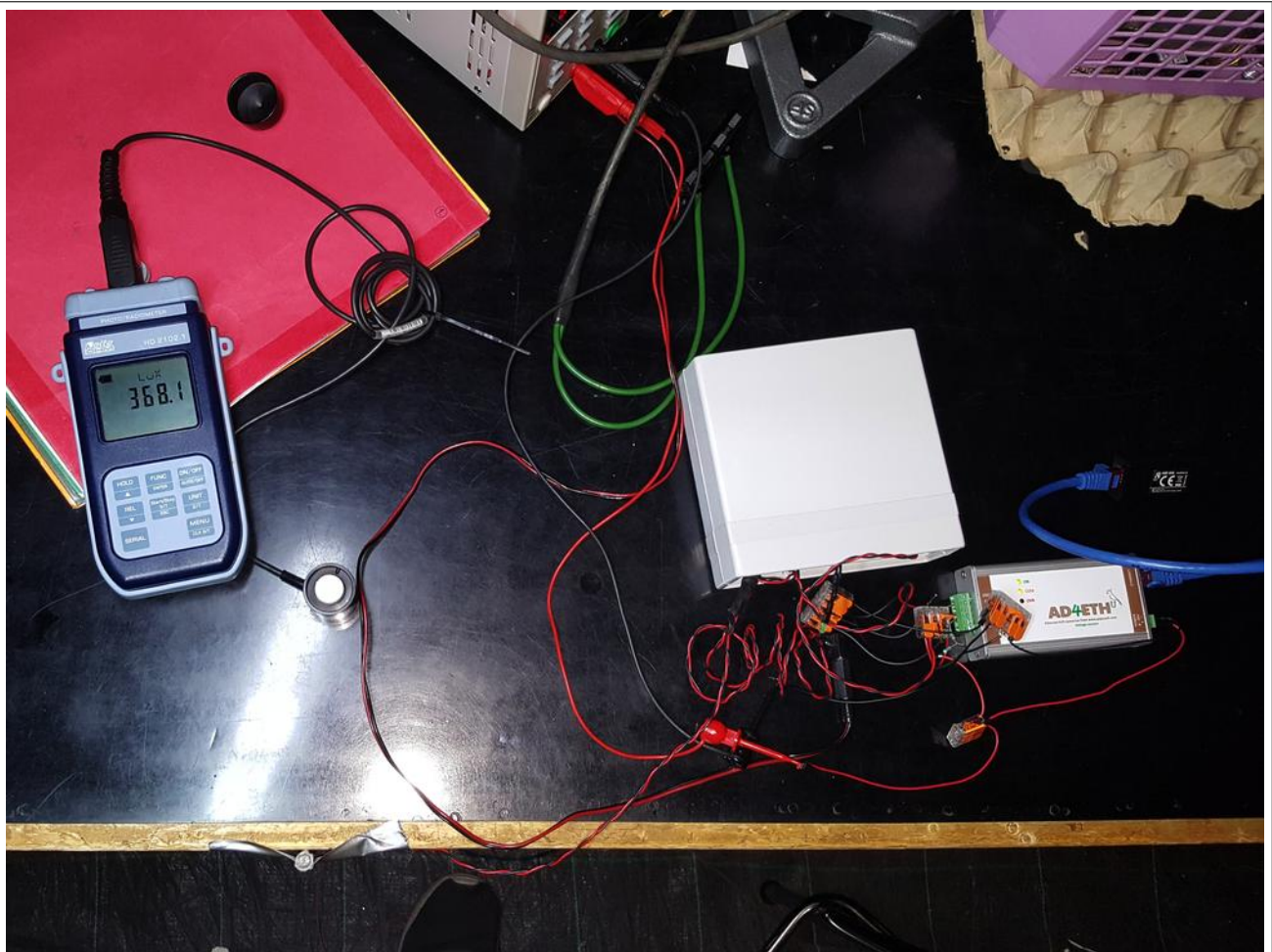
AD4ETH inputs:

1. V_{out} directly measured
2. I_{out} measured with a $500\ \Omega$ 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 workTM 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.

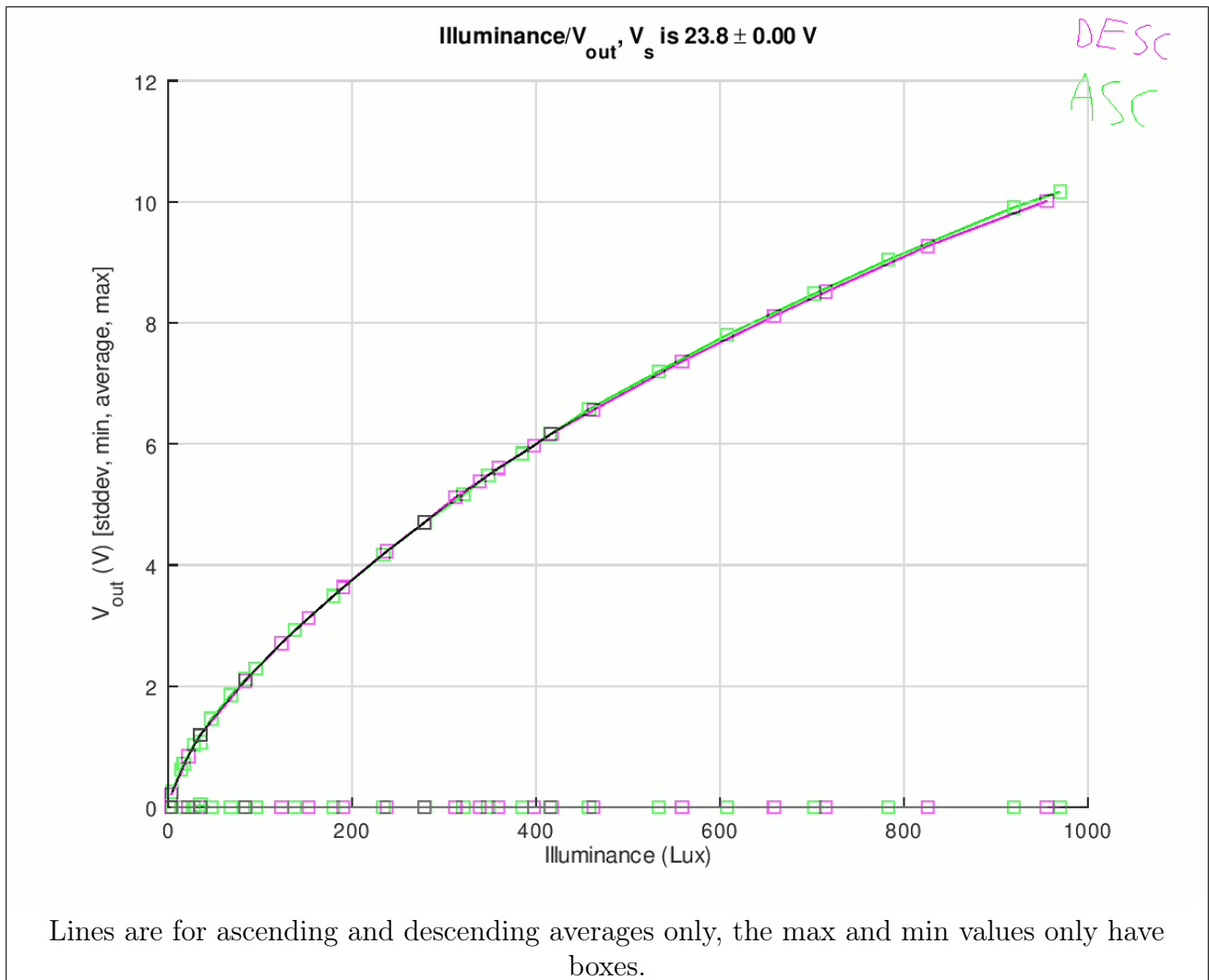
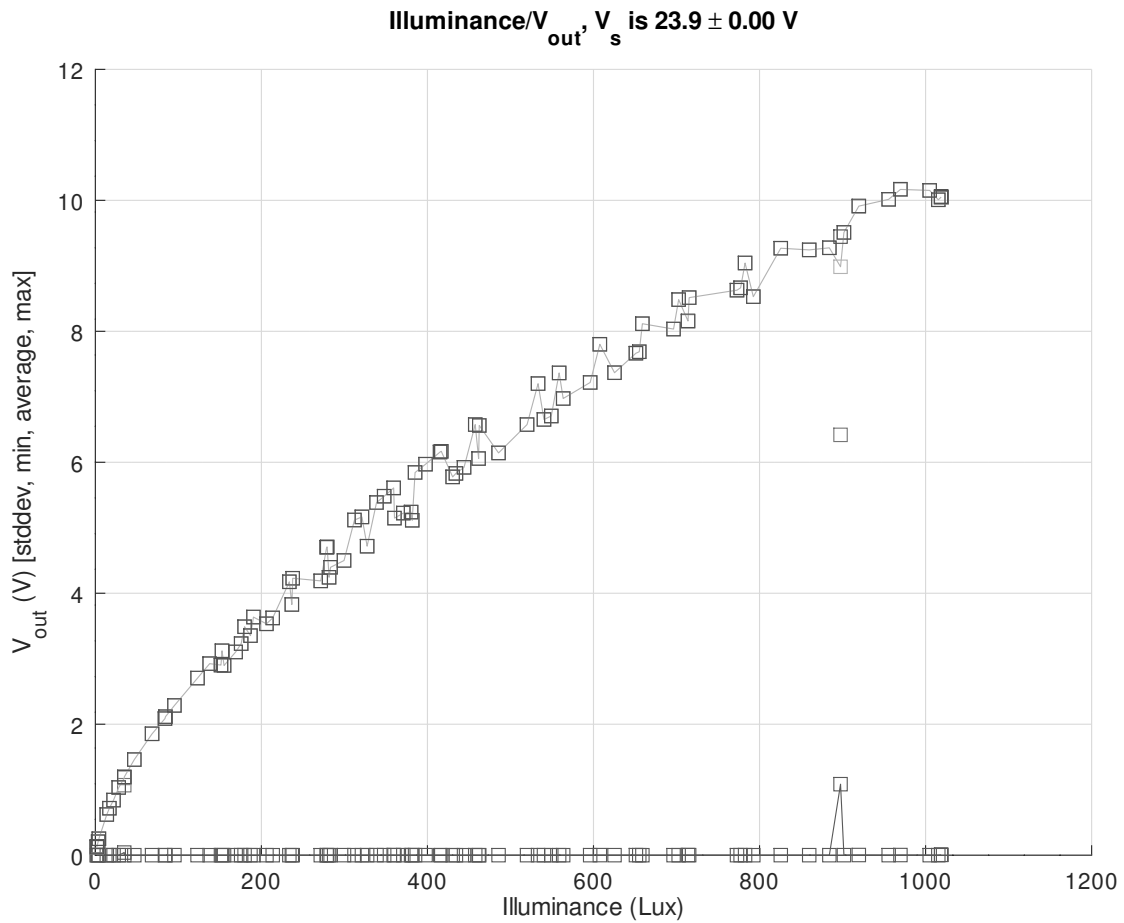


Figure 24: Hysteresis, measured on the third day



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

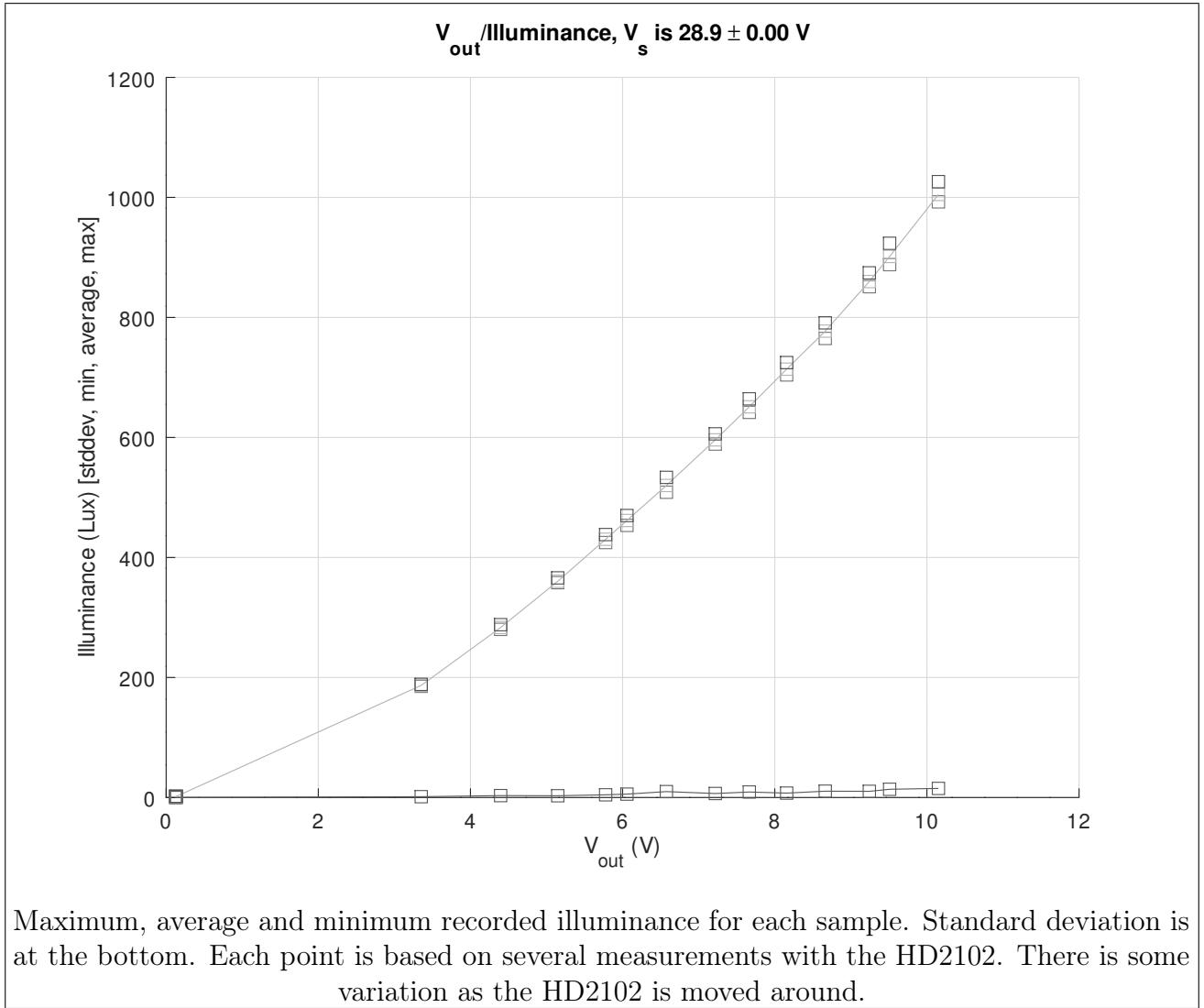


Figure 26: Measurement accuracy

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 and 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.

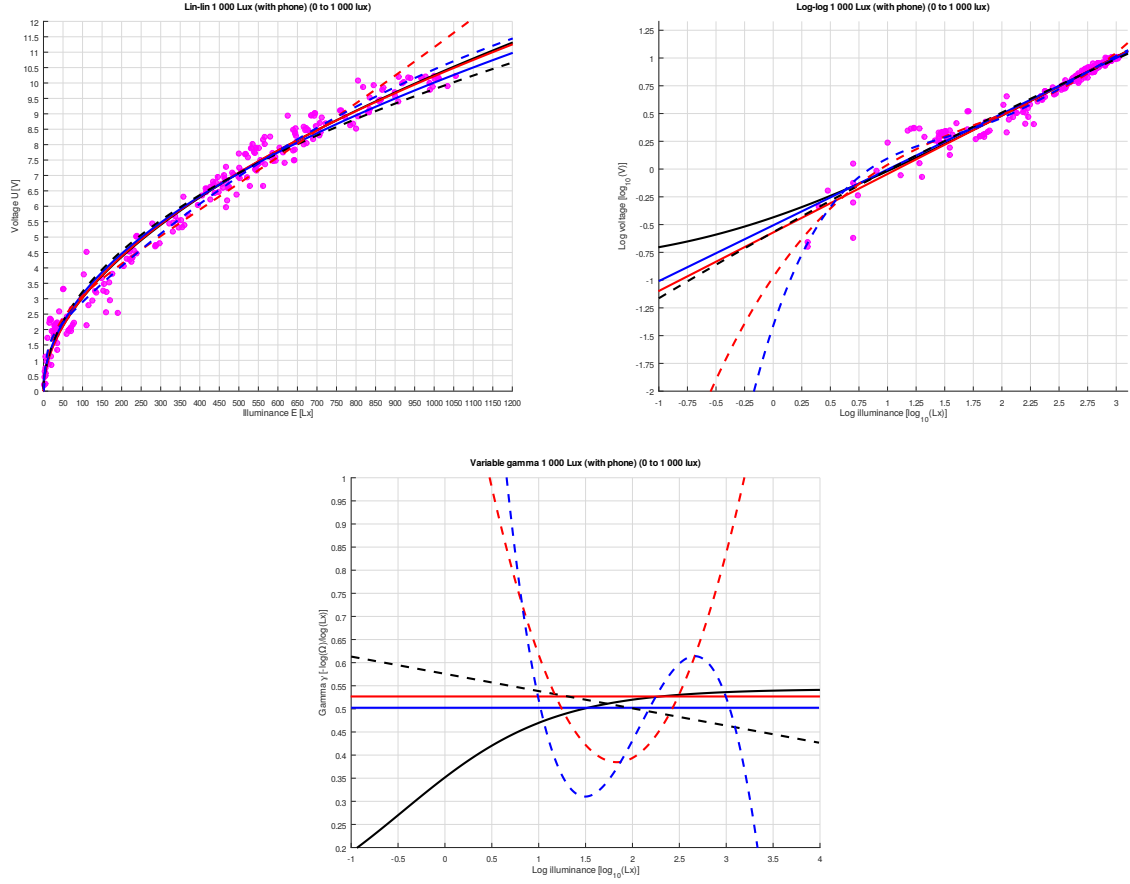
4.3.4 Results

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

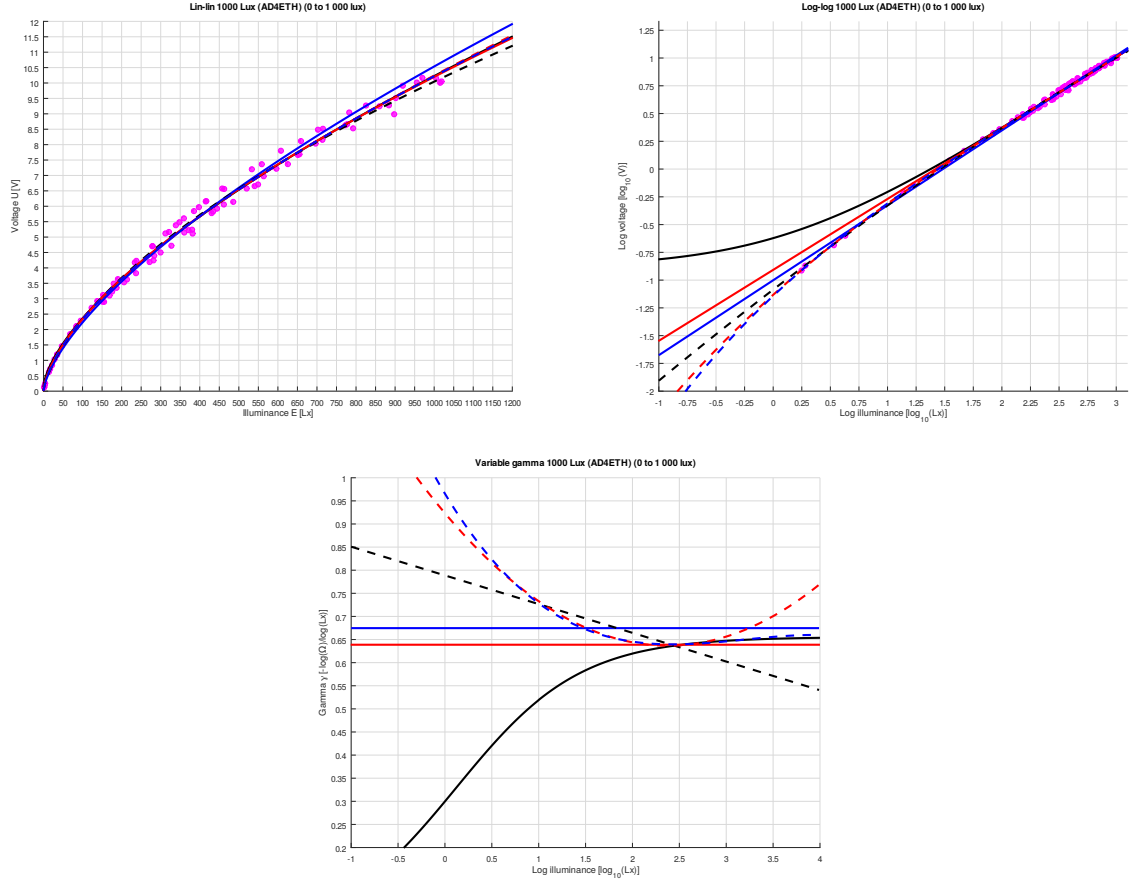
Model	r^2	γ at 10 and 100 Lux	Observations
1 (Solid black)	0.9775	(0.5431)	Bad fit on log plot
2 (Solid red)	0.9768	0.5268	Acceptable
3 (Solid blue)	0.9406	0.5024	Acceptable
4 (Dashed black)	0.9412	0.540/0.504	Acceptable
5 (Dashed red)	0.9534	0.620/0.393	Bad fit on lin plot, doesn't extrapolate
6 (Dashed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

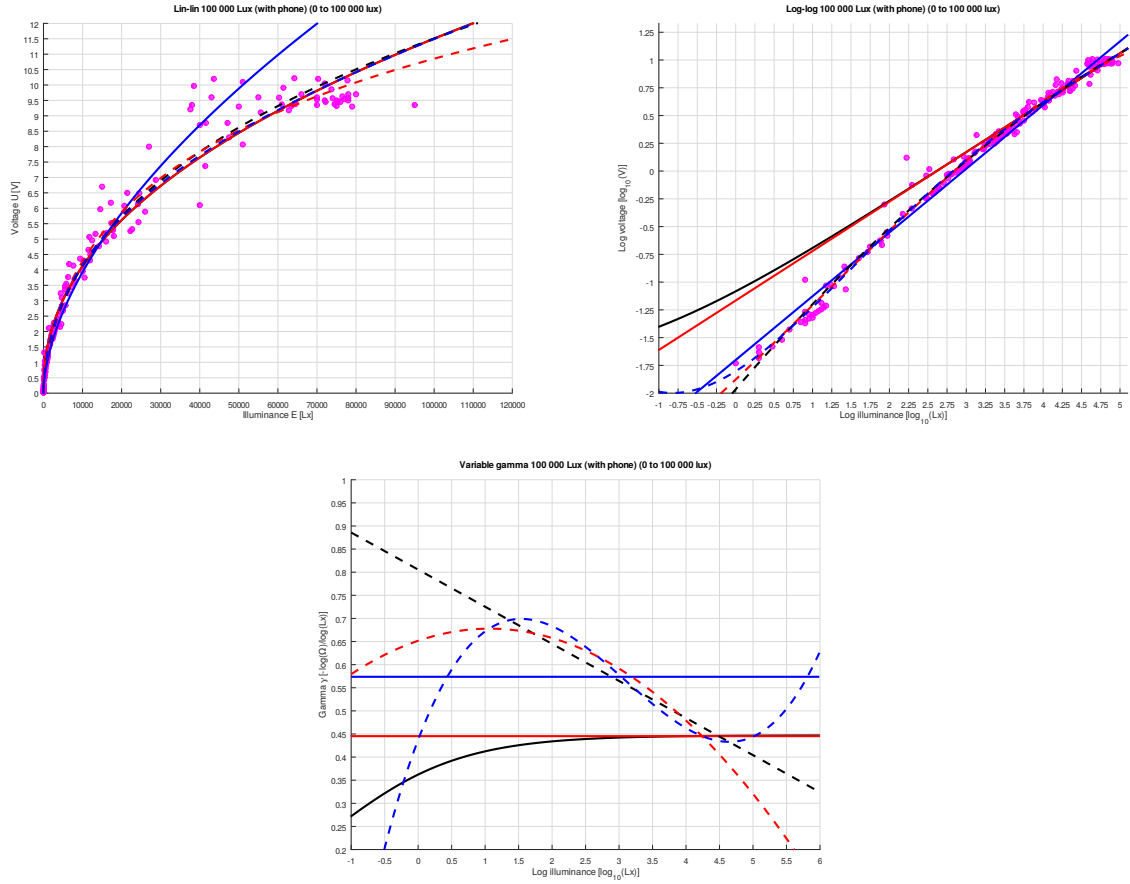
Model	r^2	γ at 10 and 100 Lux	Observations
1 (Solid black)	0.9940	(0.6556)	Bad fit on log plot
2 (Solid red)	0.9943	0.6388	Acceptable
3 (Solid blue)	0.9967	0.6747	Acceptable
4 (Dashed black)	0.9982	0.727/0.665	Acceptable
5 (Dashed red)	0.9984	0.734/0.644	Acceptable
6 (Dashed blue)	0.9984	0.729/0.693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

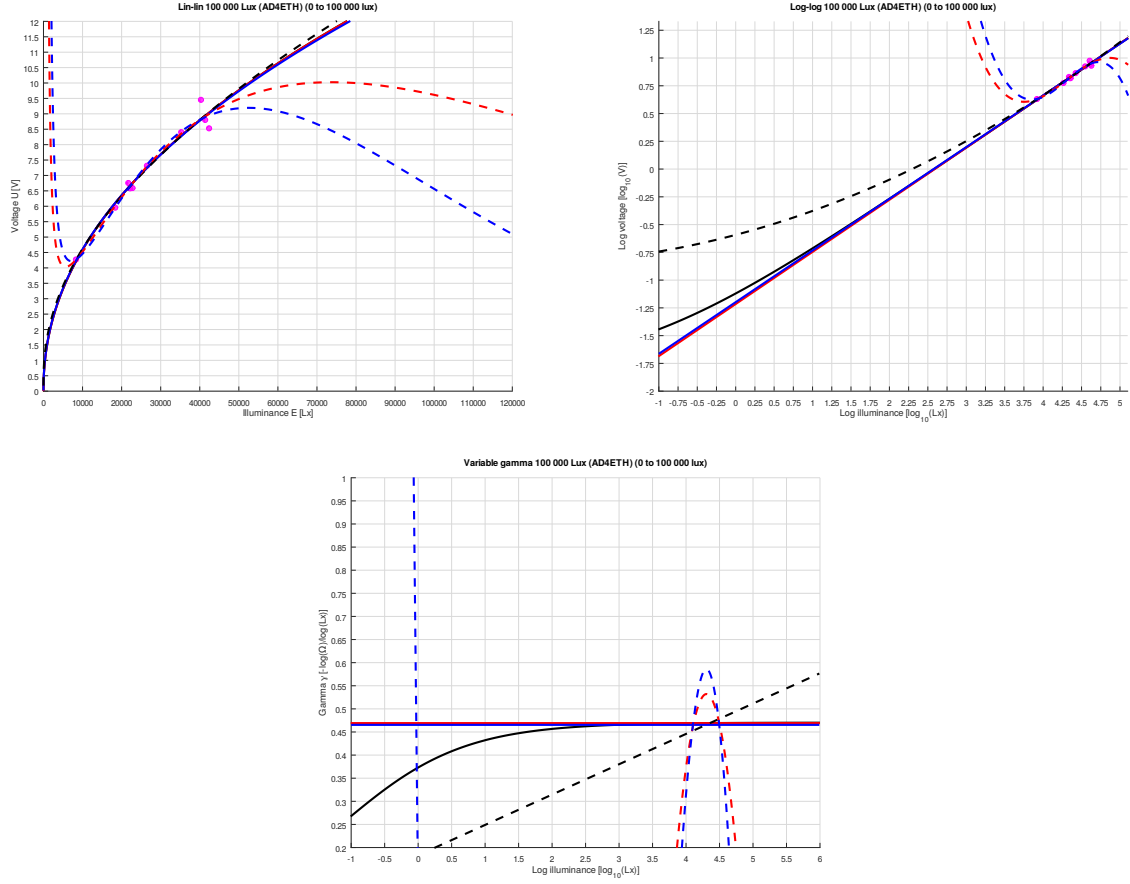
Model	r^2	γ at 10 and 100 Lux	Observations
1 (Solid black)	0.9755	(0.4470)	Bad fit on log plot
2 (Solid red)	0.9757	0.4453	Bad fit on log plot
3 (Solid blue)	0.9820	0.5737	Bad fit on lin plot
4 (Dashed black)	0.9905	0.725/0.644	Acceptable
5 (Dashed red)	0.9910	0.678/0.658	Acceptable
6 (Dashed blue)	0.9912	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)	Gamma low and increasing Doesn't extrapolate Doesn't extrapolate
2 (Solid red)	0.9647	0.4692	
3 (Solid blue)	0.9784	0.4654	
4 (Dashed black)	0.9787	0.249/0.314	
5 (Dashed red)	0.9814	-18.2/-8.61	
6 (Dashed blue)	0.9825	-7.62/-7.28	

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 decreases 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

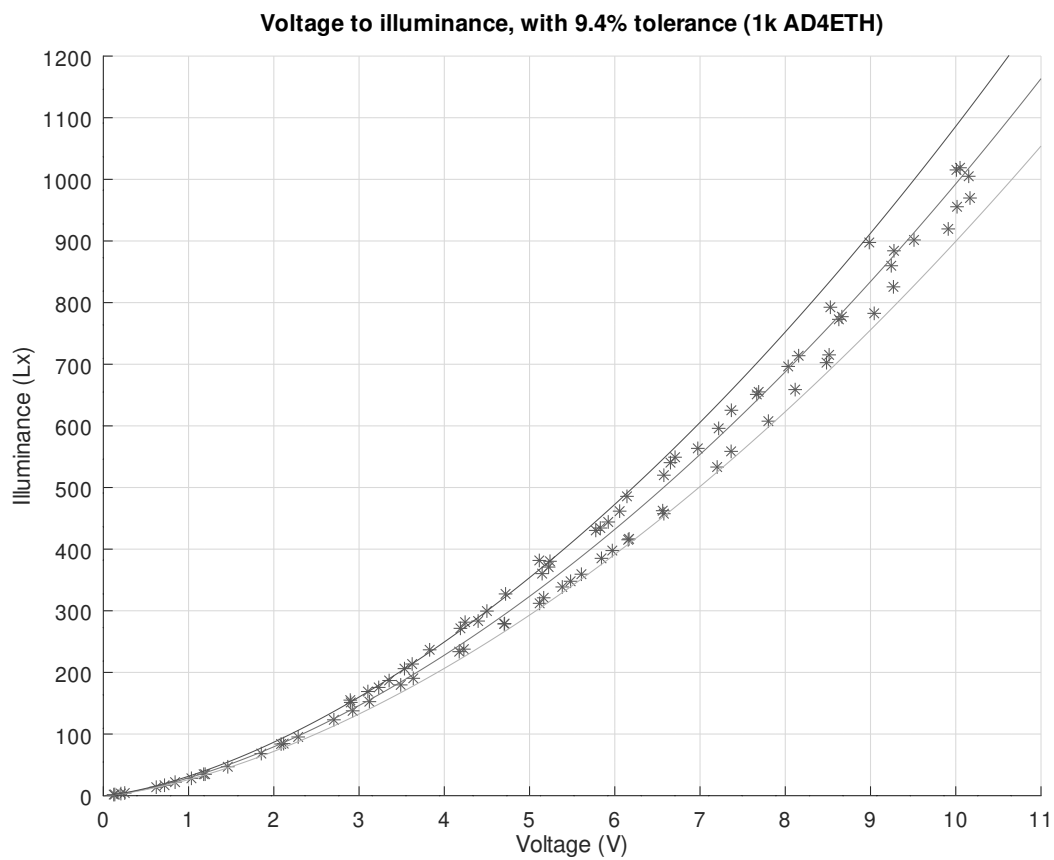
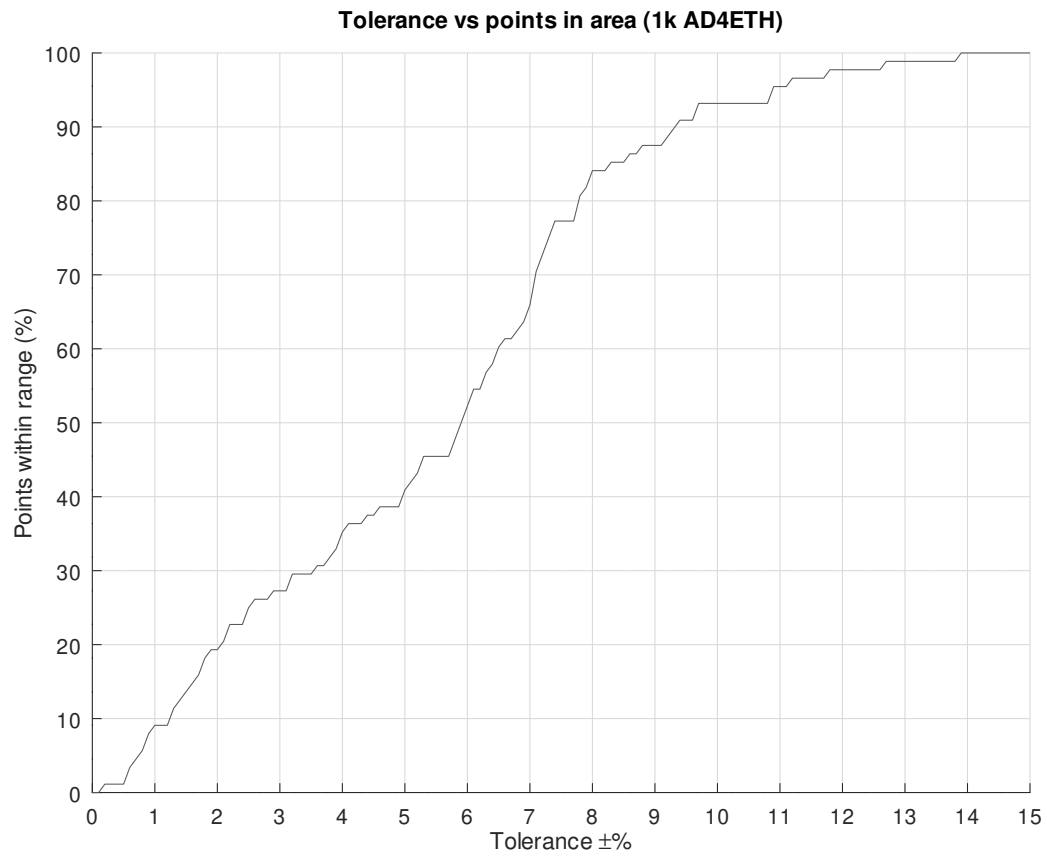
Apply numbers:

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

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

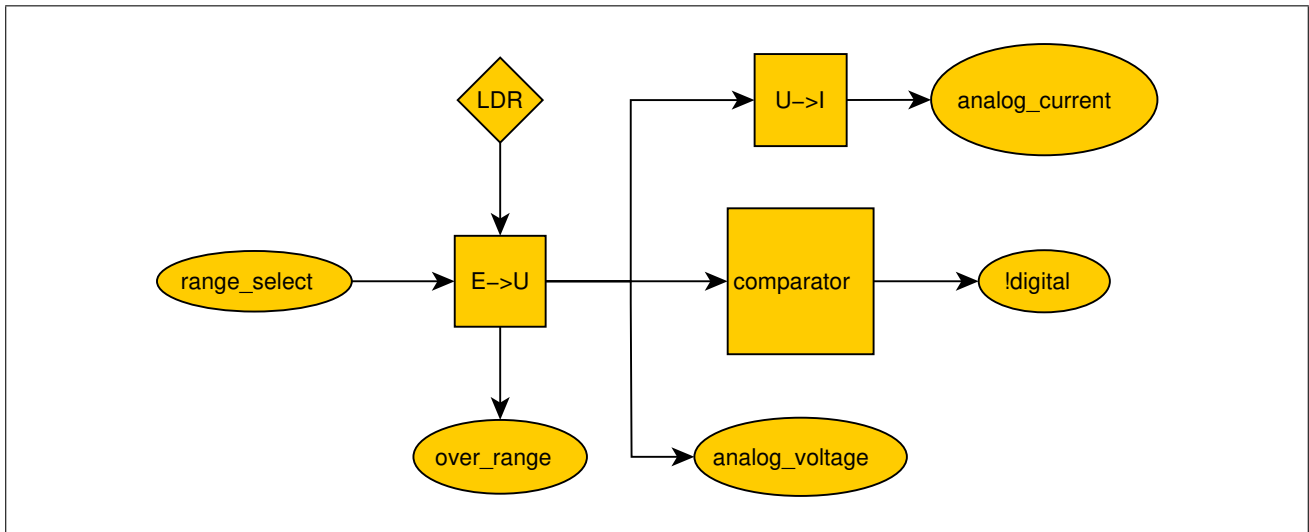


Figure 32: Topology

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	$U = (I - 4\text{ mA}) \cdot \frac{10\text{ V}}{16\text{ mA}}$
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: <code>range_select</code>	Works. Connect to supply voltage for logic 1, leave unconnected or connect to ground for logic 0. Intermediary voltages not tested.
Output: <code>over_range</code>	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: <code>digital</code>	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-\(crossreference-for-an-ltspice-simulation\)/](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`

```
1  #!/usr/bin/env python2
2
3  import code
4  import os
5  import pprint
6  import sys
7  import time
8
9  from modbustcp import ModbusTCP
10 from measure_common import stats, get_analog, get_analog_stats
11
12 from measure_timedomain import record_timedomain
13 from measure_illum import record_illum
14 from measure_vsupply import record_vsupply
15
16 '''
17 dataset = {
18     label: {
19         'type': type,
20         'scalars': {
21             key: value,
22             ...
23         },
24         'plots': [
25             {
26                 'title': title,
27                 'x-axis': column_name,
28                 'y-axes': [column_name, ...],
29                 'xlabel': xlabel,
30                 'ylabel': ylabel,
31                 'styles': [style, ...],
32             },
33             ...
34         ],
35     },
36     ...
37 }
```

```

35         'vectors': {
36             column_name: [data, ...],
37             ...
38         }
39     },
40     ...
41 }
42 '''
43
44 def main():
45     command_functions = {
46         'illum': record_illum,
47         'vsupply': record_vsupply,
48         'scope': record_timedomain,
49         'load': load_dataset,
50         'save': save_dataset,
51         'exit': prog_exit,
52         'rename': data_rename,
53         'del': data_del,
54         'plot': plot,
55         'list': list_labels,
56         'scalars': scalars,
57         'help': disp_help,
58         'realtime': realtime,
59         'connect': connect,
60         'cat': merge_data,
61         'python': python,
62         'shell': shell,
63     }
64     connection = [None]
65     dataset = {}
66     while True:
67         sys.stderr.write('> ')
68         sys.stderr.flush()
69         try:
70             line = sys.stdin.readline()
71         except KeyboardInterrupt:
72             sys.stderr.write('\n')
73             exit(0)
74         if line == '': # EOF
75             sys.stderr.write('\n')
76             break
77         if line == '\n':
78             continue
79         parts = filter(None, line.rstrip('\n').split(' '))
80         command, args = parts[0], parts[1:]
81         if command not in command_functions:
82             sys.stderr.write('Unknown command\n')
83             continue
84         #command_functions[command](dataset, connection, *args)
85         try:
86             command_functions[command](dataset, connection, *args)
87         except TypeError:
88             sys.stderr.write('Bad number of arguments\n')
89
90 def disp_help(dataset, connection):
91     print('''
92 connect <IP> [<port>]          Connect to AD4ETH
93 realtime                      Continuously read analog inputs twice a second
94 load <filename>               Load data from file

```



```

95  save <filename>                Store data to file
96  illum <label>                  Enter illuminance vs output measuring
97  vsupply <label> <lux> ...      Enter supply voltage dependence measuring
98  scope <label> <lux> ...        Measure signals for 1 second
99  plot <label> [<filename>]       Display or save an Octave plot of <label>
100 list                           List labels in loaded data
101 scalars <label>                 Print scalar values belonging to object
102 rename <old> <new>              Rename label
103 del <label>                     Delete data by label
104 cat <label> ... output <label> Merge data to new label
105 python                          Interactive Python to deal with stuff manually
106 shell [command]                 Launch a shell or run a command
107 help                            This message
108 exit                            Exit without saving
109
110 "illum" measuring
111     Constant supply voltage, varying illumination
112     Enter illuminance values (*) separated by spaces
113     Type "done" to finish
114     *) The uncertainty of the illuminance values will be recorded as well
115        as the variation in output signals over 1 second
116
117 "vsupply" measuring
118     Constant illumination, varying supply voltage
119     Press enter to take another reading
120     Type "done" to finish')
121
122 def merge_data(dataset, connection, *args):
123     if len(args) < 3:
124         sys.stderr.write('Too few arguments\n')
125         return
126     if args[-2] != 'output':
127         sys.stderr.write('Invalid syntax\n')
128         return
129     output_label = args[-1]
130     input_labels = args[:-2]
131     first_input = dataset[input_labels[0]]
132     data_type = first_input['type']
133     for input_label in input_labels:
134         if dataset[input_label]['type'] != data_type:
135             sys.stderr.write('Mixed types\n')
136             return
137     #
138     dest = {}
139     dest['type'] = data_type
140     # Scalars
141     dest['scalars'] = {}
142     for scalar in first_input['scalars']:
143         # Is it stats?
144         test = first_input['scalars'][scalar]
145         is_stats = False
146         if isinstance(test, dict):
147             items = {'count', 'min', 'max', 'average', 'stddev'}
148             if sorted(items) == sorted(test.keys()):
149                 is_stats = True
150         if not is_stats:
151             sys.stderr.write('Warning: Unkown scalar: {}\n'.format(scalar))
152             dest['scalars'][scalar] = eval(repr(test))
153         else:
154             count = 0

```

```

155         total = 0
156         sum_sqr_err = 0
157         global_min = test['min']
158         global_max = test['max']
159         for input_label in input_labels:
160             input_stats = dataset[input_label]['scalars'][scalar]
161             this_count = input_stats['count']
162             count += this_count
163             total += this_count * input_stats['average']
164             sum_sqr_err += this_count * input_stats['stddev']**2
165             global_min = min(global_min, input_stats['min'])
166             global_max = min(global_max, input_stats['min'])
167         output_stats = {}
168         output_stats['count'] = count
169         output_stats['average'] = total/count
170         output_stats['stddev'] = (sum_sqr_err/count)**.5
171         output_stats['min'] = global_min
172         output_stats['max'] = global_max
173         dest['scalars'][scalar] = output_stats
174
175         # Plots
176         sys.stderr.write('Notice: Using plot data from first data\n')
177         dest['plots'] = eval(repr(first_input['plots']))
178
179         # Vectors
180         dest['vectors'] = {}
181         for vector in first_input['vectors']:
182             dest['vectors'][vector] = []
183             for input_label in input_labels:
184                 dest['vectors'][vector] += dataset[input_label]['vectors'][vector]
185
186         # Save
187         dataset[output_label] = dest
188
189     def python(dataset, connection):
190         code.interact(local=locals())
191
192     def shell(dataset, connection, *args):
193         if args:
194             os.system(' '.join(args))
195         else:
196             os.system('$SHELL || sh')
197
198     def connect(dataset, connection, address, port="502"):
199         try:
200             connection[0] = ModbusTCP(address, int(port))
201         except:
202             sys.stderr.write('Connection failed\n')
203
204     def realtime(dataset, connection):
205         try:
206             while True:
207                 Vout, Iout, Vs = get_analog(connection)
208                 print('Vout: {:.3f} V\tIout: {:.3f} mA\tVs: {:.1f} V'.format(
209                     Vout, Iout, Vs))
210                 time.sleep(0.5)
211         except KeyboardInterrupt:
212             sys.stderr.write('\n')
213             return
214
215     def list_labels(dataset, connection):
216         sys.stderr.write('{:50} Type\n{:50} ----\n'.format('Name', '----'))
217         for label in sorted(dataset.keys()):

```

```

215         sys.stdout.write('{:50} {}\n'.format(label, dataset[label]['type']))
216
217 def scalars(dataset, connection, label):
218     try:
219         print(pprint.pformat(dataset[label]['scalars']))
220     except KeyError:
221         sys.stderr.write('No such object: {}. \n'.format(repr(label)))
222
223 def plot(dataset, connection, label, *args):
224     def matlab(vector):
225         '''Convert Python list of floats to Matlab horizontal vector'''
226         return '[' + ' '.join(map(str, vector)) + ']'
227     try:
228         data = dataset[label]
229     except KeyError:
230         sys.stderr.write('No such object: {}. \n'.format(repr(label)))
231         return
232     if len(args) > 1:
233         sys.stderr.write('Too many arguments')
234         return
235     elif len(args) == 1:
236         filename = args[0]
237     else:
238         filename = 'tmp-plot'
239     try:
240         f = open(filename, 'wx') # Open file only if it doesn't exist
241     except:
242         sys.stderr.write('Error opening file for writing.\n')
243         return
244     f.write('#!/usr/bin/env octave\n')
245     for fig_index, plot in enumerate(data['plots']):
246         f.write('fig{0} = figure;\n'.format(fig_index))
247         f.write('hold on;\n')
248         f.write('[x{0}, x{0}_order] = sort({1});\n'.format(
249             fig_index,
250             matlab(data['vectors'][plot['x-axis']]
251         ))
252     for y_index, y_name in enumerate(plot['y-axes']):
253         f.write('y{0}_{1} = {2}(x{0}_order);\n'.format(
254             fig_index, y_index,
255             matlab(data['vectors'][y_name])
256         ))
257         f.write('plot(x{0}, y{0}_{1}, {2});\n'.format(
258             fig_index, y_index, repr(plot['styles'][y_index])
259         ))
260         f.write('title({});\n'.format(repr(plot['title']).replace(r'\\', '\\\\')))
261         f.write('xlabel({});\n'.format(repr(plot['xlabel'])))
262         f.write('ylabel({});\n'.format(repr(plot['ylabel'])))
263         f.write('grid on;\nhold off;\n')
264     for i in range(len(data['plots'])):
265         f.write('waitfor(fig{0});\n'.format(i))
266     f.close()
267     os.chmod(filename, 0o755)
268     if len(args) == 0:
269         os.spawnl(os.P_NOWAIT, filename, filename)
270         # It appears that without the delay the script is removed before Octave
271         # gets to opening it.
272         time.sleep(2)
273         os.unlink(filename)
274

```

```

275 def load_dataset(dataset, connection, filename):
276     try:
277         new_dataset = eval(open(filename).read())
278     except:
279         sys.stderr.write('Load failed\n')
280         return
281     dataset.clear()
282     for key in list(new_dataset):
283         dataset[key] = new_dataset[key]
284     del new_dataset[key]
285
286 def save_dataset(dataset, connection, filename):
287     try:
288         f = open(filename, 'w')
289         f.write(pprint.pformat(dataset))
290         f.close()
291     except:
292         sys.stderr.write('Save failed')
293
294 def prog_exit(dataset, connection):
295     exit(0)
296
297 def data_rename(dataset, connection, old, new):
298     if new in dataset:
299         sys.stderr.write('Destination label exists\n')
300     try:
301         dataset[new] = dataset[old]
302         del dataset[old]
303     except KeyError:
304         sys.stderr.write('No such object: {}. \n'.format(repr(old)))
305
306 def data_del(dataset, connection, label):
307     try:
308         del dataset[label]
309     except KeyError:
310         sys.stderr.write('No such object: {}. \n'.format(repr(label)))
311
312
313 if __name__ == '__main__':
314     main()

```

7.1.2 measure_common.py

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

```

1  import time
2
3  def stats(data, old=None):
4      data = map(float, data)
5      if old is None:
6          old = {
7              'min': float("inf"),
8              'max': float("-inf"),
9              'count': 0,
10             'average': 0.0,
11             'stddev': 0.0,
12         }
13     result = {}
14     result['min'] = min(min(data), old['min'])
15     result['max'] = max(max(data), old['max'])

```

```

16     count = len(data) + old['count']
17     result['count'] = count
18     new_sum_data = sum(data)
19     old_sum_data = old['average'] * old['count']
20     avg = (new_sum_data+old_sum_data) / count
21     result['average'] = avg
22     new_sum_sqr_err = sum([(x-avg)**2 for x in data])
23     old_sum_sqr_err = old['stddev']**2 * old['count']
24     result['stddev'] = ((new_sum_sqr_err+old_sum_sqr_err)/count)**.5
25     return result
26
27 def get_analog(connection):
28     '''
29     Vout, Iout, Vs
30     '''
31     # TODO
32     #for i in range(1000):
33     #    i**i
34     #return (0.5*time.time())%10, (0.5*time.time())%16+4, (0.25*time.time())%10+19
35     # Real code here
36     response = connection[0].request([
37         0xff,          # Unit ID
38         0x04,          # Read input register(s)
39         0x00, 0x00,    # Start address 0 for channel 1
40         0x00, 0x0c    # 12 registers (3 channels * 4 registers per channel)
41     ])
42     # response[0]          # Unit ID echo
43     assert response[1] == 0x04 # Function code echo, high bit indicates error
44     assert response[2] == 0x18 # 24 bytes are to be returned
45     data = response[3:]
46     Vout = (data[2]*256 + data[3]) / 1000.0
47     Iout = (data[10]*256 + data[11]) / 500.0
48     Vs = 3 * (data[18]*256 + data[19]) / 1000.0
49     return Vout, Iout, Vs
50
51 def get_analog_stats(connection):
52     Vout_array = []
53     Iout_array = []
54     Vs_array = []
55     start_t = time.time()
56     while True:
57         Vout, Iout, Vs = get_analog(connection)
58         timestamp = time.time()
59         Vout_array.append(Vout)
60         Iout_array.append(Iout)
61         Vs_array.append(Vs)
62         if timestamp - start_t > 1:
63             break
64     return stats(Vout_array), stats(Iout_array), stats(Vs_array)

```

7.1.3 measure_illum.py

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

```

1  import pprint
2  import sys
3
4  from measure_common import stats, get_analog, get_analog_stats
5
6  default_illum = {

```

```

7     'type': 'illum',
8     'plots': [
9         {
10             'title': None,
11             'xlabel': 'Illuminance (Lux)',
12             'ylabel': 'V_{out} (V) [stddev, min, average, max]',
13             'x-axis': 'illuminate-average',
14             'y-axes': ['Vout-stddev', 'Vout-min', 'Vout-average', 'Vout-max'],
15             'styles': ['-s', 's', '-s', 's']
16         },
17         {
18             'title': None,
19             'xlabel': 'Illuminance (Lux)',
20             'ylabel': 'I_{out} (mA) [stddev, min, average, max]',
21             'x-axis': 'illuminate-average',
22             'y-axes': ['Iout-stddev', 'Iout-min', 'Iout-average', 'Iout-max'],
23             'styles': ['-s', 's', '-s', 's']
24         },
25         {
26             'title': None,
27             'xlabel': 'V_{out} (V)',
28             'ylabel': 'Illuminance (Lux) [stddev, min, average, max]',
29             'x-axis': 'Vout-average',
30             'y-axes': ['illuminate-stddev', 'illuminate-min',
31                       'illuminate-average', 'illuminate-max'],
32             'styles': ['-s', 's', '-s', 's']
33         },
34         {
35             'title': None,
36             'xlabel': 'I_{out} (mA)',
37             'ylabel': 'Illuminance (Lux) [stddev, min, average, max]',
38             'x-axis': 'Iout-average',
39             'y-axes': ['illuminate-stddev', 'illuminate-min',
40                       'illuminate-average', 'illuminate-max'],
41             'styles': ['-s', 's', '-s', 's']
42         },
43         {
44             'title': None,
45             'xlabel': 'U_{out} (V)',
46             'ylabel': 'I_{out} (mA)',
47             'x-axis': 'Vout-average',
48             'y-axes': ['Iout-average'],
49             'styles': ['-s']
50         }
51     ],
52     'scalars': {
53         'Vs': None
54     },
55     'vectors': {
56         'illuminate-stddev': [],
57         'illuminate-min': [],
58         'illuminate-average': [],
59         'illuminate-max': [],
60         'Vout-stddev': [],
61         'Vout-min': [],
62         'Vout-average': [],
63         'Vout-max': [],
64         'Iout-stddev': [],
65         'Iout-min': [],
66         'Iout-average': [],

```

```

67         'Iout-max': [],
68     }
69 }
70
71 def record_illum(dataset, connection, label):
72     '''
73     '''
74     if label in dataset:
75         if dataset[label]['type'] != 'illum':
76             sys.stderr.write('Incompatible\n')
77             return
78     if label not in dataset:
79         dataset[label] = default_illum.copy()
80     # Load data
81     data = eval(pprint.pformat(dataset[label]))
82     # Always update plot info
83     data['plots'] = default_illum['plots'][:]
84     # Update scalars and plot titles
85     scalars = data['scalars']
86     # TODO: Vs is replaced by the latest run
87     # scalars['Vs'] = stats(get_analog_stats(connection)[2], scalars['Vs'])
88     scalars['Vs'] = get_analog_stats(connection)[2]
89     Vs = 'V_s is {:.1f} \\pm {:.2f} V'.format(
90         scalars['Vs']['average'],
91         scalars['Vs']['stddev']
92     )
93     data['plots'][0]['title'] = 'Illuminance/V_{out}, ' + Vs
94     data['plots'][1]['title'] = 'Illuminance/I_{out}, ' + Vs
95     data['plots'][2]['title'] = 'V_{out}/Illuminance, ' + Vs
96     data['plots'][3]['title'] = 'I_{out}/Illuminance, ' + Vs
97     data['plots'][4]['title'] = 'V_{out}/I_{out}, ' + Vs
98     # Sample new data
99     while True:
100         sys.stderr.write('>> ')
101         sys.stderr.flush()
102         line = sys.stdin.readline()
103         if line == 'done\n':
104             break
105         try:
106             illuminance = stats(line.rstrip('\n').split(' '))
107         except:
108             sys.stderr.write('Bad input\n')
109             continue
110         Vout, Iout, _ = get_analog_stats(connection)
111         for key in ('stddev', 'min', 'average', 'max'):
112             data['vectors']['illumiance-'+key].append(illuminance[key])
113             data['vectors']['Vout-'+key].append(Vout[key])
114             data['vectors']['Iout-'+key].append(Iout[key])
115     # Store data
116     dataset[label] = data

```

7.1.4 measure_vsupply.py

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

```

1 import pprint
2 import sys
3
4 from measure_common import stats, get_analog, get_analog_stats
5

```

```

6  default_vsupply = {
7      'type': 'vsupply',
8      'plots': [
9          {
10             'title': None,
11             'xlabel': 'Supply voltage (V)',
12             'ylabel': 'V_{out} (V) [stddev, min, average, max]',
13             'x-axis': 'Vs',
14             'y-axes': ['Vout-stddev', 'Vout-min', 'Vout-average', 'Vout-max'],
15             'styles': ['-s', 's', '-s', 's']
16         },
17         {
18             'title': None,
19             'xlabel': 'Supply voltage (V)',
20             'ylabel': 'I_{out} (mA) [stddev, min, average, max]',
21             'x-axis': 'Vs',
22             'y-axes': ['Iout-stddev', 'Iout-min', 'Iout-average', 'Iout-max'],
23             'styles': ['-s', 's', '-s', 's']
24         },
25     ],
26     'scalars': {
27         'illuminance': None
28     },
29     'vectors': {
30         'Vs': [],
31         'Vout-stddev': [],
32         'Vout-min': [],
33         'Vout-average': [],
34         'Vout-max': [],
35         'Iout-stddev': [],
36         'Iout-min': [],
37         'Iout-average': [],
38         'Iout-max': [],
39     }
40 }
41
42 def record_vsupply(dataset, connection, label, *illuminance):
43     '''
44     '''
45     if label in dataset:
46         if dataset[label]['type'] != 'vsupply':
47             sys.stderr.write('Incompatible\n')
48             return
49     if label not in dataset:
50         dataset[label] = default_vsupply.copy()
51     # Load data
52     data = eval(pprint.pformat(dataset[label]))
53     # Always update plot info
54     data['plots'] = default_vsupply['plots'][:]
55     # Update scalars and plot titles
56     scalars = data['scalars']
57     scalars['illuminance'] = stats(illuminance, scalars['illuminance'])
58     illum = 'at {:.0f} \\pm {:.0f} Lux'.format(
59         scalars['illuminance']['average'],
60         scalars['illuminance']['stddev']
61     )
62     data['plots'][0]['title'] = 'Vout {}'.format(illum)
63     data['plots'][1]['title'] = 'Iout {}'.format(illum)
64     # Sample new data
65     while True:

```



```

66         sys.stderr.write('>> ')
67         sys.stderr.flush()
68         if sys.stdin.readline() == 'done\n':
69             break
70         Vout, Iout, Vs = get_analog_stats(connection)
71         data['vectors']['Vs'].append(Vs['average'])
72         for key in ('stddev', 'min', 'average', 'max'):
73             data['vectors']['Vout-'+key].append(Vout[key])
74             data['vectors']['Iout-'+key].append(Iout[key])
75         # Store data
76         dataset[label] = data

```

7.1.5 measure_timedomain.py

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

```

1  import pprint
2  import time
3
4  from measure_common import stats, get_analog, get_analog_stats
5
6  default_timedomain = {
7      'type': 'timedomain',
8      'plots': [
9          {
10             'title': None,
11             'xlabel': 'time (ms)',
12             'ylabel': 'V_{out} (V)',
13             'x-axis': 'time',
14             'y-axes': ['Vout'],
15             'styles': ['-s']
16         },
17         {
18             'title': None,
19             'xlabel': 'time (ms)',
20             'ylabel': 'I_{out} (mA)',
21             'x-axis': 'time',
22             'y-axes': ['Iout'],
23             'styles': ['-s']
24         },
25         {
26             'title': None,
27             'xlabel': 'time (ms)',
28             'ylabel': 'V_s (V)',
29             'x-axis': 'time',
30             'y-axes': ['Vs'],
31             'styles': ['-s']
32         },
33     ],
34     'scalars': {
35         'illuminance': None
36     },
37     'vectors': {
38         'time': [],
39         'Vout': [],
40         'Iout': [],
41         'Vs': [],
42     }
43 }
44

```

```

45 def record_timedomain(dataset, connection, label, *illuminance):
46     '''
47     '''
48     if label in dataset:
49         if dataset[label]['type'] != 'timedomain':
50             sys.stderr.write('Incompatible\n')
51             return
52     if label not in dataset:
53         dataset[label] = default_timedomain.copy()
54     # Load data
55     data = eval(pprint.pformat(dataset[label]))
56     # Always update plot info
57     data['plots'] = default_timedomain['plots'][:]
58     # Update scalars and plot titles
59     scalars = data['scalars']
60     scalars['illuminance'] = stats(illuminance, scalars['illuminance'])
61     illum = 'at {:.0f} \\pm {:.0f} Lux'.format(
62         scalars['illuminance']['average'],
63         scalars['illuminance']['stddev']
64     )
65     data['plots'][0]['title'] = 'Vout {}'.format(illum)
66     data['plots'][1]['title'] = 'Iout {}'.format(illum)
67     data['plots'][2]['title'] = 'Vs {}'.format(illum)
68     # Sample new data
69     start_t = time.time()
70     t = 0
71     while t < 1:
72         Vout, Iout, Vs = get_analog(connection)
73         t = time.time() - start_t
74         data['vectors']['Vout'].append(Vout)
75         data['vectors']['Iout'].append(Iout)
76         data['vectors']['Vs'].append(Vs)
77         data['vectors']['time'].append(t*1000)
78     # Store data
79     dataset[label] = data

```

7.1.6 modbustcp.py

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

```

1  # Python 2 only (at the moment)
2
3  import socket
4
5  class ModbusTCP():
6      '''
7      This is a synchronous Modbus TCP client.
8
9      foo = ModbusTCP(address, port=502)
10     response = foo.request([unit_id, function, data_bytes ...])
11     '''
12     def __init__(self, address, port=502):
13         self.conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
14         self.conn.connect((address, port))
15         self.sequence_id = 0
16
17     def request(self, modbus_request):
18         '''
19         modbus_request = [
20             unit_identifier,

```

```

21         function_code,
22         data_bytes ...
23     ]
24     '''
25     datalen = len(modbus_request)
26     modbus_tcp_request_packet = [
27         (self.sequence_id >> 8) & 0xff, # Transaction identifier, high
28         self.sequence_id & 0xff,        # Transaction identifier, low
29         0, 0,                          # Protocol identifier, high & low
30         (datalen >> 8) & 0xff,          # Length field, high
31         datalen & 0xff,                # Length field, low
32     ]
33     modbus_tcp_request_packet += modbus_request
34     # send of the request
35     self.conn.sendall(''.join(map(chr, modbus_tcp_request_packet)))
36     # Response
37     response_head = map(ord, self.conn.recv(6))
38     response_transaction = response_head[0]*256 + response_head[1]
39     response_protocol = response_head[2]*256 + response_head[3]
40     response_length = response_head[4]*256 + response_head[5]
41     assert response_transaction == self.sequence_id
42     assert response_protocol == 0
43     # Get the Modbus part of the response
44     self.sequence_id += 1
45     return map(ord, self.conn.recv(response_length))
46
47     def __del__(self):
48         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`

```

1  #!/usr/bin/env octave
2
3  # Needs package "optim"
4  pkg load optim
5
6  % Make regressions and plots for E-U data
7  % DATA: 0-1000 and 0-100000
8  % PLOTS: lin-lin, log-log, gamma
9  % LINEAR LEAST SQUARES: E^gamma + zero_offset, E^gamma
10 % LOGARITHMIC LEAST SQUARES: 1st-4th power polynomials
11
12
13 % Font size:          22      8
14 % Scatter ball size: 49      25
15
16 function [out_E, out_U] = logsafe(E, U)
17     out_E = [];
18     out_U = [];
19     for i = 1:size(E)(2)
20         if E(i) > 0 && U(i) > 0
21             out_E = [out_E E(i)];
22             out_U = [out_U U(i)];

```

```

23         end
24     end
25 end
26
27
28 function myplot(E, U, interp_arr_E, interp_arr_U, E_range, plot_title)
29     style = {"k" "r" "b" "--k" "--r" "--b"};
30     % solid black, solid red, solid blue, dashed black, dashed red, dashed blue
31
32     % LINEAR
33     f = figure('DefaultAxesFontSize', 8);
34     hold on;
35     %plot(E, U, 'dm');
36     scatter(E, U, 25, 'm', 'MarkerFaceColor', 'm');
37     for i = 1:size(interp_arr_E)(2)
38         interp_E = interp_arr_E{1,i};
39         interp_U = interp_arr_U{1,i};
40         plot(interp_E, interp_U, style{1,i}, "linewidth", 2)
41     end
42     axis([0 1.2*10^E_range 0 12]);
43     if E_range < 4
44         xticks([0:5*10^(E_range-2):1.2*10^E_range]);
45     else
46         xticks([0:10*10^(E_range-2):1.2*10^E_range]);
47     end
48     yticks([0:0.5:12]);
49     xlabel('Illuminance E [Lx]');
50     ylabel('Voltage U [V]');
51     title(['Lin-lin ' plot_title]);
52     grid on;
53     hold off;
54
55     % LOGARITHMIC
56     f2 = figure('DefaultAxesFontSize', 8);
57     hold on;
58     scatter(log10(E), log10(U), 25, 'm', 'MarkerFaceColor', 'm');
59     %plot(log10(E), log10(U), 'dm');
60     for i = 1:size(interp_arr_E)(2)
61         interp_E = interp_arr_E{1,i};
62         interp_U = interp_arr_U{1,i};
63         plot(log10(interp_E), log10(interp_U), style{1,i}, "linewidth", 2)
64     end
65     axis([-1 E_range+0.1 -2 1.33]);
66     xticks([-1:0.25:E_range]);
67     yticks([-2:0.25:1.25]);
68     %xlabel('Illuminance lg(E) (log_{10}(Lx))');
69     %ylabel('Voltage lg(U) (log_{10}(V))');
70     xlabel('Log illuminance [log_{10}(Lx)]');
71     ylabel('Log voltage [log_{10}(V)]');
72     title(['Log-log ' plot_title]);
73     grid on;
74     hold off;
75
76     % GAMMA
77     f3 = figure('DefaultAxesFontSize', 8);
78     hold on;
79     for i = 1:size(interp_arr_E)(2)
80         [interp_E, interp_U] = logsafe(interp_arr_E{1,i}, interp_arr_U{1,i});
81         interp_gamma = diff(log10(interp_U))./diff(log10(interp_E));
82         plot(log10(interp_E)(1:end-1), interp_gamma, style{1,i}, "linewidth", 2)

```

```

83     end
84     axis([-1 E_range+1 0.2 1]);
85     xticks([-1:0.5:E_range+1]);
86     yticks([0.2:0.05:1]);
87     %xlabel('Illuminance decade lg(E) (log_{10}(Lx))');
88     %ylabel('Gamma \gamma (-log(\Omega)/log(Lx))');
89     xlabel('Log illuminance [log_{10}(Lx)]');
90     ylabel('Gamma \gamma [-log(\Omega)/log(Lx)]');
91     title(['Variable gamma ' plot_title]);
92     grid on;
93     hold off;
94
95     waitfor(f);
96     waitfor(f2);
97     waitfor(f3);
98 end
99
100
101 function interpolate(E, U, data_name, is_100k)
102     plot_arr_E = {};
103     plot_arr_U = {};
104
105     if is_100k
106         plot_name = [data_name ' (0 to 100 000 lux)'];
107     else
108         plot_name = [data_name ' (0 to 1 000 lux)'];
109     end
110     printf("%s\n", plot_name);
111
112     % Excess is needed for the gamma plot
113     % E range for linear interpolation
114     if is_100k
115         interp_E = [[0:0.01:9.99] [10:0.1:99.9] [100:1:999] [1000:10:9990] ...
116                     [10000:100:99900] [10^5:10^3:10^6]];
117     else
118         interp_E = [[0:0.01:9.99] [10:0.1:99.9] [100:1:999] [1000:10:10000]];
119     end
120
121     if is_100k
122         offset = 0.0157;
123     else
124         offset = 0.1298;
125     end
126
127     % TRACE 1
128     func = @(E, param) param(1) * E.^param(2) + offset;
129     pin = [0.1 0.5];
130     [_f1, param, _kvg1, _iter1, _corp1, _covp1, _covr1, ...
131         _stdresid1, _Z1, r2] = leasqr(E, U, pin, func);
132     interp_U = func(interp_E, param);
133     printf("[SOLID BLACK]\tlin with offset:\tr^2 is %f, Gamma is %f\n",
134         r2, param(2))
135     printf("\t%f\t%f\t%f\n", param(1), param(2), offset)
136     plot_arr_E = {interp_E};
137     plot_arr_U = {interp_U};
138
139     % TRACE 2
140     func = @(E, param) param(1) * E.^param(2);
141     [_f1, param, _kvg1, _iter1, _corp1, _covp1, _covr1, ...
142         _stdresid1, _Z1, r2] = leasqr(E, U, pin, func);

```

```

143 interp_U = func(interp_E, param);
144 printf("[SOLID RED]\tlin without offset:\tr^2 is %f, Gamma is %f\n",
145         r2, param(2))
146 printf("\t%f\t%f\n", param(1), param(2))
147 plot_arr_E = {plot_arr_E{1,1:end} interp_E};
148 plot_arr_U = {plot_arr_U{1,1:end} interp_U};
149
150 % Excess is needed for the gamma plot
151 % E range for logarithmic interpolation
152 [E, U] = logsafe(E, U);
153 if is_100k
154     interp_E = [-1:0.01:6];
155 else
156     interp_E = [-1:0.01:4];
157 end
158
159 % TRACE 3
160 func = @(logE, param) param(1) + param(2)*logE;
161 pin = [0.1 0.5];
162 [_f1, param, _kvg1, _iter1, _corp1, _covp1, _covr1, ...
163     _stdresid1, _Z1, r2] = leasqr(log10(E), log10(U), pin, func);
164 interp_U = func(interp_E, param);
165 printf("[SOLID BLUE]\tlog 1st:\t\tr^2 is %f, Gamma is %f\n", r2, param(2))
166 printf("\t%f\t%f\n", param(1), param(2))
167 plot_arr_E = {plot_arr_E{1,1:end} 10.^interp_E};
168 plot_arr_U = {plot_arr_U{1,1:end} 10.^interp_U};
169
170 % TRACE 4
171 func = @(logE, param) param(1) + param(2)*logE + param(3)*logE.^2;
172 pin = [0.1 0.5 0];
173 [_f1, param, _kvg1, _iter1, _corp1, _covp1, _covr1, ...
174     _stdresid1, _Z1, r2] = leasqr(log10(E), log10(U), pin, func);
175 interp_U = func(interp_E, param);
176 printf("[DASHED BLACK]\tlog 2nd:\t\t")
177 printf("r^2 is %f, Gamma is [%f + (%f * decade)]\n",
178         r2, param(2), 2*param(3))
179 printf("\t%f\t%f\t%f\n", param(1), param(2), param(3))
180 plot_arr_E = {plot_arr_E{1,1:end} 10.^interp_E};
181 plot_arr_U = {plot_arr_U{1,1:end} 10.^interp_U};
182
183 % TRACE 5
184 func = @(logE, param) param(1) + param(2)*logE + param(3)*logE.^2 ...
185         + param(4)*logE.^3;
186 pin = [0.1 0.5 0 0];
187 [_f1, param, _kvg1, _iter1, _corp1, _covp1, _covr1, ...
188     _stdresid1, _Z1, r2] = leasqr(log10(E), log10(U), pin, func);
189 interp_U = func(interp_E, param);
190 printf("[DASHED RED]\tlog 3rd:\t\t")
191 printf("r^2 is %f, Gamma is [%f + (%f * decade) + (%f * decade^2)]\n",
192         r2, param(2), 2*param(3), 3*param(4))
193 printf("\t%f\t%f\t%f\t%f\n", param(1), param(2), param(3), param(4))
194 plot_arr_E = {plot_arr_E{1,1:end} 10.^interp_E};
195 plot_arr_U = {plot_arr_U{1,1:end} 10.^interp_U};
196
197 % TRACE 6
198 func = @(logE, param) param(1) + param(2)*logE + param(3)*logE.^2 ...
199         + param(4)*logE.^3 + param(5)*logE.^4;
200 pin = [0.1 0.5 0 0 0];
201 [_f1, param, _kvg1, _iter1, _corp1, _covp1, _covr1, ...
202     _stdresid1, _Z1, r2] = leasqr(log10(E), log10(U), pin, func);

```

```

203     interp_U = func(interp_E, param);
204     printf("[DASHED BLUE]\tlog 4th:\t\ttr^2 is %f, Gamma is ", r2)
205     printf("[%f + (%f * decade) + (%f * decade^2) + (%f * decade^3)]\n",
206         param(2), 2*param(3), 3*param(4), 4*param(5))
207     printf("\t%f\t%f\t%f\t%f\t%f\n", param(1), param(2), param(3), param(4),
208         param(5))
209     plot_arr_E = {plot_arr_E{1,1:end} 10.^interp_E};
210     plot_arr_U = {plot_arr_U{1,1:end} 10.^interp_U};
211
212     % Display traces
213     if is_100k
214         myplot(E, U, plot_arr_E, plot_arr_U, 5, plot_name);
215     else
216         myplot(E, U, plot_arr_E, plot_arr_U, 3, plot_name);
217     end
218     printf("\n");
219 end
220
221 function myscatter(E, U, plot_title, is_100k)
222     if is_100k
223         E_range = 5;
224     else
225         E_range = 3;
226     end
227
228     linfig = figure('DefaultAxesFontSize', 8);
229     hold on;
230     scatter(E, U, 25, 'k', 'MarkerFaceColor', 'k');
231     axis([0 1.1*10^E_range 0 10.5]);
232     if E_range < 4
233         xticks([0:5*10^(E_range-2):1.1*10^E_range]);
234     else
235         xticks([0:10*10^(E_range-2):1.1*10^E_range]);
236     end
237     yticks([0:0.5:10.5]);
238     xlabel('Illuminance E (Lx)');
239     ylabel('Voltage U (V)');
240     title([plot_title ' lin-lin']);
241     grid on;
242     hold off;
243
244     logfig = figure('DefaultAxesFontSize', 8);
245     hold on;
246     scatter(log10(E), log10(U), 25, 'k', 'MarkerFaceColor', 'k');
247     axis([-0.1 E_range+0.1 -2 1.1]);
248     xticks([0:0.5:E_range]);
249     yticks([-2:0.5:1]);
250     xlabel('Log illuminance [log_{10}(Lx)]');
251     ylabel('Log voltage [log_{10}(V)]');
252     title([plot_title ' log-log']);
253     grid on;
254     hold off;
255
256     waitfor(linfig);
257     waitfor(logfig);
258 end
259
260
261 %set(0, "defaultaxesfontname", "Helvetica")
262 graphics_toolkit("fltk") %("gnuplot")

```

```

263
264 [E, U] = data_1k();
265 %myscatter(E, U, '1 000 Lux (with phone)', false);
266 interpolate(E, U, '1 000 Lux (with phone)', false);
267
268 [E, U] = data_1k_ad4eth();
269 interpolate(E, U, '1000 Lux (AD4ETH)', false);
270
271 [E, U] = data_100k();
272 %myscatter(E, U, '100 000 Lux (with phone)', true);
273 interpolate(E, U, '100 000 Lux (with phone)', true);
274
275 [E, U] = data_100k_ad4eth();
276 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`

```

1  #!/usr/bin/env octave
2
3  function pass_rate = test_tolerance(E_points, U_points, solution, tolerance)
4      passes = 0;
5      fails = 0;
6      for i = 1:size(U_points)(2)
7          if U_points(i) <= 0
8              continue
9          end
10         calculated_E = solution(U_points(i));
11         calculated_error = E_points(i)/calculated_E;
12         if calculated_error > 1+tolerance || calculated_error < 1-tolerance
13             fails += 1;
14         else
15             passes += 1;
16         end
17     end
18     pass_rate = passes/(passes+fails);
19 end
20
21 function passrate_vector = tolerance_to_passrate(E, U, sol, tolerance_vector)
22     passrate_vector = [];
23     for i = 1:size(tolerance_vector)(2)
24         passrate_vector = [passrate_vector test_tolerance(E,U,sol, ...
25                                                         tolerance_vector(i))];
26     end
27 end
28
29 function tolerance = find_tolerance(E,U,sol,target_passrate)
30     tolerance = 0;
31     while test_tolerance(E,U,sol, tolerance) < target_passrate
32         tolerance += 0.001;
33     end
34 end
35
36 [E_points, U_points] = data_1k_ad4eth();

```



```

37
38 solution = @(U) 10.^((0.789041-sqrt(0.487708-0.124232*log10(U)))/0.062116);
39
40 tolerance = find_tolerance(E_points,U_points,solution, 0.90);
41 printf("Tolerance: %.1f%%\n", tolerance*100)
42
43 U_range = [0.01:0.01:11];
44
45 tolerance_vector = [0:0.001:0.15];
46 passrate_vector = tolerance_to_passrate(E_points,U_points,solution, ...
47                                     tolerance_vector);
48
49 fig = figure;
50 hold on
51 plot(U_points, E_points, "*")
52 plot(U_range, solution(U_range), "-")
53 plot(U_range, (1-tolerance)*solution(U_range), "-")
54 plot(U_range, (1+tolerance)*solution(U_range), "-")
55 xlabel("Voltage (V)")
56 ylabel("Illuminance (Lx)")
57 title(sprintf("Voltage to illuminance, with %.1f%% tolerance (1k AD4ETH)", ...
58             100*tolerance))
59 grid on
60 axis([0 11 0 1200])
61 xticks([0:1:11])
62 yticks([0:100:1200])
63 hold off
64
65 fig2 = figure;
66 hold on
67 plot(100*tolerance_vector, 100*passrate_vector, "-")
68 xlabel("Tolerance \\pm%")
69 ylabel("Points within range (%)")
70 title("Tolerance vs points in area (1k AD4ETH)")
71 grid on
72 axis([0 15 0 100])
73 xticks([0:1:15])
74 yticks([0:10:100])
75 hold off
76
77 waitfor(fig)
78 waitfor(fig2)

```