PTP time sync notes

Emil Fresk
https://github.com/korken89

Here are the notes for the PTP time sync.

1. Clock modeling

A clock can be represented as an autonomous system with a tick count T and tick rate $\dot{T}$. The tick rate of the clock, $\dot{T}$, is bounded to a nominal value that is scaled with an unknown error.

$$\dot{T} = (1 + \varepsilon)\dot{T}_{\text{nom}}$$

While ε is not known, it has known bounds from the clock's datasheet and is generally specified in parts per million (ppm). The characteristics of ε is generally temperature and voltage dependent, however we can model ε as a bounded random walk process for simulation purposes.

$$\dot{\varepsilon} = -\alpha\varepsilon + \mathcal{N}(0, \sigma^2)$$

This will give a clock that has a nominal tick rate but that slowly drifts around the nominal value. The nominal tick rate for the PTP peripheral in STM32 microcontrollers is 2^{32} which is accumulated in a 64-bit register, this means that overflows can be neglected as this would overflow about once every 136 years.

Combining the above gives the system's transfer function is as follows:

$$\begin{pmatrix} \dot{T} \\ \dot{\varepsilon} \end{pmatrix} = \begin{pmatrix} 0 & \dot{T}_{\text{nom}} \\ 0 & -\alpha \end{pmatrix} \begin{pmatrix} T \\ \varepsilon \end{pmatrix} + \begin{pmatrix} \dot{T}_{\text{nom}} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \mathcal{N}(0, \sigma^2) \end{pmatrix}$$

For the clock on the micro processor we can control the deviation from nominal tick rate of the PTP clock, giving the following:

$$\begin{pmatrix} \dot{T} \\ \dot{\varepsilon} \end{pmatrix} = \begin{pmatrix} 0 & \dot{T}_{\text{nom}} \\ 0 & -\alpha \end{pmatrix} \begin{pmatrix} T \\ \varepsilon \end{pmatrix} + \begin{pmatrix} \dot{T}_{\text{nom}} + u \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \mathcal{N}(0, \sigma^2) \end{pmatrix}$$

It's important to understand that $\dot{T}_{\text{nom}}$ can differ between two clocks using the same crystal due to manufacturing tolerances. As an example, below is a simulation over the Brownian motion of two clocks that includes these manufacturing tolerances. However one can move this error into ε without issues.

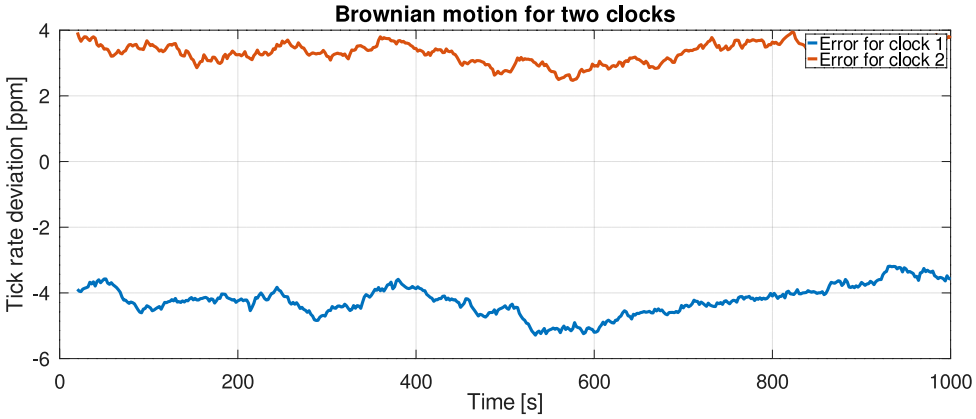

Figure 1: The simulated frequency error of two clocks.

2. Measurements

In PTP we have two clocks that we want to synchronize, one that we can control T^C, and some other autonomous clock T^A. While we can observe a timestamp from either clock we can't make an observation at the same time due to time delays to send timestamps and reading them out.

This is where the PTP peripheral in the MCU comes into play, through a set of network packet transfers we can measure the offset between the clocks as

$$y_m = T^C - T^A + w$$

where w is some measurement noise. This means that we need to find the dynamics model of the clock offset instead of individual clocks.

3. Offset modeling

From the offset

$$T^O = T^C - T^A$$

we can derive the corresponding dynamics by differentiating both sides

$$\dot{T}^O = \dot{T}^C - \dot{T}^A$$

and then simplifying based on the tick rate equation

$$\dot{T}^O = (1 + \varepsilon^C)\dot{T}_{\text{nom}} + u - (1 + \varepsilon^A)\dot{T}_{\text{nom}}$$
$$\dot{T}^O = \dot{T}_{\text{nom}}(\varepsilon^C - \varepsilon^A) + u.$$

This shows, as both ε^C and ε^A are random walk processes, that the offset will drift over time if no control action is supplied from u. Moreover, the bound on the drift is double that of each individual clock.

Finally, replacing the difference of Brownian motion into a new random walk process ε^O gives the following system that we want to control:

$$\begin{pmatrix} \dot{T}^O \\ \dot{\varepsilon}^O \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -\alpha \end{pmatrix} \begin{pmatrix} T^O \\ \varepsilon^O \end{pmatrix} + \begin{pmatrix} u \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \mathcal{N}(0, 2\sigma^2) \end{pmatrix}$$

However here an interesting question about our modeling comes into light. What are the actual dynamics of ε^O? Our modeling suggests that its dynamics are driven by something unknown, however in reality this will be driven mainly by temperature fluctuations and aging of the underlying clock.

Both which are not random processes. This means that there are most likely hidden dynamics we have not modeled here. Now one can either go down the rabbit hole and try to model these, or go with the assumption that estimating another derivative will capture enough of the dynamics. This would give the following system:

$$\begin{pmatrix} \dot{T}^O \\ \dot{\varepsilon}^O \\ \ddot{\varepsilon}^O \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} T^O \\ \varepsilon^O \\ \dot{\varepsilon}^O \end{pmatrix} + \begin{pmatrix} u \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \mathcal{N}(0, \sigma^2) \end{pmatrix}$$

We'll explore both and compare estimation errors.

4. Estimating offset

We can estimate the current offset and the ε states using a Kalman filter.

Lets start with the extended model model using the following discrete state space model:

$$\begin{pmatrix} T^O \\ \varepsilon^O \\ \dot{\varepsilon}^O \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} T^O \\ \varepsilon^O \\ \dot{\varepsilon}^O \end{pmatrix}_k + \begin{pmatrix} \Delta t \\ 0 \\ 0 \end{pmatrix} u_k + \mathcal{N}(0, \boldsymbol{Q})$$

$$y_k = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} T^O \\ \varepsilon^O \\ \dot{\varepsilon}^O \end{pmatrix}_k + \mathcal{N}(0, R)$$

Where the process covariance matrix is:

$$\boldsymbol{Q} = \begin{pmatrix} \frac{\Delta t^5}{20} & \frac{\Delta t^4}{8} & \frac{\Delta t^3}{6} \\ \frac{\Delta t^4}{8} & \frac{\Delta t^3}{6} & \frac{\Delta t^2}{2} \\ \frac{\Delta t^3}{6} & \frac{\Delta t^2}{2} & \Delta t \end{pmatrix} \sigma^2$$

and R is determined by measurement noise from a data set. Finally, the aggressiveness of the model can be tuned by varying σ, until desired responsiveness is found on a collected data set.

To reject outliers a Mahalanobis gating test will be added, it's simple but effective. However it's not suitable for initial convergence. Even more so, a Kalman Filter needs help with initial convergence to get good startup performance. Hence the initial T^O and ε^O will be estimated using least squares on a small initial sample set.

5. Control aim

The main reason for offset drift is that the effects from the random walk processes cannot be eliminated, hence a controller is needed. Given that we can estimate T^O and ε^O, we can formulate a state feedback regulator that drives the offset to 0.

That is, find a state feedback controller

$$u = -g \cdot \begin{pmatrix} T^O \\ \varepsilon^O \\ \dot{\varepsilon}^O \end{pmatrix}$$

such that

$$T^O \to 0.$$

Looking at the dynamics we can by inspection see that the full state space is uncontrollable, as the control signal has no interaction with any ε^O states. However as we only want to control T^O we can see that this state is indeed controllable. Which is good, else this would have all been a waste of time.

To make this work with available LQR methodologies (Octave/Matlab) we need to have a controllable system. To get this we need to eliminate the uncontrollable state.

5.1. Simulation results (acceleration model)

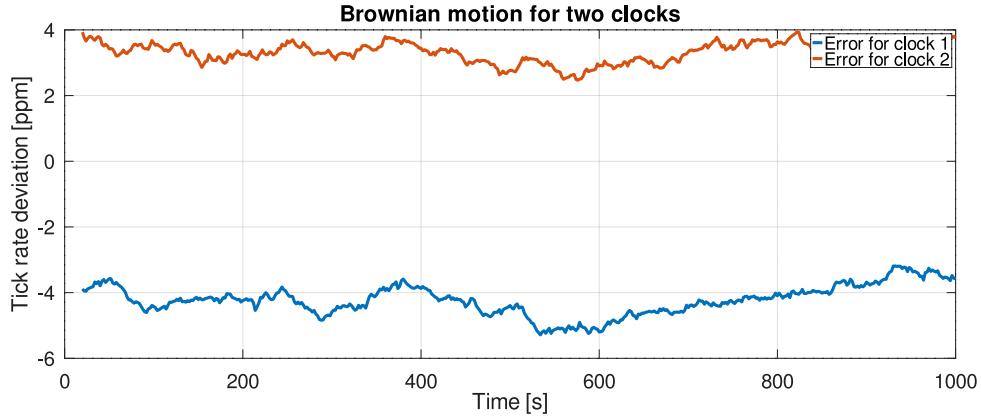

Figure 2: The simulated frequency error of the two clocks.

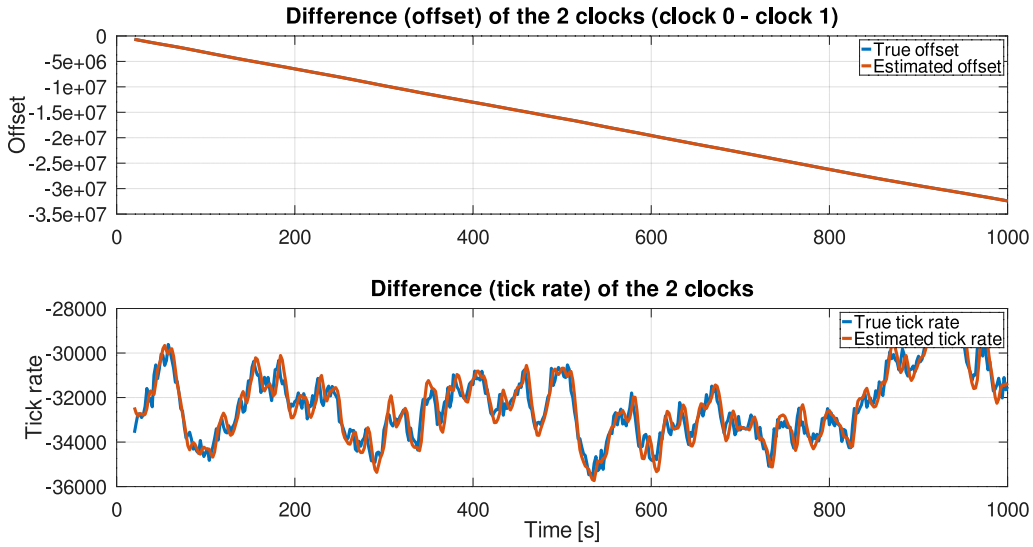

Figure 3: The tracking of the Kalman filter overlayed with true values.

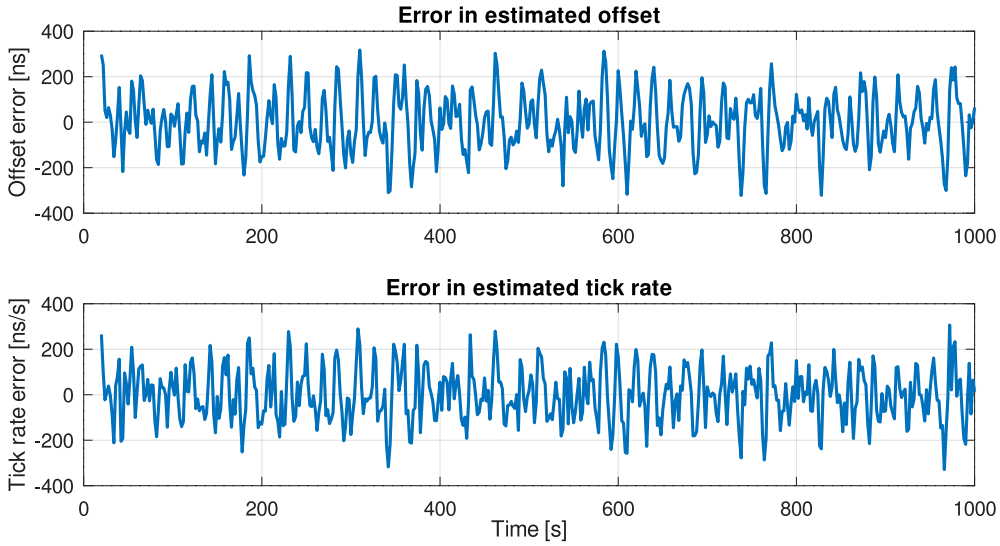

Figure 4: The tracking error of the Kalman filter.

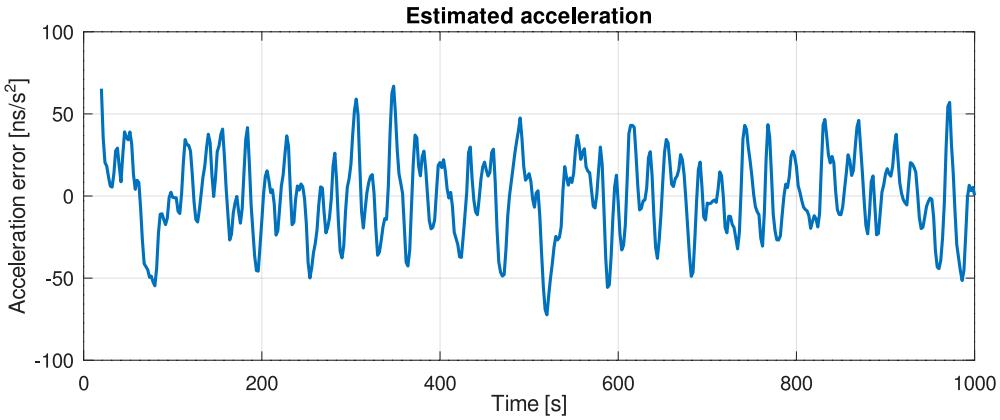

Figure 5: The estimated acceleration of the Kalman filter.