Open-Source LiDAR Time Synchronization System by Mimicking GPS-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*—Time synchronization of multiple sensors is one of the main issues when building sensor networks. Data fusion algorithms and their applications, such as LiDAR-IMU Odometry (LIO), rely on precise timestamping. We introduce open-source LiDAR to inertial measurement unit (IMU) hardware time synchronization system, which could be generalized to multiple sensors such as cameras, encoders, other LiDARs, etc. The system mimics a GPS-supplied clock interface by a microcontroller-powered platform and provides 1 µs synchronization precision. In addition, we conduct an evaluation of the system precision comparing to other synchronization methods, including timestamping provided by ROS software and LiDAR inner clock, showing clear advantages over both baseline methods.**

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

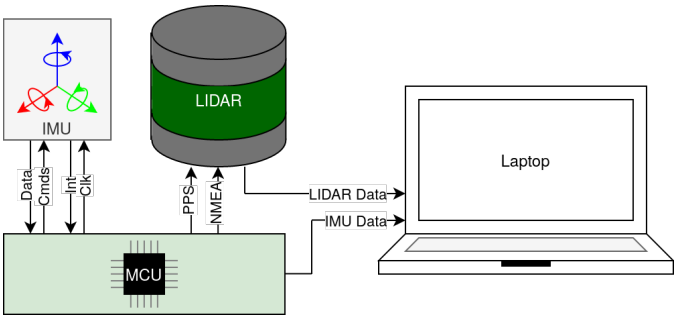
I. Introduction

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.

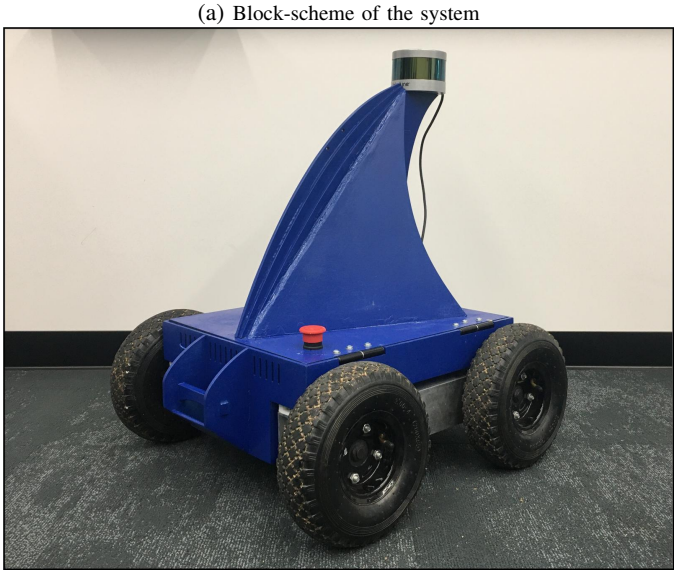
Neglecting sync considerations generates conflicting data that are hard to fuse by the algorithms. For instance, a time offset of several ms between camera and LiDAR measurements on high dynamic conditions introduces a significant position error of the same observed object. Importance of sensors sync is emphasized in a number of state-of-the-art navigation systems [1]–[3].

Time sync methods may be divided into hardware (HW) and software (SW) methods. Precise *hardware* sync based on GPS-supplied clock is commonly used for sensors that have a specific interface for obtaining time from GPS-receivers [4]. However, pure GPS signal in GPS-denied zones or urban canyon negatively affects not only localization but sync accuracy and precision as well [5].

A wide number of projects in the field of robotics are based on Robotic Operation System (ROS) software [6]. This software utilizes by default one of the simplest methods of *software* time sync. Timestamping of sensor measurements in ROS is usually made 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 [7].



(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 interrupt line for precise timestamping, *Clk* line – reference clock. *PPS* – PPS signals, *NMEA* – NMEA GPRMC messages. Signal directions depicted by arrows. IMU, MCU-board and laptop along with miscellaneous equipment of the platform placed into the platform body. Additional information on the system can be found at https://github.com/MobileRoboticsSkoltech/lidar-sync-mimics-gps.

In this work, we introduce hardware LiDAR to inertial measurement unit (IMU) time synchronization system and provide estimation of its 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 sync 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 sync tools such as GPS-receivers for sync.

The system is highly extendable and can manage sync of any other additional sensor due to simple modification of firmware program blocks. It efficiently manages HW resources making it available to implement even on low-end microcontrollers (microcontroller unit or MCU). We also provide sync 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 [8], [9]. The hardware design, software and firmware will be available at our project page [10] after the paper acceptance.

II. System Architecture

The system consists of three main parts: VLP-16 LiDAR, MPU-9150 IMU and STM32F4DISCOVERY MCU platform. All the retrieved data by the system are sent to a laptop to be processed or stored. The MCU platform is used for LiDAR-IMU data timestamps sync, IMU setup and IMU data gathering. LiDAR sends measurements directly to a laptop while the IMU needs setup and data conversion via MCU. The MCU is also responsible for IMU data timestamping. Then, the prepared IMU data with timestamps are sent to the laptop by the MCU data interface. Therefore, the system timestamping authority is the MCU clock that guarantees more 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.

LiDAR has a specific interface for HW time sync with GPS-receivers. A GPS-receiver must be connected to the interface to provide global time to LiDAR. The interface provides pulse-per-second (PPS) pulses and NMEA GPRMC messages (NGM) to LiDAR [4]. Every message contains global coordinates and the time of GPS-receiver in the instance of time of the last PPS pulse. The logic of the interface implies that any message is related to a single PPS pulse. Our system emulates GPS-receiver by the MCU-platform. Xu *et al.* follow similar trick in [2], here we also provide quantitative results of this emulation method. An IMU data interface cannot provide precise timings and, consequently, information on the time of every measurement. To reach a precise timestamping, we use the IMU interrupt functionality [11]. 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 by an external quartz crystal reference clock from MCU platform [12]. This allows IMU to get a stable sampling period of data.

III. Algotithm

LiDAR-IMU sync and IMU data gathering algorithm consists of three main parts depicted in Fig. 2. The parts are (i) initialization, (ii) main loop, (iii) interrupts.

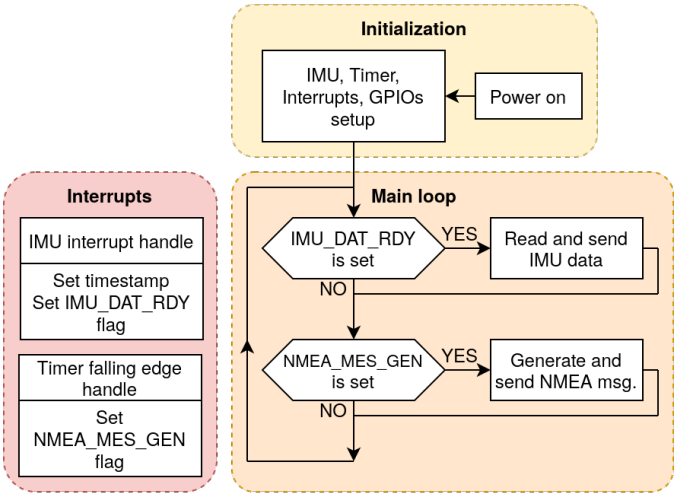


Fig. 2: The algorithm of the system. After initialization, main loop is being interrupted by interrupts where specific flags are set. The flags are being checked out at main loop, and if some flag is set, specific operations are executed after flag resetting.

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

For emulation of GPS-receiver, MCU-platform must (i) count time, (ii) generate PPS signal and (iii) send NGM to LiDAR. Due to efficient firmware design, only a single general-purpose timer is used for performing these three processes. The timer is used as a real-time clock (RTC) for IMU and LiDAR timestamping. The resolution of RTC is chosen to be a multiple of one second ($1/76.8$ µs or about 13 ns), and timer overflow is also chosen to be at every second. This decision is made to generate a positive electrical pulse at every restart of the timer – this is the rising edge of a PPS signal. Otherwise, an additional timer would be needed. The falling edge of the PPS signal, the negative pulse, is programmed to be at half of a second after the positive one. The negative pulse is used to trigger an interrupt that makes MCU to generate and send NGM containing new timestamp to LiDAR; this is the third use of the timer. The exact time of sending the message is not critical but must follow requirements described in [4].

Another interrupt is triggered by IMU indicating the readiness of a new inertial data sample. At this instance of time, MCU saves the current timestamp and sets up the IMU_DAT_RDY flag. After gathering the data sample in main loop, the timestamp along with the data sample is sent to a laptop. Interrupts play an important role in precision of LiDAR-IMU time sync. Possible MCU operational delays are minimized due to utilization of minimum CPU time of MCU and mainly using peripheral. Two types of interrupts do not interfere due to the higher priority of IMU interrupt.

Main loop stage is sequentially checking two conditions:

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**

if a new IMU sample is ready (IMU_DAT_RDY flag is set) and if a new NGM 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. NGM 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 the figure) and reset in main loop. MCU firmware is developed by STM32CubeMX IDE. ROS driver for reading data is also developed and publicly available on the project page.

IV. Synchronization Performance

In this section, we compare precision of three methods of time sync: (i) pure sync by ROS software, (ii) our improvement based on internal LiDAR timestamping scheme, (iii) external HW timestamping of LiDAR by MCU-platform.

We found that the default LiDAR ROS package [4] 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. 3. 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 timestamping scheme by our patch aimed to utilize the internal clock of 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 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 of 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.

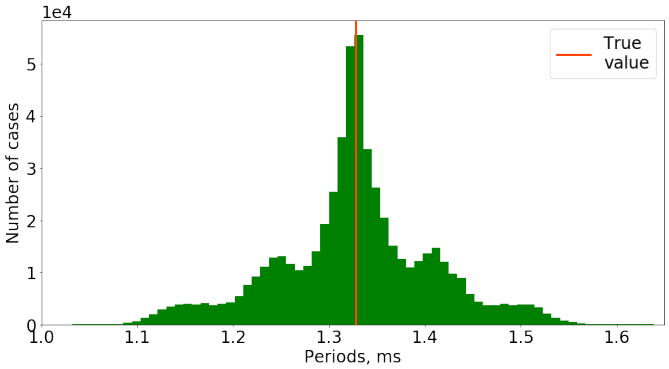


Fig. 3: Distribution of periods calculated from timestamps given by pure ROS package [13]. Red vertical line expresses true value of periods (1.328 ms according to [4]).

Our system, on the one hand, resolves inter-sensor synchronization; on the other hand, it keeps precision of timestamping at microsecond order (see Table I). Precision drops a little because of MCU clock to LiDAR time drift phenomena in-between neighboring PPS signals [12]. This drift has a local effect and is not being integrated over time because every next PPS reloads internal LiDAR time values. This fact was checked empirically during the development of the system. LiDAR can be synchronized once by sending single PPS signal followed by a single NGM, but in this case, time drift will affect sync performance by adding offset that grows over time [14], and we do not follow this strategy. IMU data timestamping precision is kept at several nanoseconds that is on the floor of RTC resolution and is not considered in whole precision estimation due to insignificant value of fluctuations.

Estimation of system accuracy (say sensor-to-sensor offset) has specific value for different sets of sensors and thus is not shown in the work because of lack of generality while precision is able to be generalized or re-estimated.

V. Conclusion

In this work, we have introduced a LiDAR time sync system to a general spectrum of sensors, including IMU, cameras, other LiDARs, etc. Our MCU-based sync 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 sync 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 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.

References

[1] 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.

[2] J. Xu, J. Lv, Z. Pan, Y. Liu, and Y. Chen, "Real-time lidar data assocation aided by imu in high dynamic environment," in *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2018, pp. 202–205.

[3] 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.

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

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

[6] 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*. Kobe, Japan, 2009.

[7] 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.

[8] 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.

[9] ——, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3144–3150.

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

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

[12] 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.

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

[14] 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.