2022 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS) | 978-1-6654-7032-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISPCS55791.2022.9918446

Open-Source LiDAR Time Synchronization System by Mimicking GNSS-clock

Marsel Faizullin*, Anastasiia Kornilova† and Gonzalo Ferrer‡
Skolkovo Institute of Science and Technology, Moscow, Russia
*ORCID: *0000-0002-1053-7771, †0000-0002-2267-9689, ‡0000-0003-2704-7186*

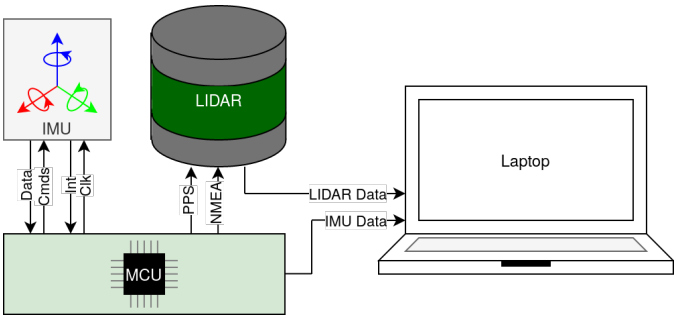
***Abstract*—Data fusion algorithms that employ LiDAR measurements, such as Visual-LiDAR, LiDAR-Inertial, or Multiple LiDAR Odometry and simultaneous localization and mapping (SLAM) rely on precise timestamping schemes that grant synchronicity to data from LiDAR and other sensors. Poor synchronization performance, due to incorrect timestamping procedure, may negatively affect the algorithms' state estimation results. To provide highly accurate and precise synchronization between the sensors, we introduce an open-source hardware-software LiDAR to other sensors time synchronization system that exploits a dedicated hardware LiDAR time synchronization interface by providing emulated GNSS-clock to this interface, no physical GNSS-receiver is needed. The emulator is based on a general-purpose microcontroller and, due to concise hardware and software architecture, can be easily modified or extended for synchronization of sets of different sensors such as cameras, inertial measurement units (IMUs), wheel encoders, other LiDARs, etc. In the paper, we provide an example of such a system with synchronized LiDAR and IMU sensors. We conducted an evaluation of the sensors synchronization accuracy and precision, and state 1 µs performance. We compared our results with timestamping provided by ROS software and by a LiDAR inner clocking scheme to underline clear advantages over these two baseline methods.**

***Index Terms*—clock synchronization, time synchronization, sensor networks, LiDAR, IMU, ROS, GNSS-clock, GNSS-time**

I. Introduction

Time synchronization (or shortly sync) is a crucial stage in the development of sensor networks aiming to solve data fusion, robot navigation or simultaneous localization and mapping (SLAM) problems. Synchronization matches measurements of different sensors with different clocks and thus allows a sensor network to create a common timing reference recorded by different clocks or systems.

Neglecting synchronization considerations generates conflicting data that are hard to fuse by the algorithms. A time offset of several ms between measurements of two LiDARs on high dynamic conditions introduces a significant position error of the same observed object. For instance, during a multi-LiDAR platform rotation on a spot (differential wheeled robot) with 5 radian per second angular velocity, time offset of 10 milliseconds leads to rotation error of 3 degrees that corresponds to 50 cm shift between observations of an object located at 10 meter distance. Other examples that underline essentiality of sensor synchronization are considered in [1]–[4].



(a) Block-scheme of the system

(b) Common view of the moving platform

Fig. 1: Block-scheme of the system (a) and common view of a 4WD moving platform carrying the system (b). *Data* and *Cmds* lines are used for IMU data gathering and IMU setup, *Int* is the interrupt line for precise timestamping, *Clk* line – reference clock. *PPS* – PPS signals, *NMEA* – NMEA GPRMC messages. Signal directions are depicted by arrows. IMU, MCU-board, and laptop, along with miscellaneous equipment of the platform, are placed into the platform body. Additional information on the system can be found at https://github.com/MobileRoboticsSkoltech/lidar-sync-mimics-GPS.

Time synchronization methods may be divided into hardware (HW) and software (SW) methods. Precise *hardware* synchronization based on GNSS-supplied clock is commonly used for sensors that have a specific interface for obtaining

time from GNSS-receivers [5]. However, a poor GNSS signal in GNSS-denied zones or urban canyon negatively affects not only localization but synchronization accuracy and precision as well [6].

A wide number of projects in the field of robotics are based on Robot Operation System (ROS) software [7]. This software by default utilizes one of the simplest methods of *software* time sync. Timestamping of sensor measurements in ROS is usually performed by matching the time of arrival of messages to a ROS-powered computer. However, the accuracy and precision of this technique are far from desired. The characteristic fluctuation of time of arrived messages usually reaches several milliseconds and more while delays between the true moment of measurements vary from one interface to another [8].

In this work, we introduce hardware LiDAR to inertial measurement unit (IMU) time synchronization system and provide estimation of its accuracy and precision. Any other sensor can be installed in place of IMU while keeping precision value at the same level. The system combines advantages of precise HW synchronization and ubiquitous open-source ROS software. At the same time, the system is entirely independent of the computer and can be used without ROS. Moreover, it does not need any specific synchronization equipment such as GNSS-receivers, and is based on general-purpose microcontroller (microcontroller unit or MCU).

The system is highly extendable and can manage synchronization of any other additional sensor by simple modification of firmware program blocks. It efficiently manages HW resources making it available for implementation even on a low-end MCU. We also provide synchronization precision analysis of our system, comparing the results with the pure ROS timestamping technique. Our system keeps the same magnitude of precision as the best possible on original separate timestamping schemes of the sensors. The current setup is used for correct data gathering for LiDAR-IMU Odometry algorithms [9]. The hardware design, software, and firmware are open-source, and available at our project page [10].

II. Synchronization Principle and System Architecture

In this section we briefly describe the LiDAR synchronization principle. We have chosen the VLP-16 LiDAR [5] due to its wide use in literature [11]–[14]. The principle below is generalizable, and correct for any other LiDAR that supports connection to an external GNSS-receiver. After the description, we introduce the HW architecture of our system.

VLP-16 LiDAR possess a dedicated HW interface for synchronizing data with global time provided by a GNSS-receiver [5]. The interface, through the physical connection, receives

- pulse-per-second (PPS) electrical signal and
- NMEA GPRMC (NG) messages.

NMEA GPRMC stands for National Marine Electronics Association Global Positioning Recommended Minimum Specific GNSS Data, more information can be found in the respective standard [15].

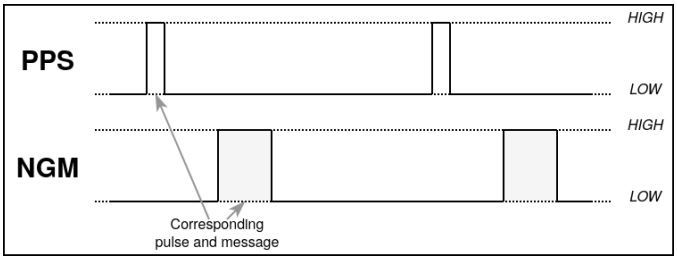


Fig. 2: Output of a GNSS-receiver clock. Pulse-per-second (PPS) signals are followed by NMEA GPRMC messages (NGM). The signal and messages are mimicked by our system.

GNSS-receiver PPS signals consist of short pulses arising every 1 second, they provide HW synchronization between the global GNSS and the internal LiDAR clock. Every PPS signal is followed by its corresponding NG message (Fig. 2). Every NG message is an ASCII string that contains a header, time, date, global coordinates, checksum and other information stored in fields separated by commas [16]. The LiDAR parses the string to obtain a timestamp. An example of an NG message is

```
$GPRMC,144326,A,5107.0017737,N,....
```

The second field (bold) corresponds to GNSS-clock timestamp of the PPS signal, and in this example counts 14 hours 43 minutes 26 seconds of Coordinated Universal Time (UTC). The timestamp may also include sub-seconds. The date is also parsed but is not shown in the example. Thus, the combination of the pulses and messages allows the LiDAR to adjust its internal clock to the global clock. Our system emulates GNSS-receiver by the MCU-platform. We applied this principle to synchronize the LiDAR sensor on the HW level and developed a system that emulates GNSS-receiver by the MCU-platform and synchronizes LiDAR and IMU sensors. In our case, no global time is given, only local time synchronization between the sensors is applied. Below is the description of the system architecture.

The system consists of three main parts:

1) STM32F4DISCOVERY MCU platform,
2) VLP-16 LiDAR,
3) MPU-9150 IMU.

All the data retrieved by the system are sent to a laptop or mini PC to be processed or stored: LiDAR directly to the laptop, IMU through MCU-platform.

The MCU platform performs (i) GNSS-clock emulation and feeding the clock to LiDAR, (ii) timestamped IMU data gathering and transmission. Thus, the LiDAR already contains the MCU-based clock within its data packets. IMU samples are timestamped by the MCU as well. Other sensors may need a different synchronization and/or data gathering schemes and interfaces; however, the common clock emulation principles

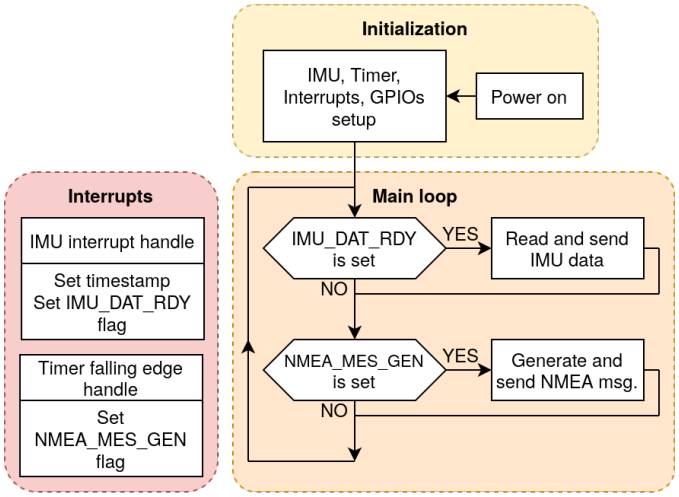


Fig. 3: The algorithm of the system. After initialization, the main loop is being interrupted by interrupts where specific flags are set. The flags are being checked in the main loop, and if some flag is set, specific operations are executed after the flag is reset.

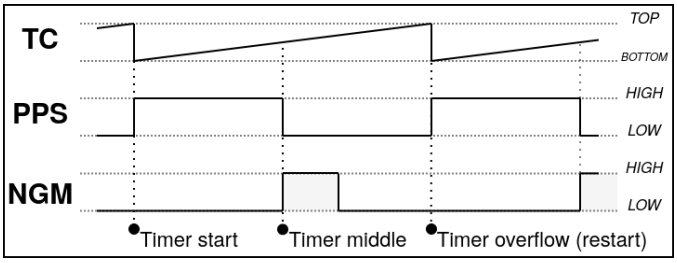


Fig. 4: Temporal diagram relating the timer counter values (TC) with the PPS signal and NG messages. The timer performs as (i) real-time clock (RTC), (ii) PPS signal generator, (iii) trigger for NG message transmission start. Timer start corresponds to a new second of RTC and the rising edge of the PPS signal, timer middle corresponds to the falling PPS signal edge and start of NG message transmission.

described in this work can be applied to them with no or minor modifications.

Therefore, the system timestamping authority is the MCU HW clock that guarantees sub-microsecond precise sync. The block-scheme of the system along with the common view of the platform powered by the system are depicted in Fig. 1.

An IMU data interface cannot provide precise timings and, consequently, information on the time of every measurement. To reach a precise timestamping, we employ the IMU interrupt functionality [17]. This tool provides an electrical pulse right after every new IMU data sample is ready. The application of this feature is explained in the next section. In addition, the IMU is provided with an external quartz crystal reference clock from MCU platform [18] to guarantee stability of the 1 kHz IMU sample rate.

III. Algorithm

The LiDAR-IMU synchronization and IMU data gathering algorithm consists of three main components depicted in Fig. 3. The components are (i) initialization, (ii) main loop, (iii) interrupts.

Initialization is aimed to set up all peripherals such as timer, the IMU reference clock, general-purpose input/output (GPIO), interrupts and other standard parameters and peripheral devices. The IMU is also set up at this stage. Although initialization is a common process, the role of the timer needs description in more details.

For emulation of a GNSS-receiver, the MCU-platform must (i) count time, (ii) generate PPS signals, and (iii) send NG messages to the LiDAR. Due to efficient firmware design, only a single general-purpose timer is utilized for performing these three processes. Otherwise, up to three timers would be needed.

For the first purpose (time count), the timer is used as a real-time clock (RTC) for IMU and LiDAR timestamping. The resolution of RTC is $1/76.8$ µs (about 13 ns), such a high resolution allows for the assignment of timestamps of the measurements with high precision. The timer overflow happens at every 1 second; this trick allows the timer peripheral to generate a positive electrical pulse on an MCU GPIO-pin at every restart of the timer. This electrical pulse is nothing but the rising edge of a PPS signal for the LiDAR.

The falling edge of the PPS signal, the negative pulse, is programmed to be at a half of a second after the positive one. This is the second usage of the timer. The negative pulse is also used to trigger an interrupt that sets up the NMEA_MES_GEN flag and makes the MCU generate and send an NG message containing a new timestamp to LiDAR; this is the third use of the timer. The exact time of sending the NG message is not critical for synchronization performance but must follow requirements described in [5]. NG messages do not include sub-second information since the PPS rising edges happens exactly at every new second with zero sub-second part. Another interrupt is triggered by the IMU indicating the readiness of a new inertial data sample. At this instant of time, the MCU saves the current timestamp and sets up the IMU_DAT_RDY flag. After gathering the data sample in the main loop, the timestamp along with the data sample is sent to a laptop. Interrupts functionality is crucial for providing high accuracy and precision of LiDAR-IMU time sync. They minimize possible MCU operational delays thanks to utilization of minimum CPU time of the MCU and mainly using the peripheral. Two types of interrupts, described below, do not interfere due to the higher priority of the IMU interrupt.

The main loop stage is sequentially checking two conditions: if a new IMU sample is ready (IMU_DAT_RDY flag is set) and if a new NG message needs to be generated and sent to LiDAR (NMEA_MES_GEN flag is set). If the first condition is fulfilled, then IMU data are read and sent to the laptop. The NG message containing the current timestamp is generated and sent to LiDAR if the second condition is met. The flags are set at the according interrupt handles (see Fig. 3) and reset in main loop. The MCU firmware is developed by STM32CubeMX IDE. The ROS driver for reading data is also developed and publicly available on the project page.

TABLE I: The timestamping techniques precision comparison

Timestamping technique	STD [µs]	Min. (max.) period [µs]	Time sync availability
ROS	82.64	5.72 (8931.40)	**Yes**
Internal LiDAR clock	**0.31**	**1327.0 (1328.0)**	No
MCU-based clock (ours)	1.35	1327.0 (1365.0)	**Yes**

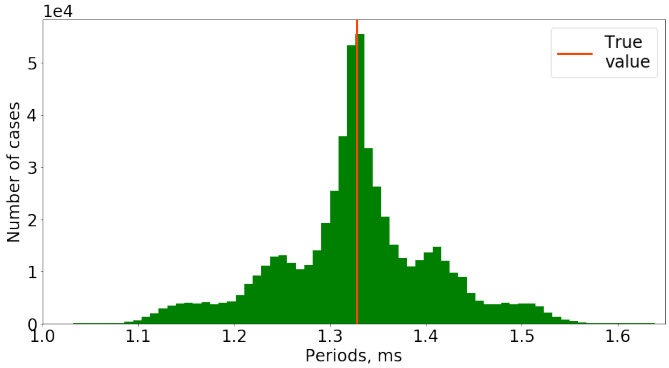


Fig. 5: Distribution of periods calculated from timestamps given by the pure ROS package [19]. The red vertical line expresses the true value of periods (1.328 ms according to [5]).

IV. Synchronization Performance Evaluation

In this section, we compare the precision of the three methods of time sync: (i) pure synchronization by ROS software, (ii) our improvement based on internal LiDAR timestamping scheme, (iii) external HW timestamping of LiDAR by MCU-platform. We also provide a theoretical estimation of synchronization accuracy of our system.

We found that the default LiDAR ROS package [5] suffers from fluctuations in the timestamping procedure. This happens because the timestamping scheme by default assigns the time of UDP-packets arrival to a computer. These arrivals are loosely related to true time instances of data sampling. We measured the stability of the packet-to-packet period (difference between neighboring timestamps) of LiDAR data for about ten minutes, gathering more than half a million UDP-packets and their timestamps generated by ROS. The histogram of the distribution of periods calculated by these timestamps is depicted in Fig. 5. We used standard deviation (STD) as a measure of obtained periods' precision. In addition, we measured the difference between the maximum and the minimum obtained period. These results are shown in Table I.

STD of several decades of microseconds may negatively influence on precision and performance of LiDAR-based navigation and mapping algorithms. For instance, any point in a distance of ten meters from LiDAR will inherit uncertainty of five centimeters in the tangential direction of LiDAR rays (for default LiDAR spinning velocity of ten rounds per second). Outliers (the third row of the table) also harm data processing adding errors.

To eliminate this drawback, we changed the timestamping scheme by our patch aimed to utilize the internal clock of the LiDAR as a time source for its measurements. Please refer to our project page for a description of the patch [10]. The LiDAR clock starts during the power-on sequence of the LiDAR and is related neither to another sensor time nor global time. According to the table, this method of timestamping has excellent precision comparing the technique described above. However, the key problem of the LiDAR internal clock consists of isolation of LiDAR timestamps from other sensors in a common sensor network (IMU in our case). This issue needs to be solved for correct data fusion of sensors including LiDAR.

Our system, on the one hand, resolves inter-sensor synchronization; on the other hand, it keeps precision of timestamping at the microsecond order (see Table I). Precision drops a little because of MCU clock to LiDAR time drift phenomena in-between neighboring PPS signals [18]. This drift has a local effect and is not being integrated over time because each subsequent PPS reloads internal LiDAR time values. This fact was checked empirically during the development of the system. LiDAR can be synchronized once by sending a single PPS signal followed by a single NGM but, in this case, time drift will affect synchronization performance by adding an offset that grows over time [20], and we do not follow this strategy. IMU data timestamping precision is kept at several nanoseconds, which is on the floor of RTC resolution and is not considered in whole precision estimation due to insignificant value of fluctuations.

In this paragraph we provide an estimation of system accuracy according to the sensors' documentation. Empirical estimation involves data driven evaluation methods and is beyond the scope of this work. The LiDAR documentation [5] states no delay between GNSS-based clock and computed data timestamps and provides 1 µs resolution of data timestamps. The IMU delay between a data sample and its corresponding IMU_DAT_RDY interrupt is explicitly provided for different setups of the IMU internal low-pass filter with 10 µs precision [17]. All the MCU operating delays related to timestamping are kept under 1 µs order by design and confirmed in our previous work [20]. Considering this findings, we state that time synchronization accuracy is kept as best as 10 µs. In case of utilization of two or more of the same type LiDARs, the accuracy can reach 1 µs order.

V. Conclusion

In this work, we have introduced a LiDAR time synchronization system to a general spectrum of sensors, including IMU, cameras, other LiDARs, etc. Our MCU-based synchronization allows sensor networks to be tightly synchronized at the HW level. Additional data timestamping, as by ROS, is not needed due to automatic online LiDAR timestamps assignment. The system can work as a tight complement to ROS software as well as be completely independent on it for specific projects.

We have evaluated our system in comparison with SW synchronization via ROS timestamping, showing the superiority in precision of our approach while we maintain all the advantages and flexibility that ROS offers. We also provided an analysis of precision and showed that our system keeps the

same magnitude of precision (1 µs of STD) as the best possible original clocking schemes of the sensors used. We showed that the synchronization accuracy order is 10 µs for our setup and can be improved in case of the same type sensors used.

REFERENCES

[1] E. Olson, "A passive solution to the sensor synchronization problem," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 1059–1064.

[2] A. English, P. Ross, D. Ball, B. Upcroft, and P. Corke, "Triggersync: A time synchronisation tool," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 6220–6226.

[3] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 431–437.

[4] F. Tschopp, M. Riner, M. Fehr, L. Bernreiter, F. Furrer, T. Novkovic, A. Pfrunder, C. Cadena, R. Siegwart, and J. Nieto, "Versavis—an open versatile multi-camera visual-inertial sensor suite," *Sensors*, vol. 20, no. 5, p. 1439, 2020.

[5] Puck lidar sensor (VLP-16). Velodyne Lidar. Accessed Jun. 13, 2021. [Online]. Available: https://velodynelidar.com/products/puck/#downloads

[6] P. H. Dana, "Global positioning system (gps) time dissemination for real-time applications," *Real-Time Systems*, vol. 12, no. 1, pp. 9–40, 1997.

[7] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[8] J. Park, R. Delgado, and B. W. Choi, "Real-time characteristics of ros 2.0 in multiagent robot systems: An empirical study," *IEEE Access*, vol. 8, pp. 154 637–154 651, 2020.

[9] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3144–3150.

[10] Open-source LiDAR Time Synchronization System by Mimicking GNSS-clock. Mobile Robotics Lab, Skoltech. Accessed Jun. 13, 2021. [Online]. Available: https://github.com/MobileRoboticsSkoltech/lidar-sync-mimics-gps

[11] D. J. Yeong, G. Velasco-Hernandez, J. Barry, J. Walsh, *et al.*, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, p. 2140, 2021.

[12] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3144–3150.

[13] V. Ponnaganti, M. Moh, and T. Moh, "Deep learning for lidar-based autonomous vehicles in smart cities," *Handbook of smart cities*, pp. 1–25, 2020.

[14] O. Sahin, R. V. Nezafat, and M. Cetin, "Methods for classification of truck trailers using side-fire light detection and ranging (lidar) data," *Journal of Intelligent Transportation Systems*, vol. 26, no. 1, pp. 1–13, 2021.

[15] NMEA 0183 Standard. National Marine Electronics Association. Accessed Jul. 22, 2022. [Online]. Available: https://www.nmea.org/content/STANDARDS/NMEA_0183_Standard

[16] GPRMC Sentence. NovAtel. Accessed Jul. 25, 2022. [Online]. Available: https://docs.novatel.com/OEM7/Content/Logs/GPRMC.htm

[17] Invensense, *IMU MPU-9150*. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-9150-Datasheet.pdf

[18] T. Schmid, Z. Charbiwala, J. Friedman, Y. H. Cho, and M. B. Srivastava, "Exploiting manufacturing variations for compensating environment-induced clock drift in time synchronization," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 1, pp. 97–108, 2008.

[19] Velodyne lidar ROS driver. ROS.org. Accessed Jun. 13, 2021. [Online]. Available: http://wiki.ros.org/velodyne

[20] M. Faizullin, A. Kornilova, A. Akhmetyanov, and G. Ferrer, "Twist-n-sync: Software clock synchronization with microseconds accuracy using mems-gyroscopes," *Sensors*, vol. 21, no. 1, p. 68, 2021.