Supplementary Material

I. DESCRIPTION OF CIRCUIT IMPLEMENTATION

We present a schematic and detailed description of the digital time delay circuit, and explain the decisions made in designing the circuit and accompanying printed circuit board (PCB).

A. Main Delay Circuit

1. Power Supplies

The circuit requires +5 V and −5 V power supplies for its analog circuitry (the ADC and DAC), as well as a 3.3 V supply for the FIFO. An external power input of 7 V to 16 V is required to power the regulators for these supplies. This power input also directly powers the Arduino Due through its `Vin` pin (see Sec. I A 6), which prevents noise from a computer connected via USB from interfering with the sensitive analog circuitry.

The regulators are connected as follows. The external power input from either the barrel jack or the screw terminal (`V+`) is connected via a reverse-voltage protection P-channel MOSFET (`Q1`, *DMP3099L*) to both the Arduino Due and the +5 V regulator (`U6`, *LM2940CT-5.0*). The +5 V output of the regulator is connected to the 3.3 V regulator (`U8`, *LM1117-3.3*) and the −5 V charge pump inverter (`U15`, *LM2776*).

2. Manually Setting the Delay

The delay circuit may be used standalone, without computer control, by the use of an Adafruit 128x32 I2C OLED Display and a Grayhill 61C series optical encoder. To set the number of delay words, Arduino code enables the use of the encoder to adjust the current digit, and move between digits when the button is pressed. This way, the user interface is easy to use and similar to that of many commercial function generators.

3. Analog to Digital Converter

The *LTC1740* is a 6 Msps, 14-bit sampling analog to digital converter. It has been configured for single-ended operation, though it is capable of accepting differential inputs. The `OF` output of the device has been connected to an indicator LED (`D1`) to indicate when the input is outside of the sampling range of ±2.5 V. The ADC starts conversion on the rising edge of its `CLK` input, which is connected to the `ADC_CLK` signal.

The ADC does not output data to the FIFO immediately. After the initial rising edge of `CLK` that begins conversion, it takes two additional clock cycles and the conversion time ($t_{\text{conv}} \approx 100\,\text{ns}$) to output data. That is, for a clock with period T, data is available after $2T + t_{\text{conv}}$. The consequence of this is that data may only be read on the third clock cycle after conversion started, so the ADC adds three clock periods of delay.

4. First-In-First-Out Memory

The *IDT_72V2105* is a deep First-In-First-Out (FIFO) memory, with 262 144×18 memory organization. This means that we can store up to 262 144 different 18-bit words in the memory.

Initially, the Arduino Due sets control signals and initializes a master reset. After a master reset (MR), the contents of the FIFO memory are cleared and both the write and read pointers are set to zero. The control signals configure the FIFO for First Word Fall Through (FWFT) mode and for serial programming of the flag offsets.

To program the delay, the Arduino Due disables `WCLK` (see Sec. I A 7), then sets $\overline{\texttt{LD}}$ and $\overline{\texttt{SEN}}$ low to initiate flag programming. The flag data are then set on `FWFT/SI` and clocked in on the rising edge of `WCLK`, which is controlled by the Arduino Due. After the flag data have been written, $\overline{\texttt{SEN}}$ and $\overline{\texttt{LD}}$ are set high, the external write clock is enabled, and a partial reset of the FIFO is performed to reset the write and read pointers to zero.

After the delay is programmed and a partial reset is performed, $\overline{\texttt{PAF}}$ is high and the Arduino Due sets its $\overline{\texttt{REN}}$ (override) pin high, so the FIFO $\overline{\texttt{REN}}$ pin is high, mirroring $\overline{\texttt{PAF}}$. The read pointer is disabled and remains at zero, while the write pointer is clocked onward and the FIFO fills with signal data. Once there are more than the programmed number of delay words in the FIFO, $\overline{\texttt{PAF}}$ goes low, enabling reading of the FIFO memory and transferring the read words to the FIFO output pins. The write and read pointers are then both clocked along, maintaining equal distance from each other.

The *74LVC1G08* AND gate (`U12`) was included to allow the Arduino Due to forcefully enable reading, even if the FIFO is not almost full. This permits the utilization of the delay circuit beyond just implementing a fixed delay, as discussed in this paper, by enabling dynamically adjustable delays through clocking `WCLK` and `RCLK` at different frequencies.

5. Digital to Analog Converter

The *AD7840* is a 14-bit digital to analog converter with a 3 V buried Zener reference. This DAC is used to convert the digital signals read from the FIFO into an analog output. While the signal input range for the ADC is ±2.5 V, the DAC output range is ±3.0 V. To correct this discrepancy so that the delay device has unity gain, we use the DAC with a diminished reference voltage. The potentiometer `RV2` is adjusted to divide down the 3 V from

the REF_OUT pin and feed it to REF_IN, setting the DAC output voltage range to ±2.5 V. The *AD7840* datasheet warns that doing this may result in degraded performance from the part, but this has not been observed.

After power-on, Arduino Due toggles the $\overline{\text{CS}}$ and $\overline{\text{LDAC}}$ pins for proper startup and to configure the DAC for parallel data loading. When reading data from the FIFO, the DAC is clocked at $\overline{\text{WR}}$ by RCLK.

Since the DAC has both an input latch and an output (DAC) latch, and given the timing of the signal on $\overline{\text{WR}}$, DAC conversion adds another two clock periods of delay.

Like many R-2R DACs, the *AD7840* output may exhibit glitch-impulses on major carry transitions, such as during zero-crossings of the input. This necessitates the use of an external low-pass filter (LPF), the cutoff of which depends upon the signal frequencies the device is intended to delay. For some applications, a simple one-pole filter will suffice, such as that formed by R6 and C34. For other applications, a fourth-order or higher Bessel LPF is recommended, because of its maximally flat group delay.

6. Arduino Due and External Control

The Arduino Due was chosen as a microcontroller for this circuit because of its speed and convenient digital header with sufficiently many pins. Our PCB is designed so that the Arduino Due simply plugs on to the bottom, and may be easily replaced or removed to be programmed. The Arduino Due sends all control signals to the ADC, FIFO, DAC, optical encoder, and OLED screen, and accepts serial commands from a computer.

The start trigger (STRIG) pin is available on the BNC connector J7. This signal is low during FIFO programming and goes high when the external write and read clocks are enabled, allowing a connected oscilloscope to trigger and view the beginning of delayed signals.

7. Clock Control Multiplexers

Three *SN74LVC2G157* multiplexers (MUXes) are used to give the Arduino Due precise control over clock signals. MUX U2 controls the write clock for the FIFO, allowing the Arduino Due to select either the external clock XWCLK or the Due-controlled clock signal WCLK_S, which is used for programming. Analogously, MUX U3 allows the ADC to be clocked by either XWCLK or ADC_CLK_S, which allows the ADC and FIFO to read inputs independently (see Sec. I B). Finally, MUX U4 is included to allow operation with separate write and read clocks. It is normally controlled to always select XWCLK, so that the write and read clocks will be the same. The strobe input $\overline{\text{G}}$ is also used by the Arduino Due to disable FIFO reading and DAC conversion during programming. The *74LVC1G04* inverter (U7) allows for the correct logic control of the strobe input at speeds faster than those from using a separate Arduino Due PROG pin.

B. Programming the Initial History

The initial memory of the FIFO may be programmed with 12-bit resolution by the Arduino Due. To do so, a serial command including the number of words of memory to write is sent to the Arduino Due. The buffer of 16-bit data is read over serial, MSB first, in 1 kB blocks. As they are read, each block of data is programmed into the FIFO memory. To do this, an analog switch (Sec. I B 1) switches the input to the ADC from the external signal input to the output of programming signal operational amplifiers (Sec. I B 2), which rescale and offset the programming signal from the PROG_D (DAC1) pin on the Arduino Due. The Arduino Due writes a data word to PROG_D and then uses the MUXes to clock both the ADC and FIFO in order to write the word into the FIFO. Once the specified number of words have been programmed (making sure to account for the three-word offset in the ADC output), the Arduino Due waits for a start command. The Arduino Due then sets the delay to the number of words programmed, switches the analog switch to the signal input, and enables the external clocks again. This start event also sends a start trigger signal, as described in Sec. I A 6.

1. Analog Switch

The *TS12A12511* analog switch (U1) was chosen for its speed and low on-resistance. Either the input signal from J1 on the normally-open pin (NO) or the programming signal PROG_ADC on the normally-closed pin (NC) are connected to the COM pin and to the ADC, as determined by the $\overline{\text{PROG}}$ control signal.

2. Programming Signal Operational Amplifiers

The Arduino Due's 12-bit DAC output (PROG_D) ranges from 0.55 V to 2.75 V, but the ADC input range is ±2.5 V. The *TL082* dual JFET-input operational amplifier (U16) has the low noise and large slew rate needed to offset and rescale the programming signal as needed, without distortion. A *LM4041LP-ADJ* adjustable voltage reference (U14) provides the offset voltage, which is adjusted precisely in the needed range by RV1. Similarly, RV3 and adjoining resistors provide stable adjustment of the gain of the rescaling circuit. The input filter formed by R6 and C34 attenuates Arduino Due DAC glitch-impulses and logic switching noise.

C. Printed Circuit Board Design

The printed circuit board layout for the circuit is vital to the performance of the circuit. A four-layer controlled-impedance board was used, with solid internal ground and 3.3 V power planes. Analog components such as the ADC and DAC were placed so that return currents

from the FIFO to the power supply section are avoided, thus preserving the integrity of analog reference voltages. Bypass capacitors were used frequently to provide a power reservoir and a path for signal return currents. For example, C22 provides a low-inductance path for RCLK return current. Input and output signals from BNC connectors and the WCLK and RCLK signals are conducted along appropriately-sized micro-strip transmission lines. The 50 Ω input lines are end-terminated, and the FIFO clock lines are source-terminated. Components and vias were laid out to minimize trace length, as well as to avoid breaking up the ground or power planes.

D. EDA Files

The schematic and PCB for the circuit were created with KiCAD. The associated files and other documentation may be found at `https://github.com/lucasilling/AJP_TimeDelay_Git.git`.

II. SOLUTION OF A SIMPLE DDE

Solutions of

$$\dot{y} = -\begin{cases} -1 & \text{if } y(t-\tau) < 0 \\ 1 & \text{if } y(t-\tau) \geq 0 \end{cases}, \tag{1}$$

with $\tau = 1$, i.e. Eq. (2) in the paper, and some of their properties are known [1, 2]. For example, Fridman et al. [2] show that there exist periodic solutions with periods $4/(4p+1)$ for each integer $p \geq 0$ and that the slowly oscillating solution ($p = 0$) is non-asymptotically stable, whereas the fast oscillating periodic solutions ($p \geq 1$) are unstable. For completeness of presentation we will sketch the proof for this statement here by adding to the discussion of slowly oscillating initial functions an explicit construction and analysis of solutions that arise from initial functions that are fast oscillating. We do so by deriving a discrete-time map that generates all fast-oscillating solutions of Eq. (1) and then analyzing this map using linear algebra.

To start, let us repeat the definition of slowly oscillating and make precise the term fast oscillating: A continuous function $y : \mathbb{R} \to \mathbb{R}$ is called *slowly oscillating* at t if either $|y| > 0$ on $[t-\tau, t]$, or y has precisely one zero at $t^* \in [t-\tau, t]$, $\dot{y}(t^*)$ exists and $\dot{y}(t^*) \neq 0$. A continuous function $y : \mathbb{R} \to \mathbb{R}$ is called *fast oscillating* at t if y has at least two zeros $t_1^*, t_2^* \in [t-\tau, t]$, $\dot{y}(t_i^*)$ exists and $\dot{y}(t_i^*) \neq 0$ for $i = 1, 2$.

For DDEs, the object of interest at each instant of time is a function, one that specifies the value at the current time and values at all past times within the delay. Accordingly, the following notation is common [3, 4]: Denote with $\mathcal{C}([-\tau, 0], \mathbb{R})$ the Banach space of continuous functions mapping the interval $[-\tau, 0]$ ($\tau > 0$) into $\mathbb{R}$. For each fixed t, $y \in \mathbb{R}$ is a scalar real number, as usual, whereas

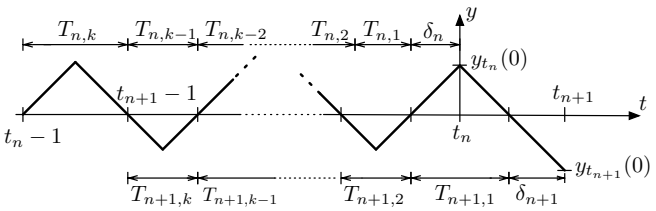

FIG. 1. Construction of the time shift map for the frequency preserving case.

we let y_t designate the function in $\mathcal{C}([-\tau, 0], \mathbb{R})$ given by $y_t(\theta) = y(t+\theta)$, $\theta \in [-\tau, 0]$. We call $y_t(0) = y(t)$ the head point at time t, while $y_t(\theta)$ with $-\tau \leq \theta < 0$ is the associated history segment.

Integration of Eq. (1) from $t = 0$ to $t = \tau$, starting with any continuous initial function, will result in a solution y on $[0, \tau]$ that is piecewise linear with slopes of magnitude one. By further extending the integration by an appropriate amount into the future, to some time that we call t_0, one can always obtain a solution y that is not only piecewise linear on $[t_0 - 1, t_0]$ but is also zero at the beginning of the memory interval, i.e. at time $t_0 - 1$, and has positive slope at that instance of time. Without loss of generality, we may therefore assume such a waveform to be the initial function whose asymptotic behavior we seek to understand.

An example of such a waveform is shown in Fig. 1, where instead of t_0 the current time is denoted by t_n, which represents the nth iterate, as will be discussed below, with the initial function having iteration number $n = 0$. Whereas the current time is t_n (t_0 for the initial function), the function to the left of t_n represents the memory associated with the time delay of length $\tau = 1$. To integrate further it suffices to know the head point $y_{t_n}(0)$ and whether the history segment $y_{t_n}(\theta)$ is positive or negative for a given $\theta \in [-1, 0]$. This means that it is enough to keep track of zero crossings, which we do by defining k time-intervals between crossings, denoted by $T_{n,j}$ with $T_{n,j} > 0$ and $j = 1 \ldots k$. We refer to the $T_{n,k}$-interval when we mean the time interval between zero crossings most distant in the past yet within the memory at time t_n, i.e. times $t \in [t_n - 1, t_n - 1 + T_{n,k})$. The time between the most recent zero crossing in memory and the current time is denoted by δ_n. In terms of this notation, the assumed waveform satisfies

$$y_{t_n}(-1) = 0 \tag{2a}$$

$$y_{t_n}(\theta) > 0 \qquad -1 < \theta < -1 + T_{n,k}. \tag{2b}$$

The fact that the delay is constant and equal to one implies the identity $1 = \sum_{j=1}^{k} T_{n,j} + \delta_n$.

The aim is to construct a map that adds a linear piece to the solution and shifts y_t forward in time along the solution. Repeated application of that map will give the entire solution.

At time t_n, y is positive on the $T_{n,k}$-interval by assumption, which implies that Eq. (1) reduces to $\dot{y} = -1$ up to

time $t_{n+1} = t_n + T_{n,k}$, making integration simple. The known solution y on $t \in [0, t_n]$ is extended by the piece

$$y(t) = y(t_n) - (t - t_n) \qquad t \in (t_n,\, t_{n+1}]. \tag{3}$$

After the shift, the history segment $y_{t_{n+1}}(\theta)$ can either have the same number k of zero crossings as the history segment $y_{t_n}(\theta)$ before the shift, if the new straight-line solution piece in Eq. (3) crosses the $y = 0$ axis, or it has $k - 1$ zero crossings. It is important to note that the number of zero crossings can never increase. If the number of zero crossings continues to decrease under repeated application of the map, then, after a sufficient number of iterations, $y_t(\theta)$ will have just one zero crossing, therefore be a slowly oscillating function and converge to the period-four solution (see Sec. III A). It follows that convergence to the period-four solution is avoided only if there exist frequency preserving solutions, i.e. solutions that retain the number k of zero crossings for some k.

We therefore look at frequency-preserving solutions. For these solutions, the head point and the δ_n interval are related by $y_{t_n}(0) = (-1)^k\, \delta_n$ because for frequency-preserving solutions each straight line-solution piece has to cross the zero axis and the $T_{n,k}$ interval is positive by assumption. Utilizing this relation, we may write the condition for the map that shifts the history segment $y_{t_n}(\theta)$ to the segment $y_{t_{n+1}}(\theta)$ to be frequency preserving as

$$0 < (-1)^k\, \delta_n < T_{n,k}. \tag{4}$$

Note that condition Eq. (4) implies that the number k of zero crossings on the unit interval $(t_n - 1,\, t_n]$ has to be even.

If Eq. (4) is satisfied, then the map from t_n to t_{n+1} is given by (see Fig. 1)

$$T_{n+1,1} = 2\delta_n \tag{5a}$$
$$T_{n+1,j} = T_{n,j-1} \qquad j = 2, \ldots, k. \tag{5b}$$
$$\delta_{n+1} = T_{n,k} - \delta_n \tag{5c}$$

This can be expressed conveniently in vector notation as the linear difference equation

$$\mathbf{v}_{n+1} = \mathbf{A}\, \mathbf{v}_n, \tag{6}$$

where the constant matrix $\mathbf{A} \in \mathbb{R}^{(k+1)\times(k+1)}$ and vector $\mathbf{v} \in \mathbb{R}_{\geq 0}^{(k+1)}$ are, respectively,

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 2 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & & \ddots & & \vdots & & \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -1 \end{pmatrix}, \quad \mathbf{v}_n = \begin{pmatrix} T_{n,1} \\ T_{n,2} \\ T_{n,3} \\ \vdots \\ T_{n,k} \\ \delta_n \end{pmatrix}. \tag{7}$$

Whereas the function y_{t_n} associated with vector $\mathbf{v}_n$ satisfies the assumptions Eq. (2), the function $y_{t_{n+1}}$ associated with the vector $\mathbf{v}_{n+1}$ satisfies Eq. (2a) but not Eq. (2b) because y has negative values on the $T_{n+1,k}$-interval. We therefore integrate again, this time solving $\dot{y} = 1$, to obtain the solution piece

$$y(t) = -\left[(-1)^k\, \delta_{n+1} - (t - t_{n+1})\right] \tag{8}$$

on the time interval $t \in [t_{n+1},\, t_{n+2}]$, where we made use of the relation $y_{n+1} = (-1)^{k+1}\, \delta_{n+1}$. It is straightforward to check that the time-shift map retains the number of crossings k, if and only if the condition for frequency preservation (Eq. (4) with $n \to n + 1$) is satisfied. Furthermore, the map is again given by the linear difference equation Eq. (6), such that

$$\mathbf{v}_{n+2} = \mathbf{A}^2\, \mathbf{v}_n. \tag{9}$$

The solution associated with vector $\mathbf{v}_{n+2}$ satisfies all the assumptions of Eq. (2). Any frequency preserving solution can therefore be obtained by iterating Eq. (6).

Since Eq. (6) is linear, any solution can be written in terms of the eigenvectors $\hat{\mathbf{e}}_i$ and eigenvalues λ_i of $\mathbf{A}$. The eigenvalues λ of $\mathbf{A}$ are roots of the polynomial

$$0 = (1 + \lambda)\, \lambda^k - 2. \tag{10}$$

There is a single eigenvalue equal to one

$$\lambda_0 = 1 \tag{11}$$

with eigenvector

$$\hat{\mathbf{e}}_0 = \frac{1}{2k+1}\, (2,\, 2,\, 2,\, \ldots,\, 2,\, 1)^{\mathrm{T}}, \tag{12}$$

where $^{\mathrm{T}}$ denotes the transpose and the normalization has been chosen such that the dot product of $\hat{\mathbf{e}}_0$ with the vector $\mathbf{n} = (1, 1, \ldots, 1)^{\mathrm{T}}$ is equal to one ($\hat{\mathbf{e}}_0 \cdot \mathbf{n} = 1$).

All other eigenvalues λ_i ($i = 1, \ldots, k$) have a magnitude strictly larger than one. The corresponding eigenvectors can be chosen as

$$\hat{\mathbf{e}}_i = \left(\lambda_i^{k-1}(1 + \lambda_i),\, \ldots,\, \lambda_i(1 + \lambda_i),\, (1 + \lambda_i),\, 1\right)^{\mathrm{T}}, \tag{13}$$

and are seen to be orthogonal to $\mathbf{n}$, as

$$\hat{\mathbf{e}}_i \cdot \mathbf{n} = (1 + \lambda_i)\frac{1 - \lambda_i^k}{1 - \lambda_i} + 1 = \frac{2 - \lambda_i^k\, (1 + \lambda_i)}{1 - \lambda_i} = 0. \tag{14}$$

Any frequency preserving solution to the difference equation Eq. (6) may be written as

$$\mathbf{v}_n = \sum_{i=0}^{k} c_i\, (\lambda_i)^n\, \hat{\mathbf{e}}_i \tag{15}$$

with the coefficients c_i determined by the initial vector $\mathbf{v}_0$. Noting that

$$\mathbf{v}_0 \cdot \mathbf{n} = \sum_{i=0}^{k} c_i\, \hat{\mathbf{e}}_i \cdot \mathbf{n} = c_0 \tag{16}$$

and utilizing the definition of $\mathbf{v}_0$ to find

$$\mathbf{v}_0 \cdot \mathbf{n} = T_{0,1} + T_{0,2} + \cdots + T_{0,k} + \delta_0 = 1, \qquad (17)$$

it is seen that $c_0 = 1$. Since $\lambda_0 = 1$, one may decompose any frequency preserving solution as $\mathbf{v}_n = \hat{\mathbf{e}}_0 + \mathbf{u}_n$. In $k+1$ dimensional vector space, the solutions lie within a k dimensional plane with normal vector $\mathbf{n}$. The vector $\hat{\mathbf{e}}_0$ points from the origin to this plane and the vector $\mathbf{u}_n$ lies within this plane. For a solution vector $\mathbf{v}_n$ to correspond to a valid frequency preserving solution, it has to lie in the bounded region that satisfies $T_{n,j} > 0$ for ($j = 1, \ldots, k$), $\delta_n > 0$, and $T_{n,k} > \delta_n$. It is the latter boundary, coming from Eq. (4), that will be crossed after a sufficient number of iterations, due to the fact that the eigenvalues associated with the eigenvectors $\hat{\mathbf{e}}_i$ in Eq. (13) all have a magnitude larger than one. As a result, the solution will lose one zero crossing and cease to be frequency preserving. The sole exception are the triangular period $4/(2k+1)$ fast oscillating solutions ($k = 2, 4, \ldots$) that results from the initial condition $\mathbf{v}_0 = \hat{\mathbf{e}}_0$. These solutions are unstable because any perturbation results in dynamics that are not frequency preserving.

[1] A. N. Sharkovsky, Y. L. Maistrenko, and E. Y. Romanenko, *Difference Equations and their Applications* ((Russian) Nauka Dumka, Kiev 1986; (English) Kluwer, 1993).

[2] E. Fridman, L. Fridman, and E. Shustin, "Steady modes in relay control systems with time delay and periodic disturbances," J. Dyn. Sys., Meas., Control **122**, 732–737 (2000).

[3] J. K. Hale and S. M. V. Lunel, *Introduction to Functional Differential Equations*, vol. 99 of *Applied mathematical sciences* (Springer-Verlag, New York, 1993).

[4] J. Hale, L. T. Magalhães, and W. L. Oliva, *Dynamics in Infinite Dimensions*, vol. 47 of *Applied mathematical sciences* (Springer-Verlag, New York, 2002), 2nd ed.

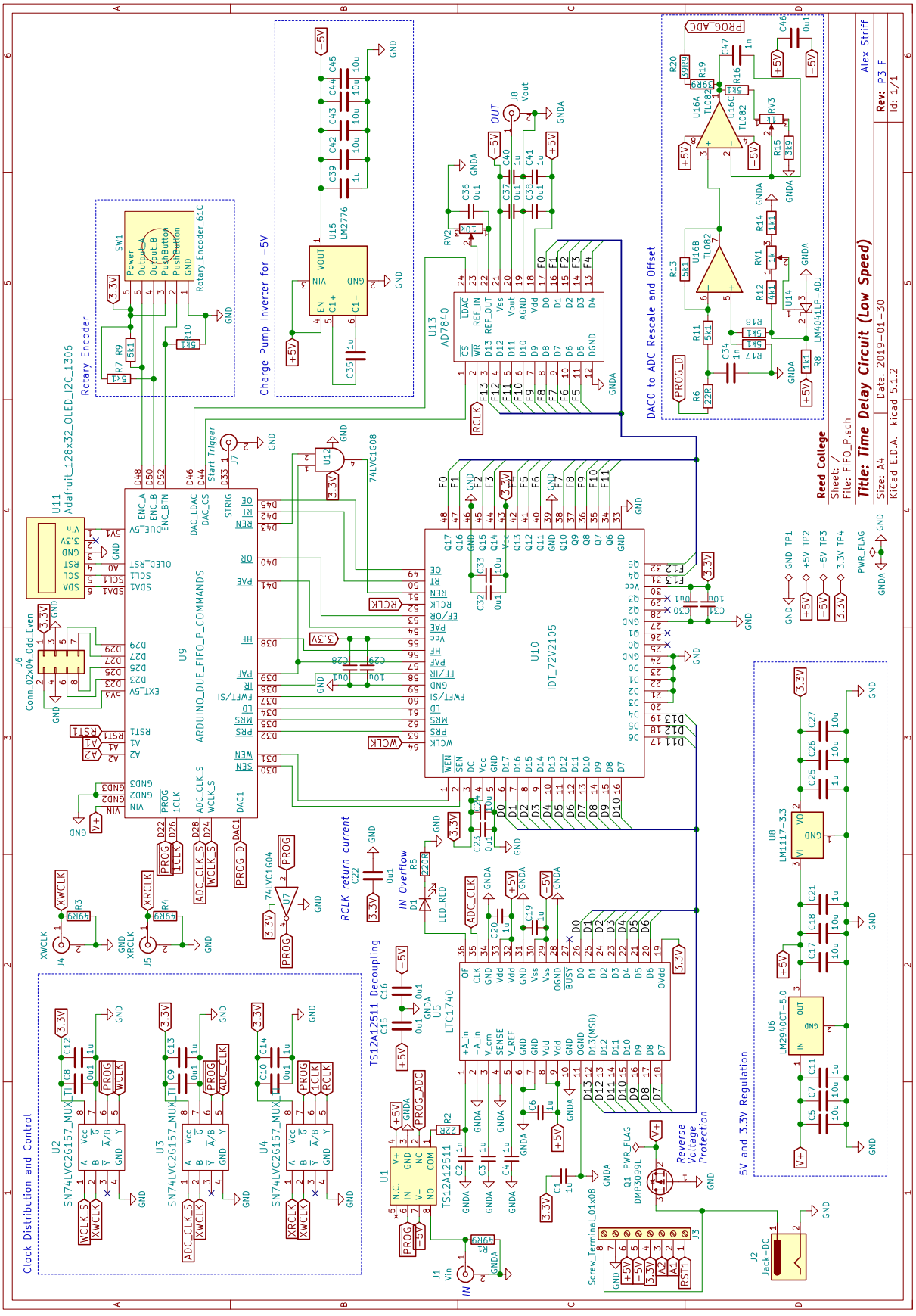

FIG. 2. The schematic of the time delay circuit.