Thermoino Verison 2.5 with PWM for CTC feature → LUIGI

Contents

The PWM feature

Setting the following:

cTCBinMs: 10

cTCPos: 7

0 9

1 9

2 9

3 -9

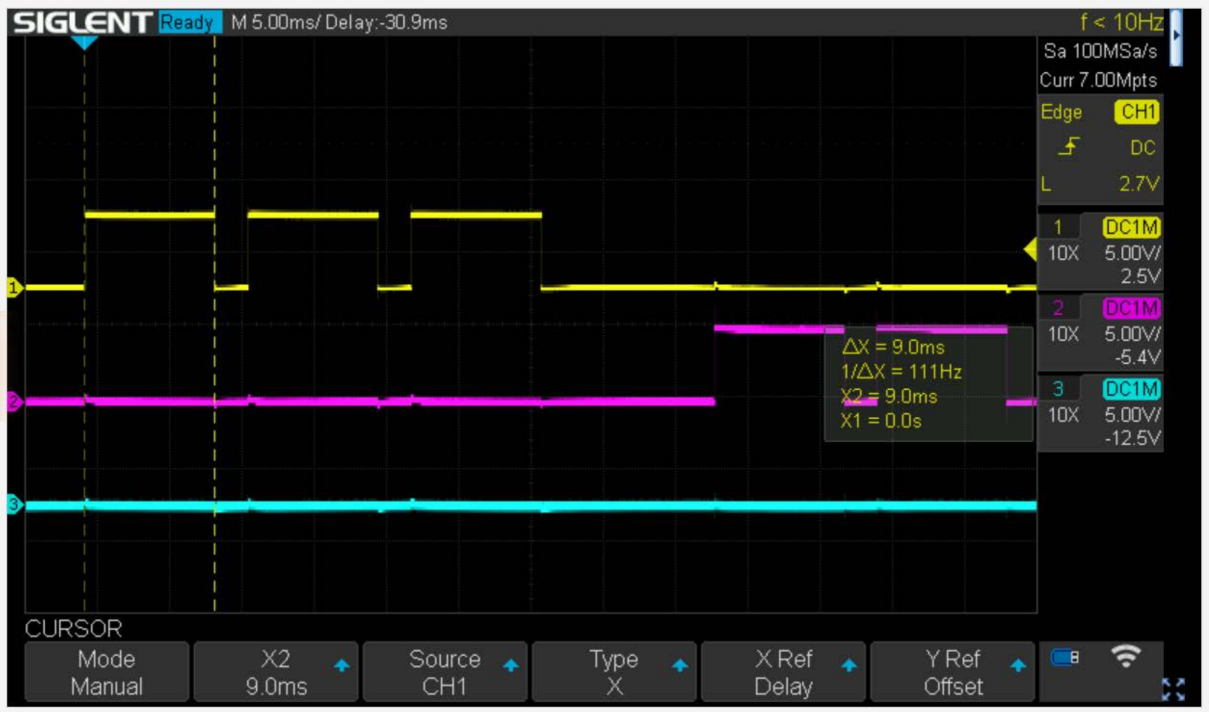
4 -9

5 -9

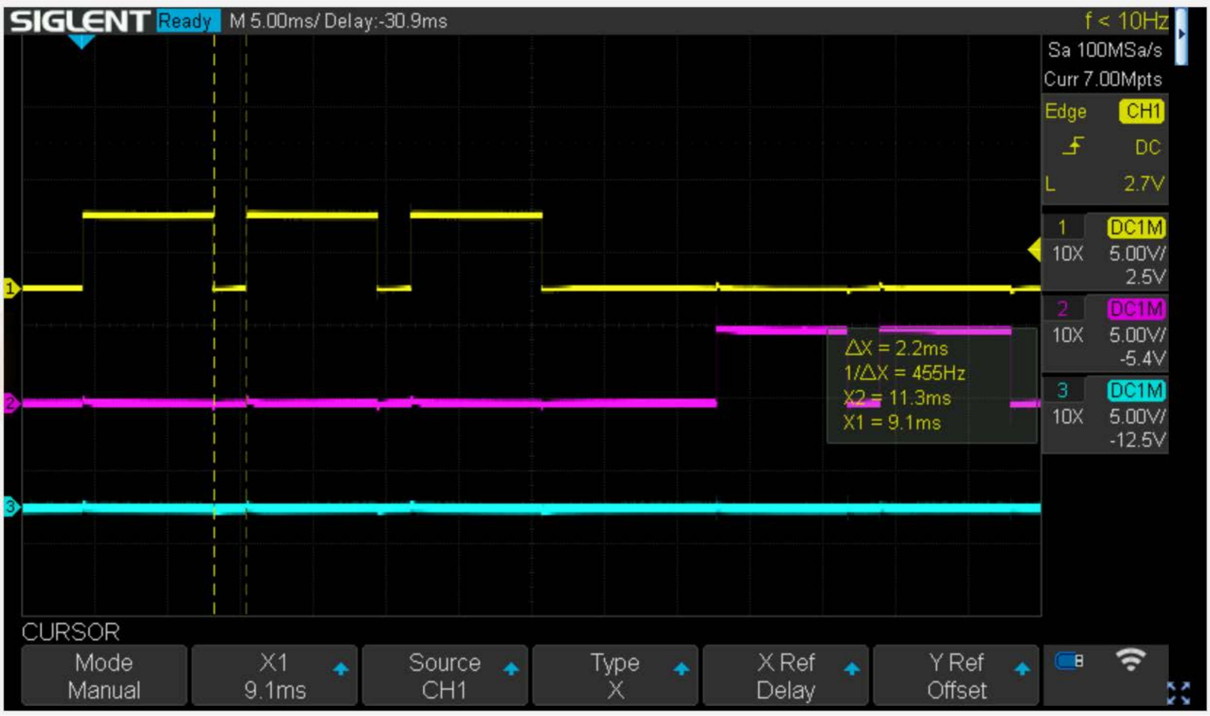
6 0

Should give 9ms pulses (3 up, 3 down) with 1 ms pauses

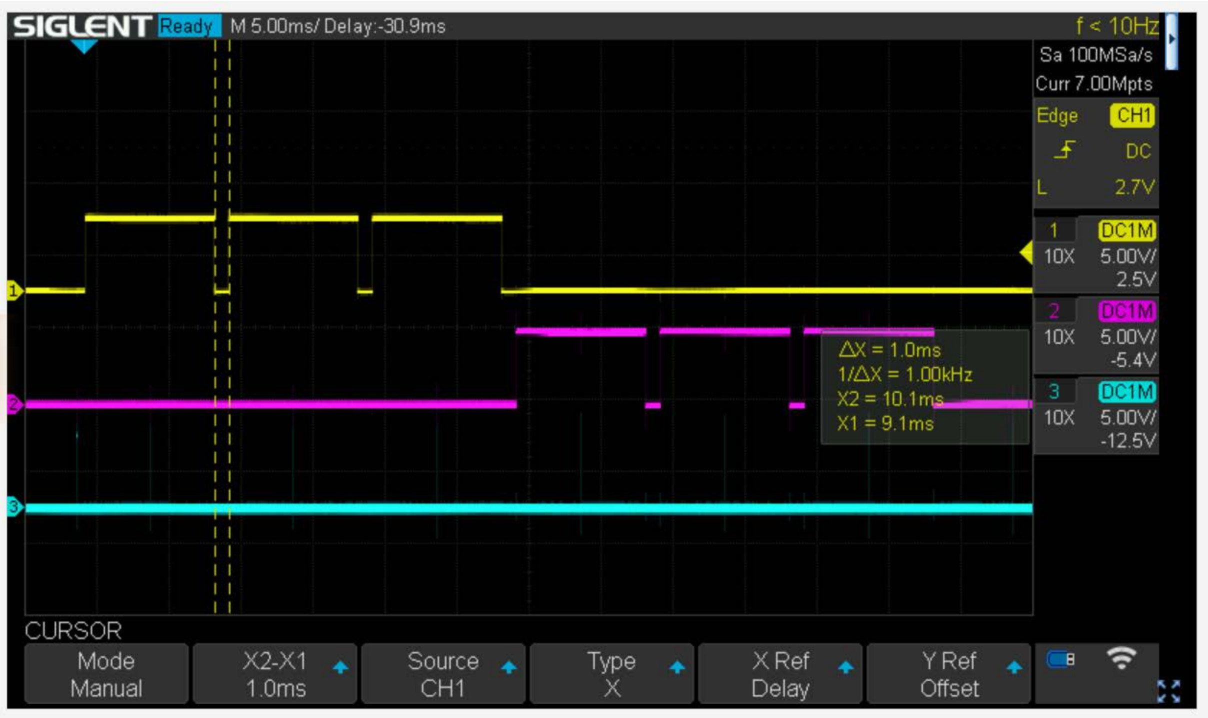
This is what EXECCTC produces:



Pulses are 9ms, but time between pulses is >2ms:



AND the first negative pulse is missing (this changes from run to run and can be worse). It is related to the between-pulse-timing using the loop function. Instead we can use the phase correct PWM feature to generate precise pulse timing for CTC functionality:

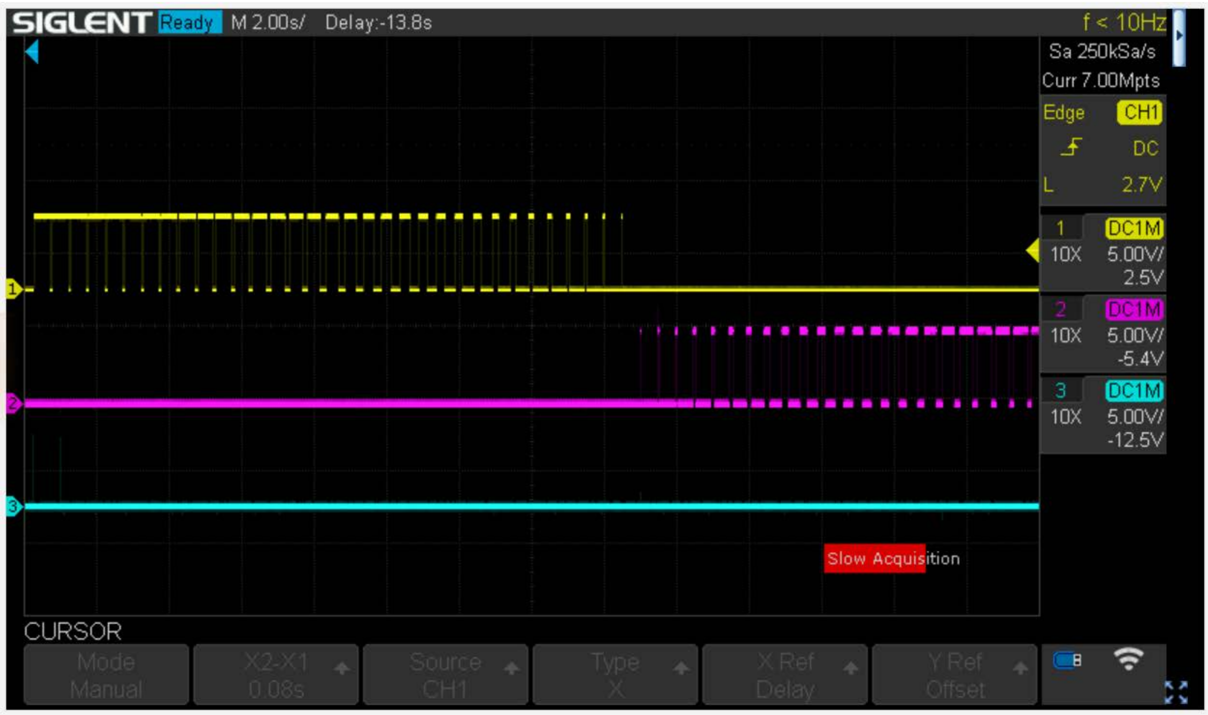


Time between pulses = 1ms, pulse duration = 9ms

This is implemented for a max cTCBinMs of 500ms, which is a good resolution. 2500 CTC entries are allowed, ie giving a max stimulus duration of 2500*0.5s = 1250s > 20 minutes

Also fixed a bug in EXECCTC: the last pulse was always dropped → changed the end of loop statement

Works nicely: sine wave



How it works

We are using a 16 bit timer, timer1 in phase correct PWM mode (mode 10).

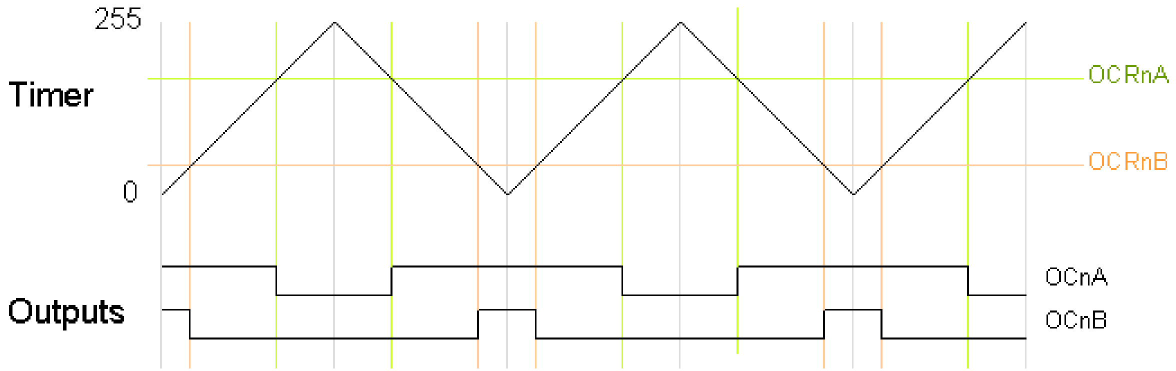
Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV Flag set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, phase and frequency correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, phase and frequency correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, phase correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, phase correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	-	-	-
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Timing is set by the prescaler:

CS12	CS11	CS10	Description
0	0	0	No Clock Source
0	0	1	System Clock
0	1	0	Prescaler = 8
0	1	1	Prescaler = 64
1	0	0	Prescaler = 256
1	0	1	Prescaler = 1024
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

And we are using a prescaler of 64. The Mega2560 runs at 16MHz, prescale by 64 gives a tick per 4us. This is a good divider if we want ms resolution, ie 1ms = 250 ticks.

Phase correct PWM with TOP defined by ICR1 means in one cycle the counter counts from 0 to ICR1 and back again. Now we can define thresholds with OCR1A and OCR1B to generate he pulse. Each pulse is symmetric around a trough.

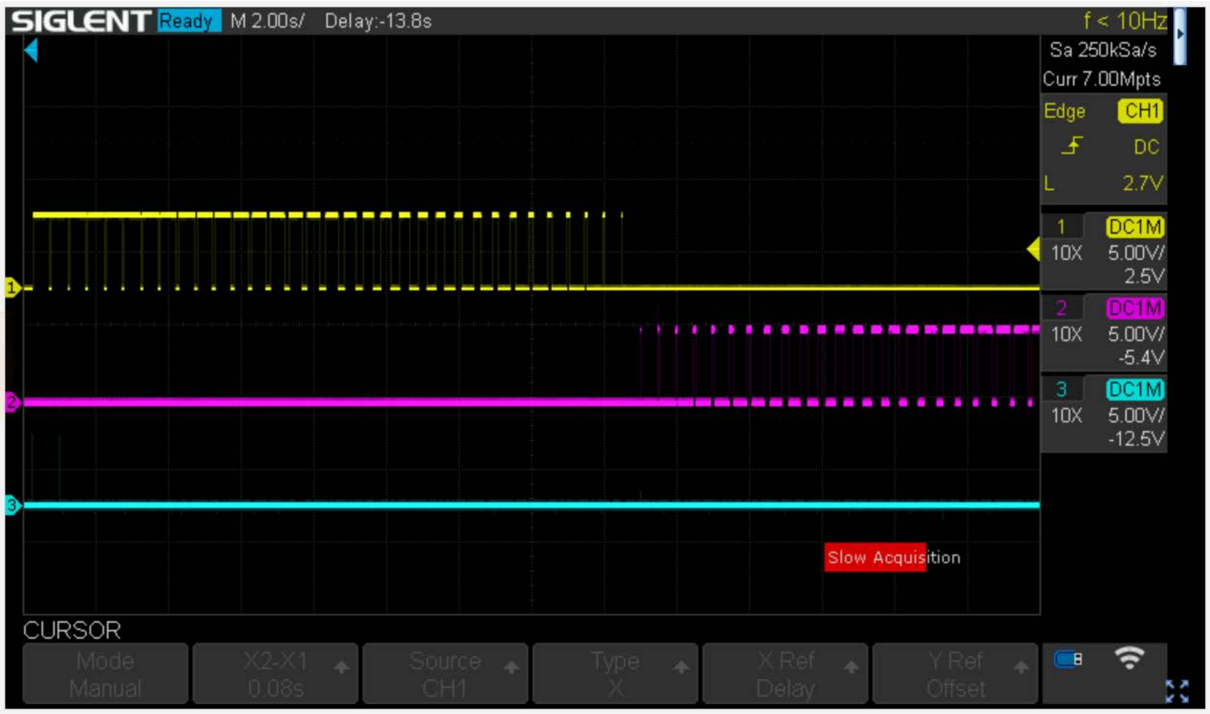


Now how to change pulse width from cycle to cyle? Each time the counter (TCNT1) hits the bottom an interrupt is fired (timer overflow) which we can service using

ISR(TIMER1_OVF_vect)

In this interrupt handler we simply set the new values for OCR1A and OCR1B. Note however, that these only come into effect at the next peak, because OCR1X are double buffered registers.

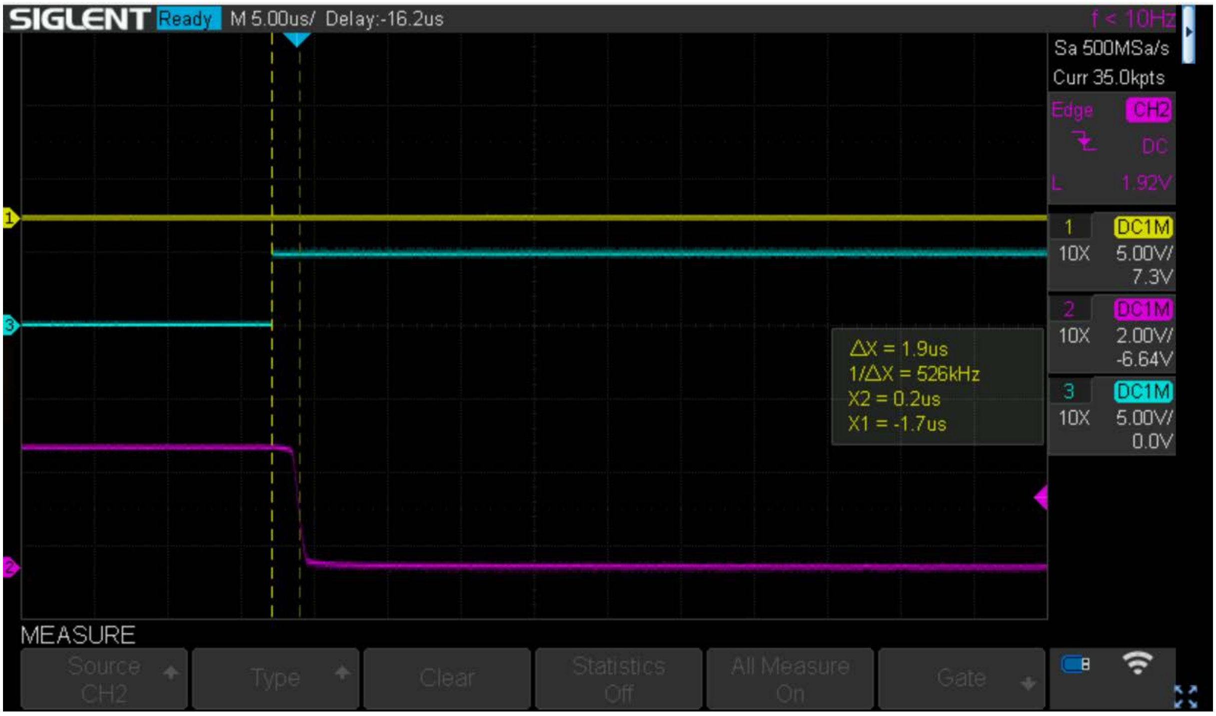
Works nicely: part of a sine wave



How the optocoupler changes the signal

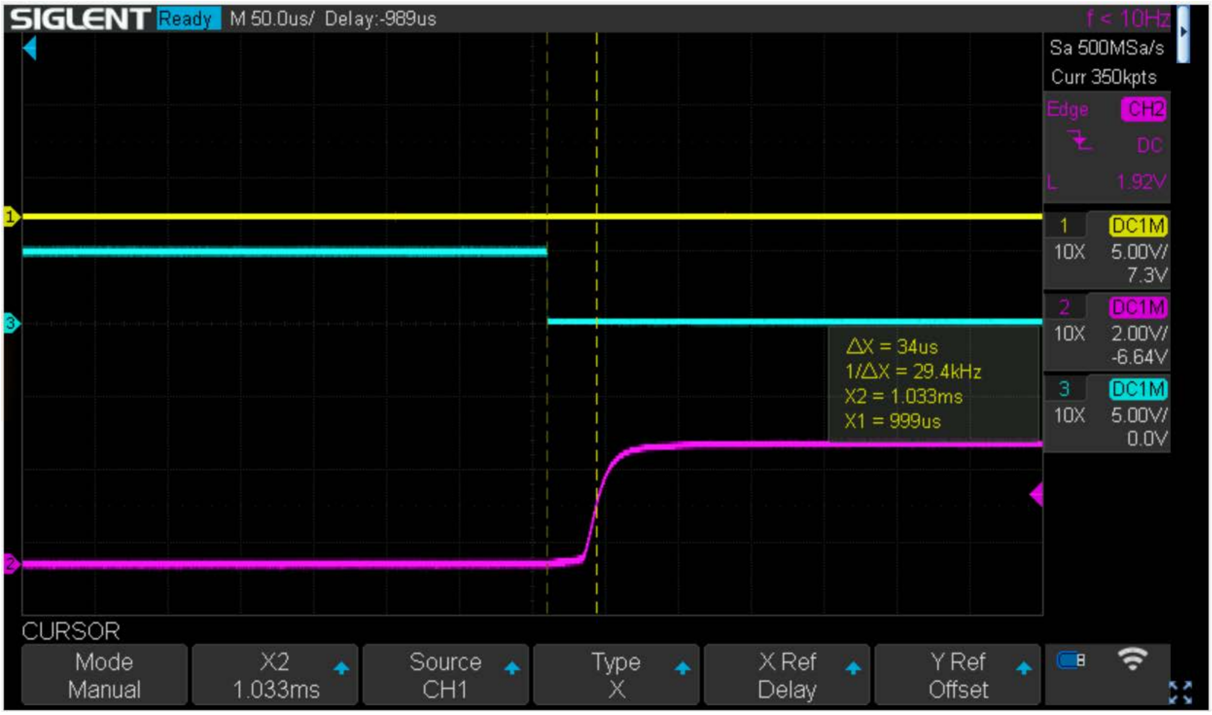
The Optocoupler is a quite low speed PC817:

Onset of pulse:

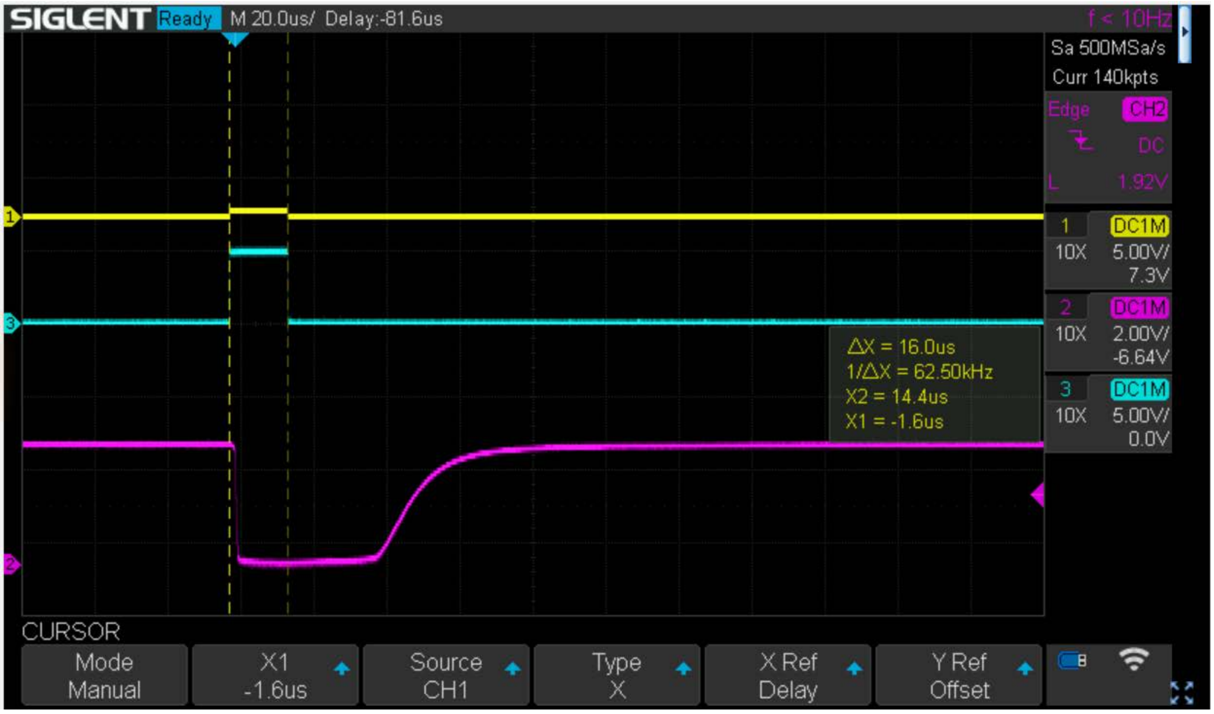


A delay of roughly 2us

How about the offset

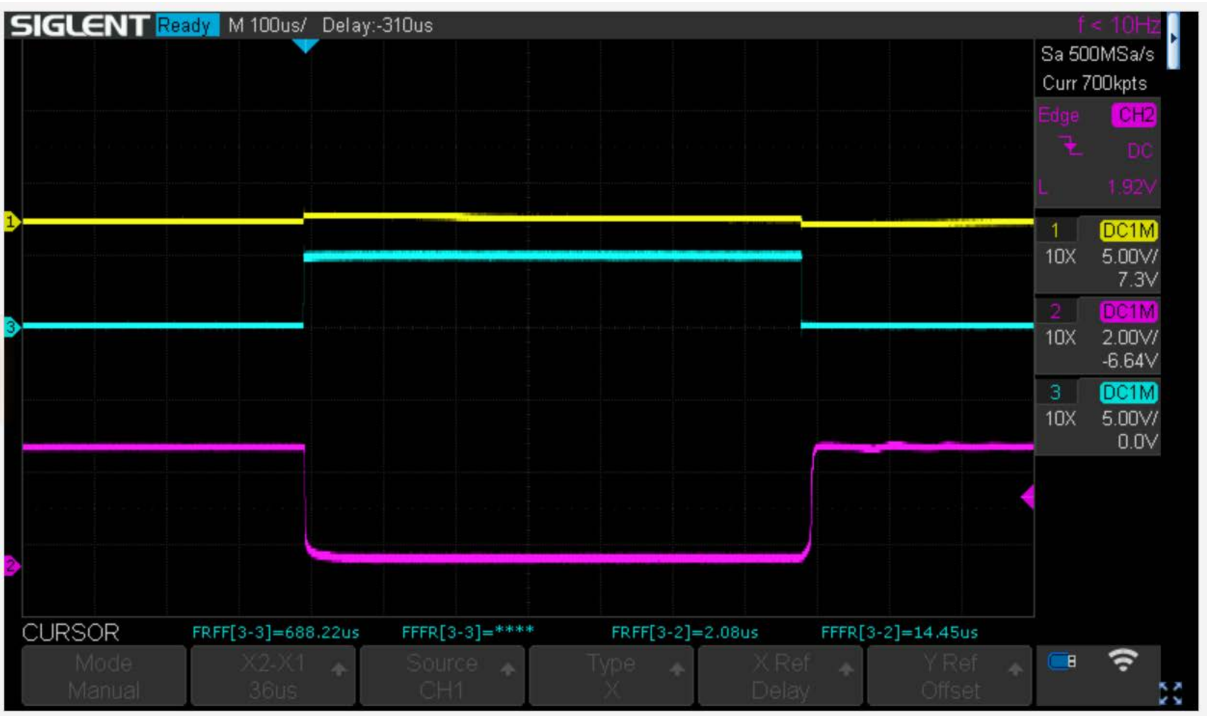


This looks worse with >30us (with a 4R7K pullup), but the slow recovery is related to the large pullup (i.e. slow discharge of capacitance of the photo transistor)



E.g. a 16us pulse gets translated to a 48us pulse with a very badly defined rise. Luckily, we do not operate at that time scale.

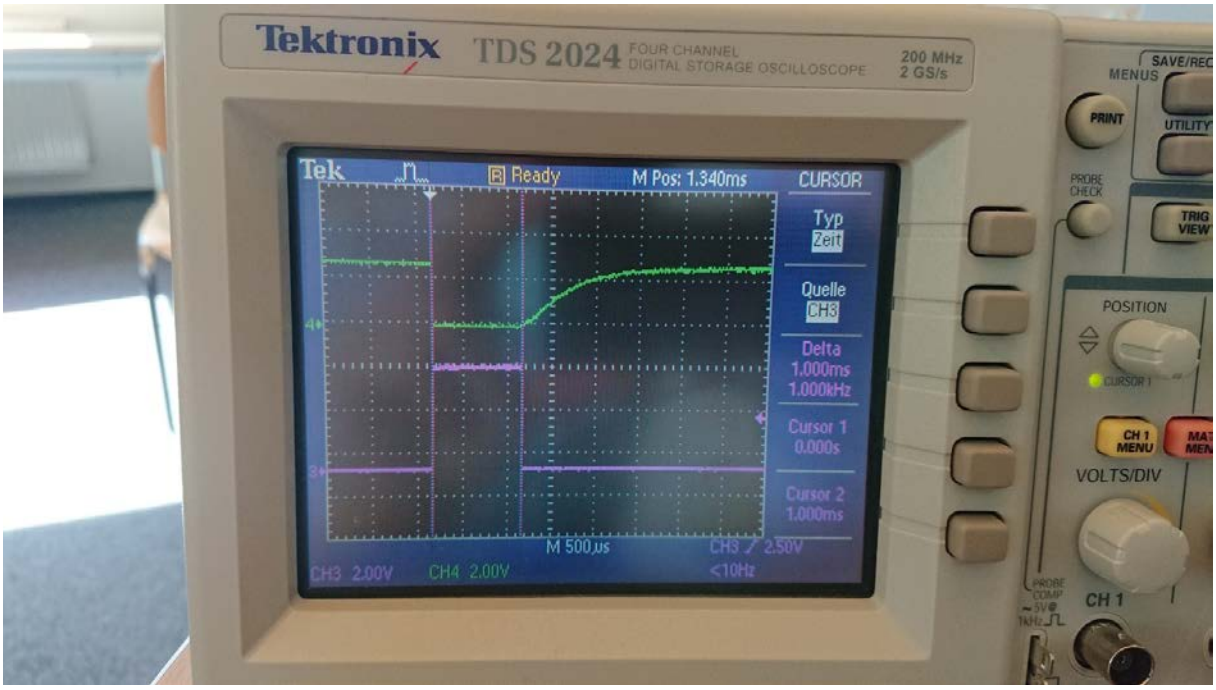
Now this gets better if the pullup gets smaller, using 470R get's us the following



A pretty consistent 2us onset and a 14us offset delay, this is constant at pulses >100us, i.e. in the ms range we are ok and each pulse is roughly 12us too long … We have a timer resolution of 4us, but using count-up/down as in phase correct PWM, the actual resolution is 8us. So we could subtract one tick from every pulse which would bring down the error to 4us, but I guess that is overkill for the thermode…

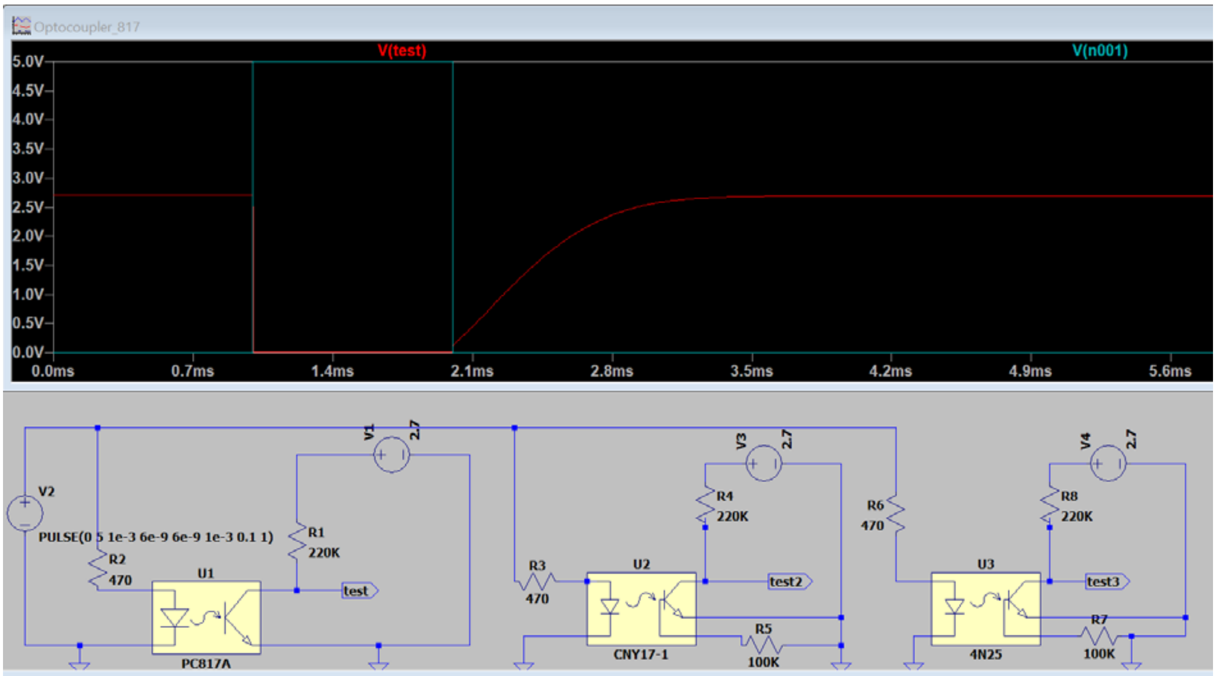
Timing update

Timed everything in the lab with the thermode attached:

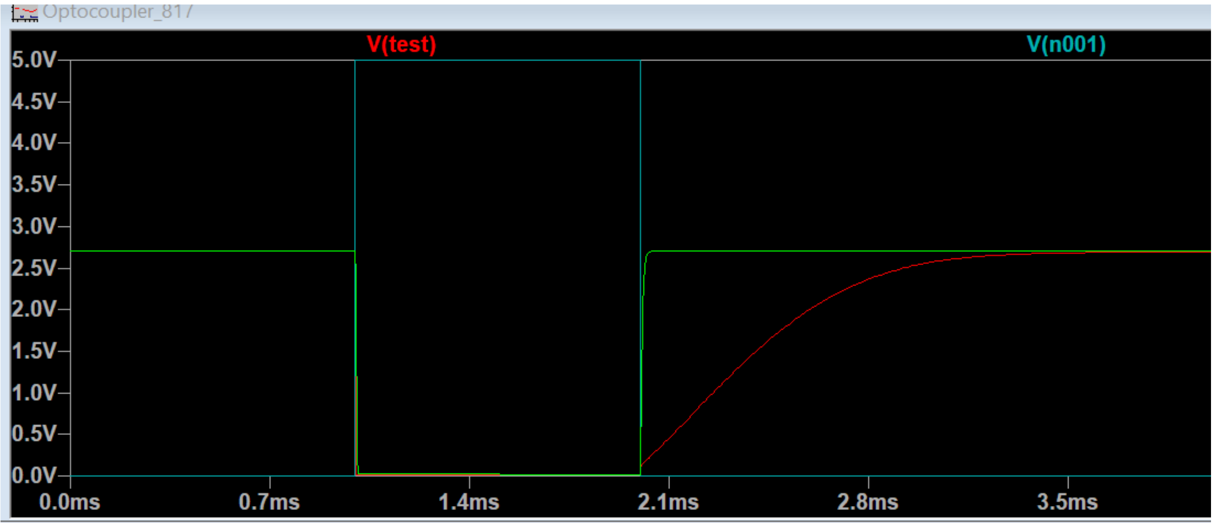


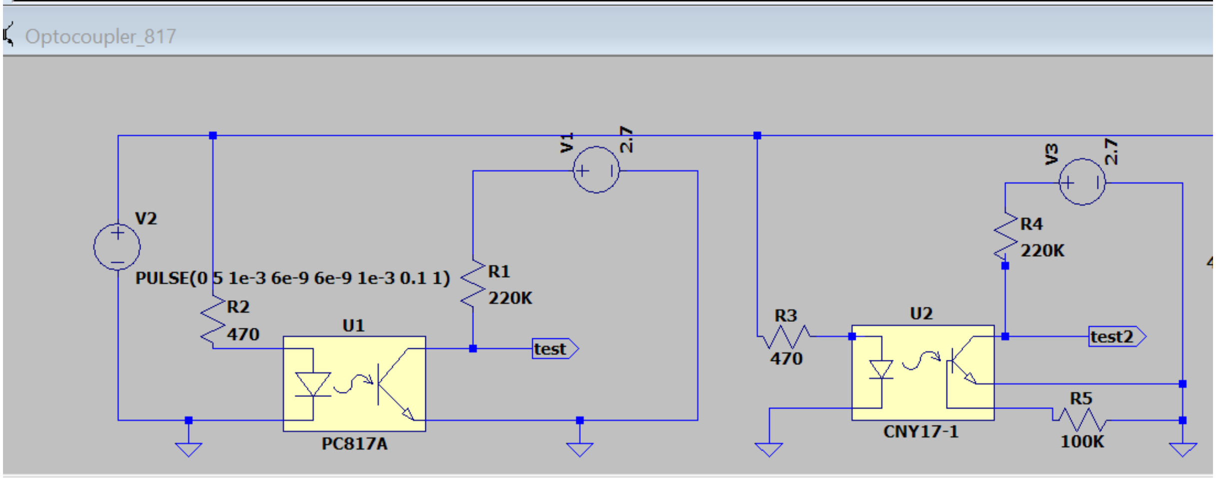
Pretty bad, a pulse of 1ms with heavy bleeding. Back to the drawing board. The idea was to use an optocoupler with an exposed base. This can be connected to GND (100k) to allow to quickly discharge the base capacitance. I had a few CNY 17-3 and looked at it in LTSpice. Based on the time constant, I guessed that the pullup in the thermode is about 220K. Voltage is about 2.7V.

So let's see a simulation of the old PC817



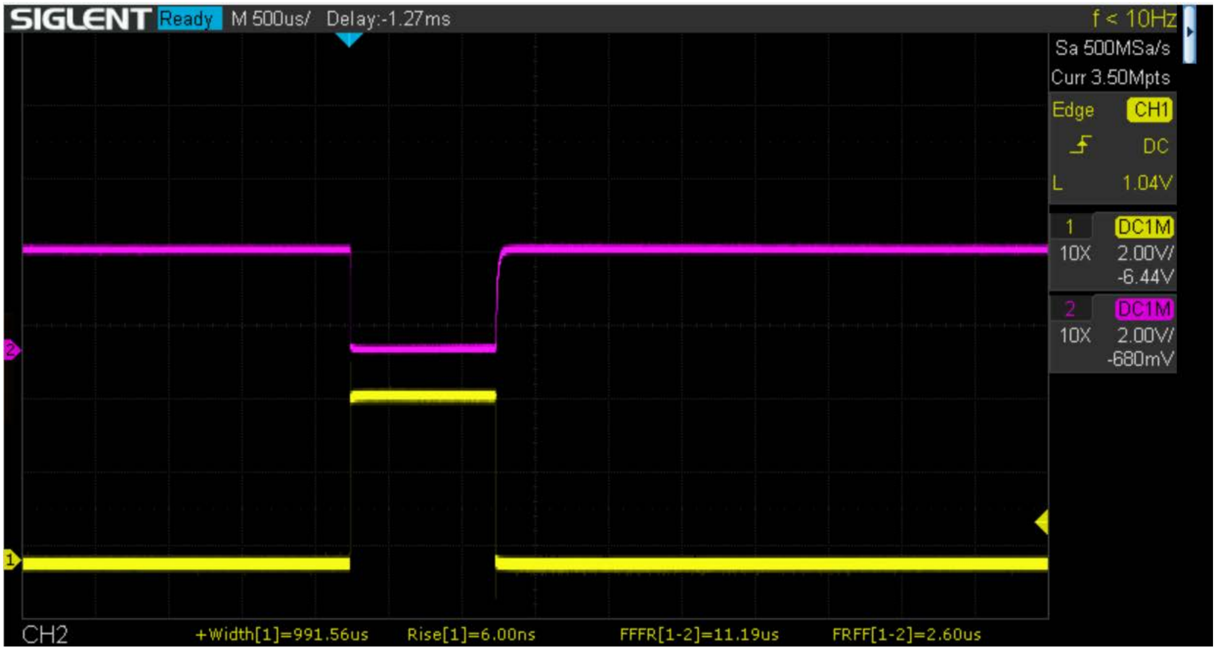
Fits the bill, the red trace is the PC817 output (green-blue Arduino output). Now let's have a look at the CNY17 circuit with the base grounded:





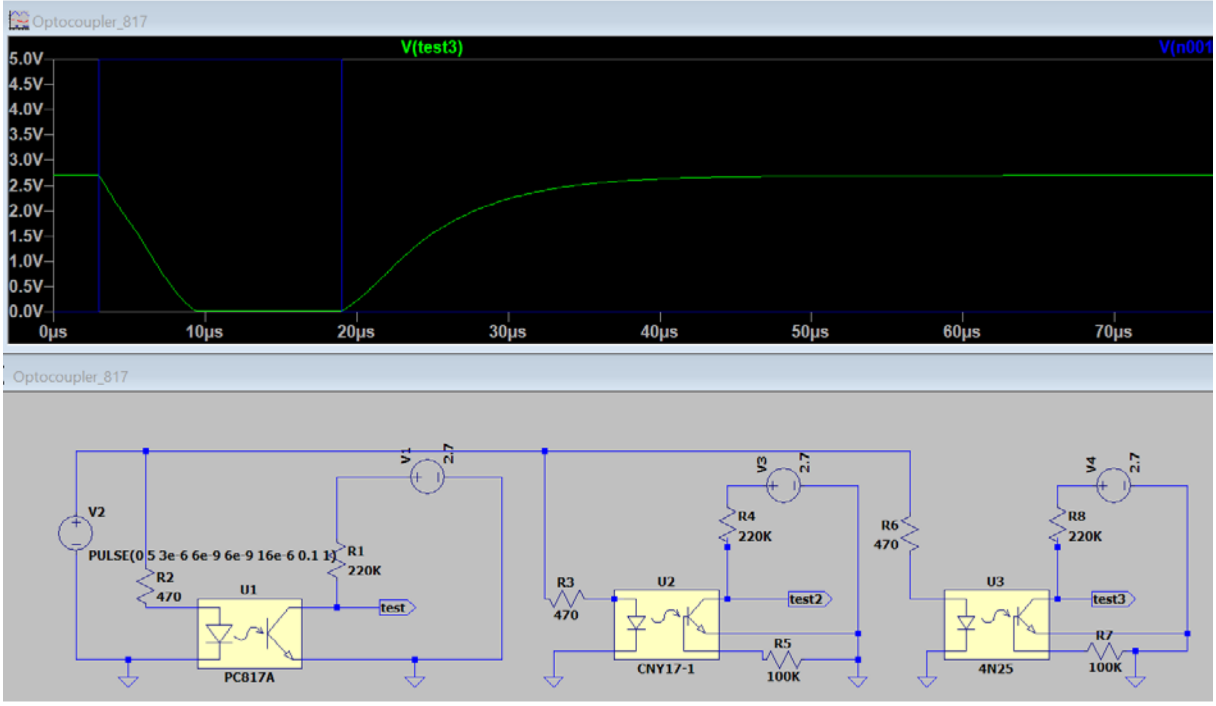
Looks perfect (at that time scale).

So now the reality check, using the new circuit with a CNY17

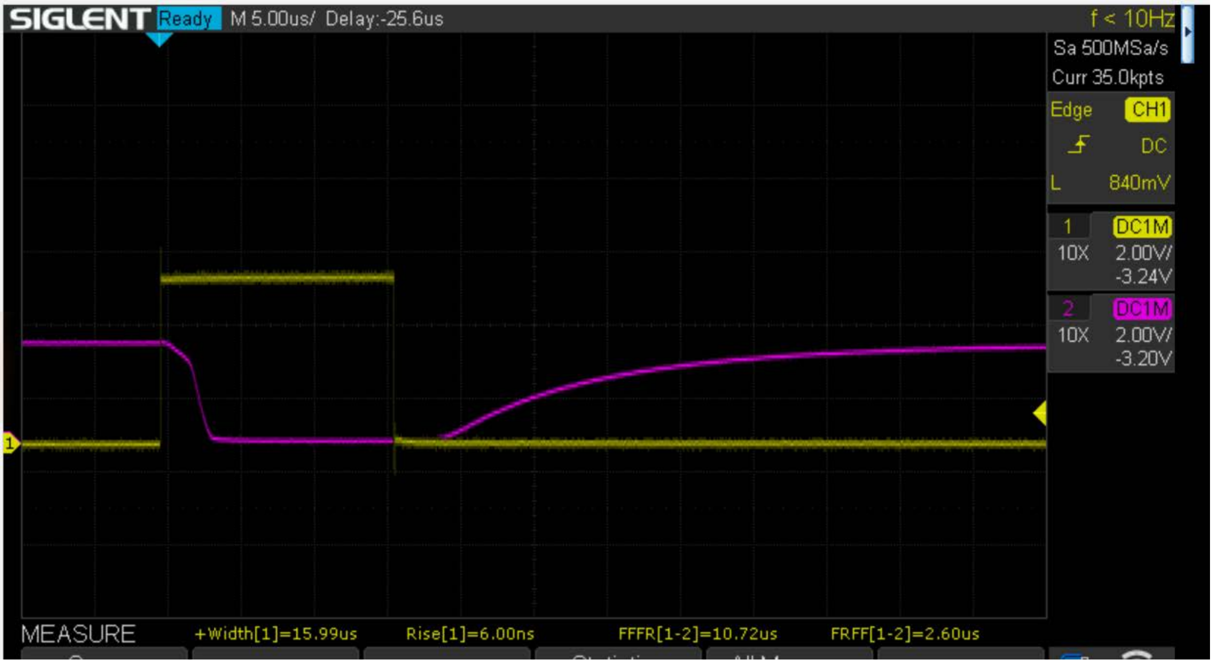


looking good, delay of rising edge is much better, as expected from the LT spice simulation

Finally a very brief pulse of 16us with LTspice:

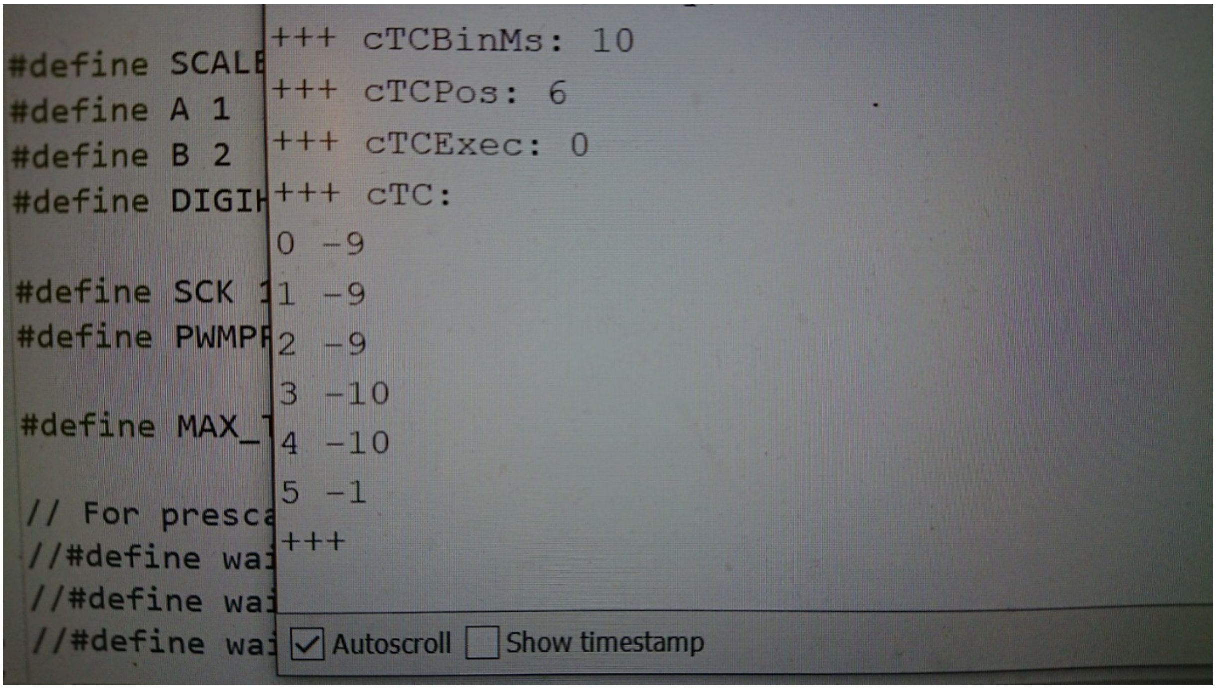


And reality:

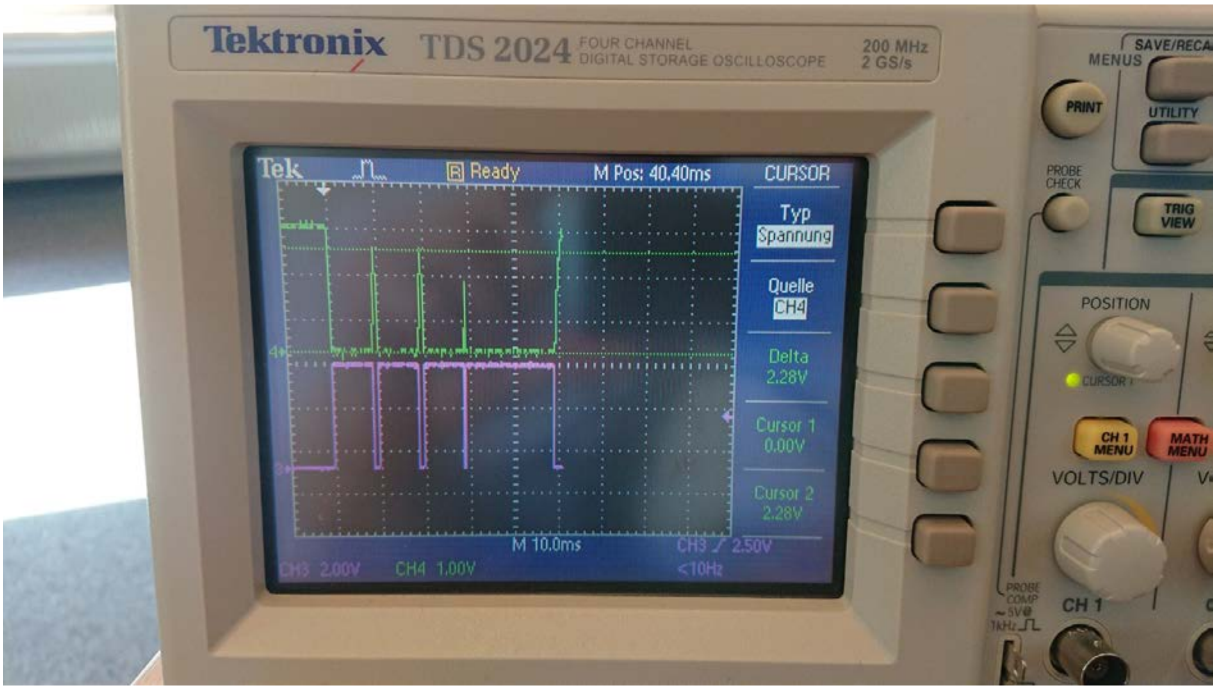


Summary: All thermoinos need an overhaul and get new optocouplers !!

Now a real world example: the following CTC:



On the old circuit (PC817):



Last pulse not shown.

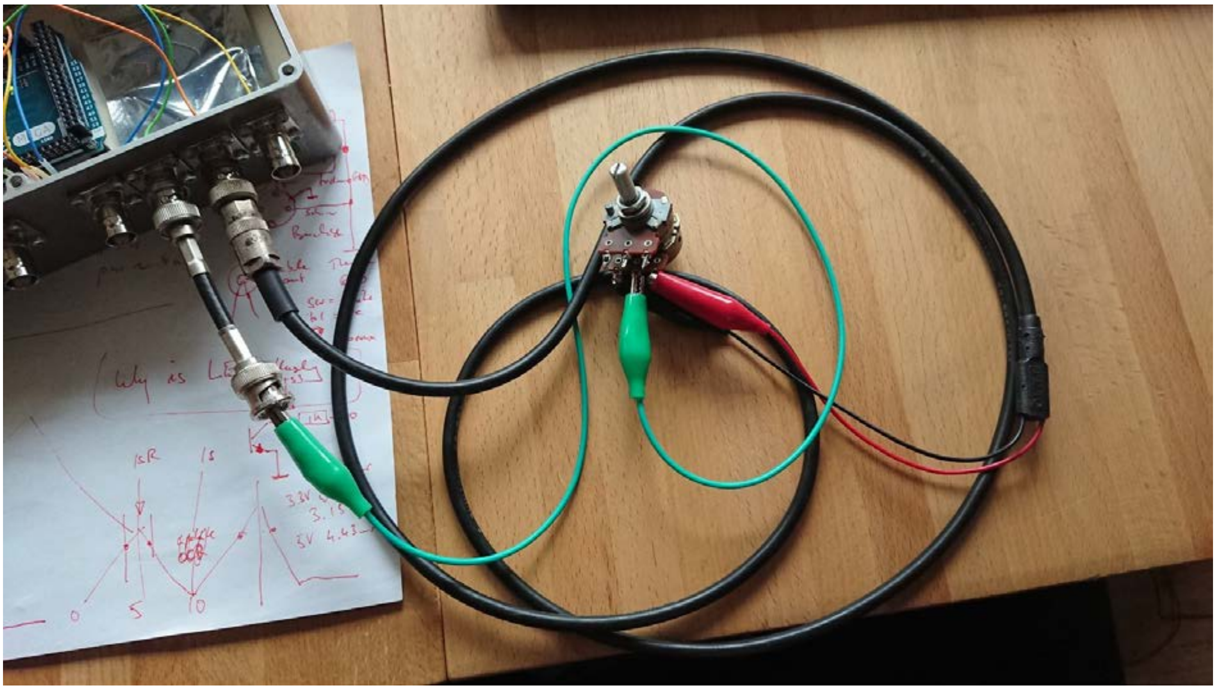
Now with the new circuit:

Looks perfect.

The VAS feature

Finally, I have added an experimental VAS feature. The command READVAS returns the time in ms (like GETTIME) since the Thermoino has been powered-up and a reading of the potentiometer (0..1023). This might interfere with EXECCTC, but will not interfere with EXECCTCPWM.

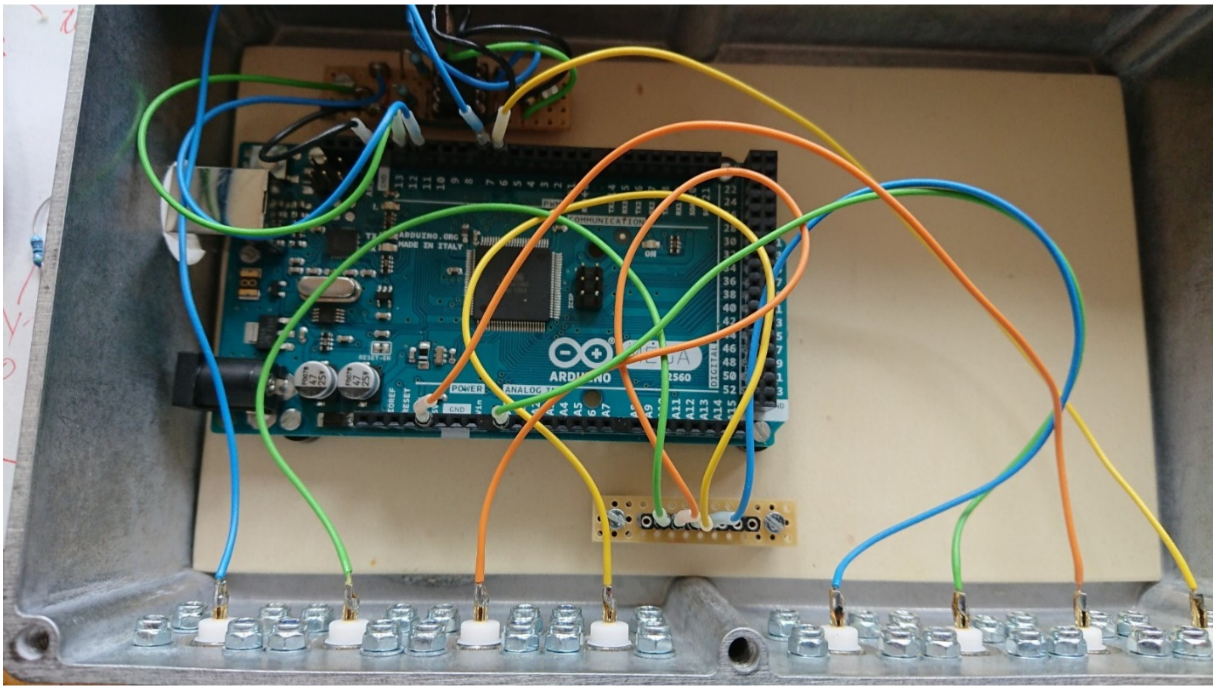
Simply connect a potentiometer (>= 10KOhms) to the following BNCs:

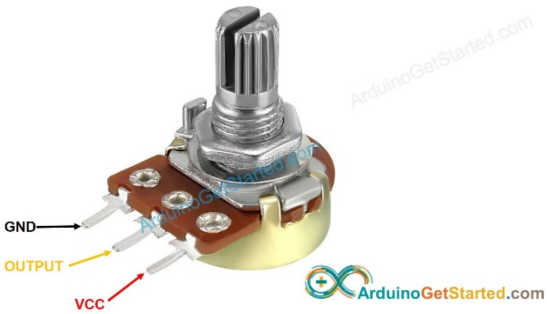


The rightmost has +5V in the middle and GND at the shield (red, black)

The one to the left one (green clip) has the A0 (ADC0) pin in the middle.

This is how it is wired inside LUIGI:





VCC and GND will get connected to the second last BNC. The middle tap of the potentiometer (yellow) gets connected to the third last BNC.

NEVER connect the middle tap of the potentiometer to the second last BNC (this will fry the potentiometer and the Arduino).

The potentiometer needs to be AT LEAST 10K Ohms.