Open Source USB Digital Delay Generator/Gate Project

V1. 3 March 2019

# Overview

This open source project aims to guide the user through building a software-controlled [digital delay generator/gate](https://en.wikipedia.org/wiki/Digital_delay_generator) (DDG) based on the [Teensy 2.0](https://www.pjrc.com/store/teensy.html) (ATMega32) microcontroller for a total cost of approximately USD 100. This project was inspired and tested in working laboratories as part of an [optical cavity ringdown](https://en.wikipedia.org/wiki/Cavity_ring-down_spectroscopy) apparatus[[1]](#footnote-1), and was intended as a low-cost alternative to much more expensive (but also much more capable) commercially available scientific equivalents. We think that other ringdown experimentalists will find this useful. In addition, a portable, USB-powered and programmable DDG may be generically useful in many hobbyist situations where a nicely-conditioned pulse signal is required in response to an analog signal reaching a predetermined threshold condition. For example, in high-speed photography, a sudden increase of an analog light or sound level is often used to trigger the firing of a strobe to capture an event such water drops, balloons bursting, lightning etc.

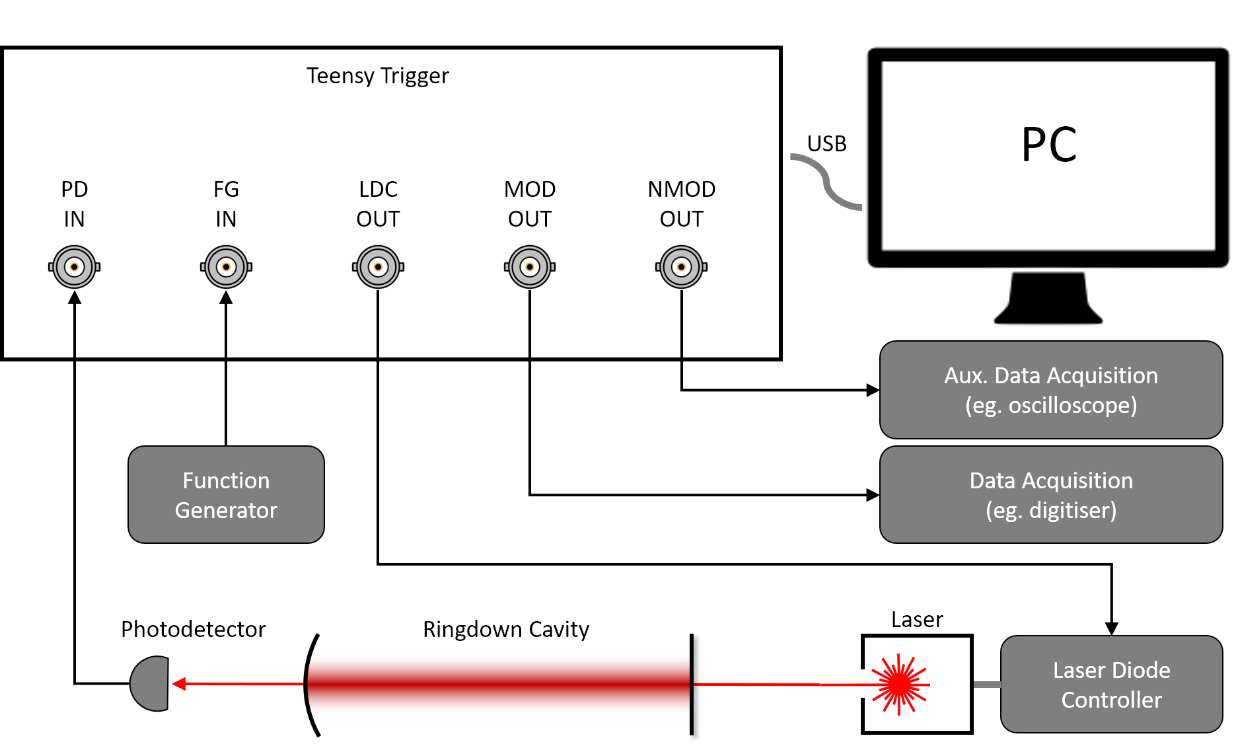


Figure 1. Typical usage of the Teensy Trigger in a cavity ringdown experiment. The circulating optical power in the ringdown cavity is monitored with the photodetector (PD) in transmission. When the analog PD signal rises above a software-settable threshold, square pulses are emitted by the MOD and NMOD outputs (which are complementary to each other) to trigger data acquisition, and the LDC output is blanked (taken to 0V) for a brief duration. When there are no trigger events, the LDC output is a pass-through of the FG input (typically used to provide a DC bias and small dither for the laser).

# About

The concept for this project was conceived by Georg Winkler (University of Vienna, Austria). Gar-Wing Truong designed and maintains this open source project. We thank Crystalline Mirror Solutions LLC for funding the project, and Blake Haist who built the prototype device shown in this document and provided some of the more detailed documentation enclosed. Correspondence should be directed to GWT ([garwing@crystallinemirrors.com](mailto:garwing@crystallinemirrors.com)).

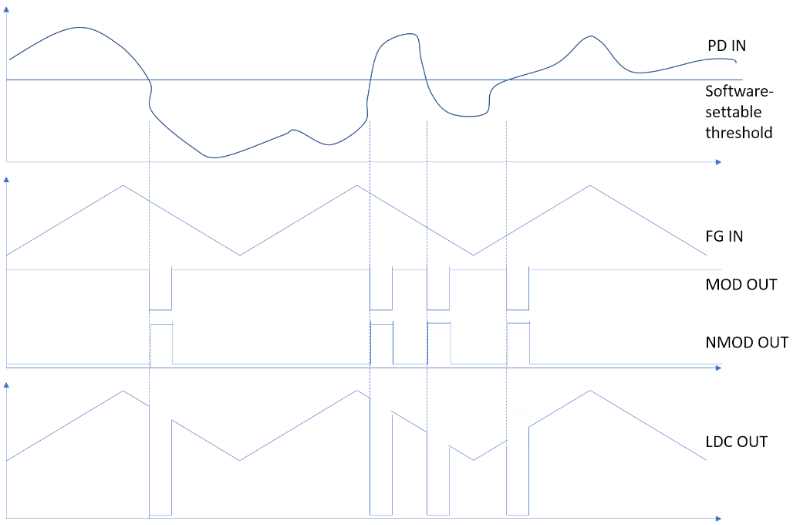
This project is released under the GNU General Public License version 3 (GPL-3.0) license. This means you are free to use and modify this project, but release of the modified project must also be offered under this GNU GPL license alike.

# Specifications and Attributes

Although we have not exhaustively tested the performance of the DDG under a large variety of input and output loads, we think that you will be easily able to replicate the performance we achieved under most reasonable conditions.

|  |  |
| --- | --- |
| Photodiode input (PD IN) |  |
| Range | 0V to 5 V |
| Pre-amp Voltage Gain | 1, 2, 4, 8 |
| Bandwidth | 37 kHz |
| Threshold Voltage |  |
| Range | 0V to 5 V |
| Resolution | 20 mV |
| Modulation Outputs (MOD OUT, NMOD OUT) |  |
| Main and Inverted output into HiZ | 0V, 5V |
| Minimum Output Delay | TBD (< 10 µs) |
| Rise/Fall Time | < 0.1 µs |
| Gated Signal Input/Output (FG IN, LDC OUT) |  |
| Range | 0V to 10V |
| Communication and Power | USB 2 |

A timing diagram is provided below to further illustrate the functionality of the Teensy Trigger.



# Quick Start

## Folder Structure

In the main /ddg/ folder, you will find this ReadMe and license information. The three main components of the project (hardware, teensy software, and python software) are separated into folders.

## Building the Hardware (/ddg/hardware)

A Bill of Materials (BOM.xlsx) is included in Excel format with suggested links for purchase provided for each of the major components. The gerber files for producing the custom PCB is included for those who are familiar with the process. You are welcomed to contact the author to see if there are spare boards that can be provided to you.

All components, with the exception of one IC, are either through-hole or socketed devices and should present no problems for hand-soldering the project. The AD8251 programmable gain amplifier is only available in a surface mount package. It is still possible to hand-solder this device and there are many resources on the internet that describes how this is done. We have also chosen to use a surface-mount-to-DIP adapter to allow this chip to be easily replaced.

Note that the current revision uses a USB DC-DC converter to produce +/-12V. This was used out of convenience when transferring from our breadboard prototype. A more economical option can be used. In fact, the USB power is introduced via the Teensy connection, and not directly on the USB converter.

A schematic of the circuit is included in the folder. After the photodetector signal is amplified by the low-noise programmable amplifier, the signal is input to the Teensy (PE6). An internal analog comparator is used to detect when the input signal rises above a threshold level. This threshold level is generated by using a PWM output (PD7) and externally filtering this with the two-pole low-pass filter.

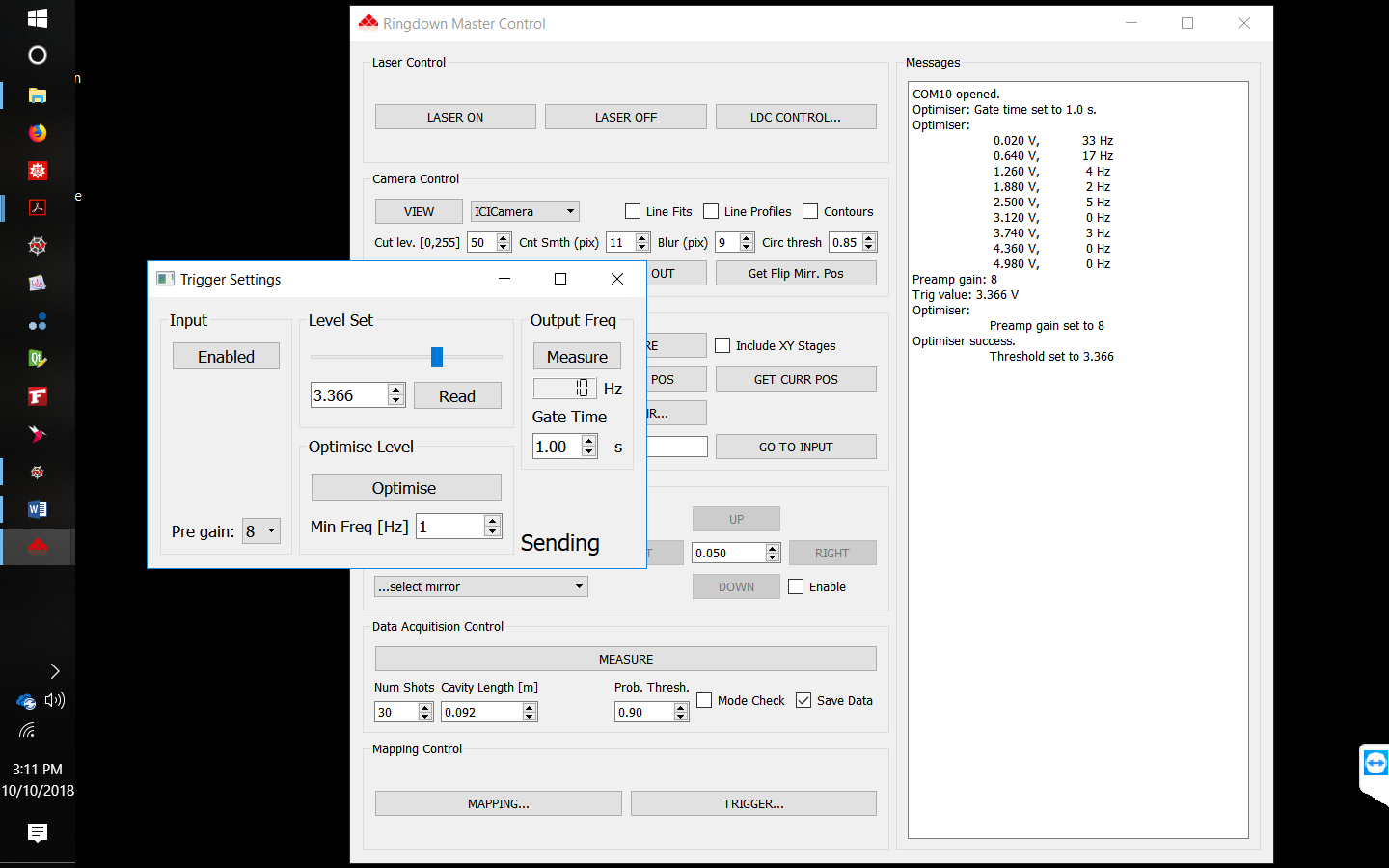
When the comparator output is asserted, the Teensy is programmed to issue the MOD and NMOD pulses (with settable duration and delay). The MOD signal is internally connected to the control pin of a solid-state switch, which grounds out the LDC OUT signal for the duration of the pulse.

## Uploading the Teensy Software (/ddg/teensy)

You will need to download the Arduino development environment and follow the [instructions](https://www.pjrc.com/teensy/first_use.html) for connect the Teensy to a computer using a USB cable. Upload the ino file in the teensy folder to your Teensy – this will be much like uploading an Arduino sketch. Open Windows’ Device Manager to determine which COM port has been assigned to the Teensy.

## Running the Python GUI (/ddg/python)

Download and install your favorite Python 3 distribution. This software was developed and known to work on the [WinPython-64bit-3.4.4.6Qt5](https://sourceforge.net/projects/winpython/files/WinPython_3.4/3.4.4.6/) package. Open the TeensyTriggerSoftware\_v\_a6\_2.py file, and enter the appropriate COM port number for the variable ‘Teensy\_addr’ at line 303. Running the Python script should bring up a GUI like the one shown below.



Input:

* **Enabled**: Push button allowing trigger events to be either ignored or active. The text on the push button shows the current state.
* **Pregain**: This drop-down menu allows the user to select from one of four fixed gains (1,2,4,8).

Level Set

* **Level Slider**: Selects a threshold voltage between 0 and 5 V. This can also be done by entering a value into the spinbox.
* **Read**: User can also push to read the currently-set threshold from the Teensy.

Output Freq:

* **Measure**: Returns the average rate of trigger events, measured over a time specified by the Gate Time. This can be used in conjunction with the Level Slider to set the highest threshold value that still gives reasonably frequent trigger events.
* **Gate Time:** User-settable time period over which the trigger frequency is measured. Note that this will return 0 Hz if the input is disabled.

Optimise Level (beta):

* **Optimise:** Push to automatically find the optimum Pregain and threshold levels (ie lowest pregain and highest threshold) to achieve a minimum trigger rate of Min Freq. This assumes you have a steady-state signal at the input that will remain for a few minutes. You may find this functionality buggy at this stage.
* **Min Freq:** Target trigger rate for optimisation routine.

# Further Documentation (/ddg/docs)

A laboratory-style report is provided by Blake Haist, which details the design principles and operation of the entire project. The novice user is encouraged to explore the entirety of this folder before embarking on the project.

1. Truong, G-W., et. al. “Near-Infrared Scanning Cavity Ringdown For Optical Loss Characterization of Supermirrors”, [ArXiv link when available] [↑](#footnote-ref-1)