2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

EverySync: An Open Hardware Time Synchronization Sensor Suite for Common Sensors in SLAM

Xuankang Wu, Haoxiang Sun, Rongguang Wu, Zheng Fang*

***Abstract*— Multi-sensor fusion systems have been widely applied in various fields, including mobile robot, simultaneous localization and mapping (SLAM), and autonomous driving. For a tightly coupled multi-sensor fusion system, strict time synchronization between sensors will improve the accuracy of the system. However, there is currently a lack of open-source and general-purpose hardware synchronization systems for Cameras, IMUs, LiDARs, GNSS/RTK in the academic community. Therefore, we propose EverySync, an open hardware time synchronization system to address this gap. The synchronization accuracy of the system was evaluated through multiple experiments, achieving an accuracy of less than 1 ms. And, real-world experiments proved that hardware time synchronization improves the accuracy of the SLAM system. This open-source system is available on GitHub.**

I. INTRODUCTION

In recent years, multi-sensor fusion systems have been widely applied in various fields such as mobile robot, simultaneous localization and mapping (SLAM), and autonomous driving. In SLAM systems, the utilization of visual-inertial measurement units (IMU) in combination with LiDAR or GNSS/RTK modules [1]–[4] has shown the ability to enhance system robustness and accuracy, enabling adaptability to unstructured and dynamic environments. In many frameworks, time offsets or delays between sensors can lead to poor results, or even result in failure of the entire method. Some frameworks, such as VINS-Mono [1], can estimate time offsets online during the estimation process. However, online estimation methods are susceptible to various errors, such as inaccurate camera parameters, LiDAR distortions, IMU noise, and incorrect matching [5]. Therefore, hardware time synchronization is very important for multi-sensor fusion systems. Assigning precise timestamps to sensor data through hardware circuit boards can further improve the accuracy and robustness of pose estimation [6].

SLAM systems often incorporate multiple Cameras, LiDARs, IMUs, as well as GNSS/RTK and other common sensors. The multi-sensor information obtained based on a hardware time synchronization system exhibits a high level of temporal consistency and accuracy, which plays a crucial role in the accuracy and robustness of SLAM systems. For commercially and academically established sensors, such as the Skybotix VI-Sensor [7] used in the EuROC Micro Aerial Vehicle (MAV) dataset [8] and the Intel RealSense [9] employed in the PennCOSYVIO dataset [10], as well as the SkyAware sensor utilized in the works of Honegger et al. [11] or PIRVS [12], their hardware configurations are constrained by the image sensor, lens, camera baseline, and IMU. It is generally challenging to further expand these frameworks to achieve fusion with other sensors.

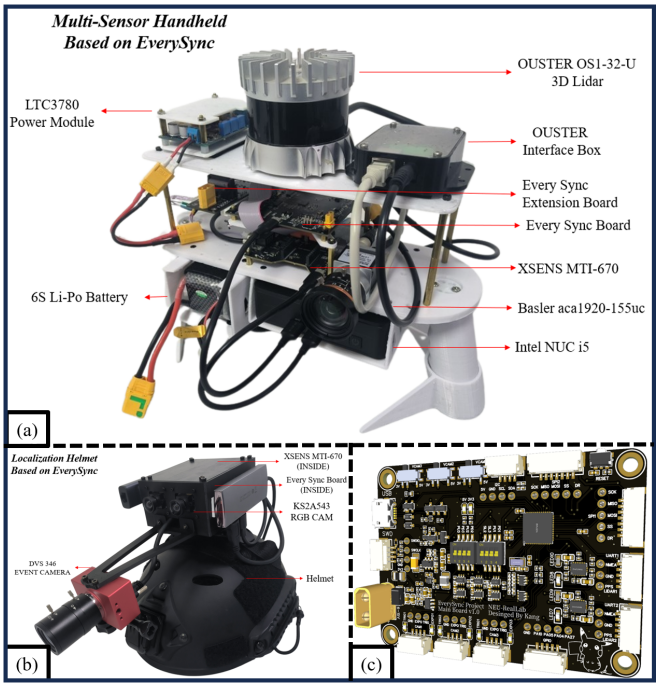

Fig. 1. EverySync System: multi-Cameras, multi-LiDARs, IMU, and GNSS/RTK hardware time synchronization system suitable for SLAM field.(a)Multi-Sensor Handhold. (b) Localization Helmet (c) EverySync MCU Board.

In this paper, we propose EverySync[1], a complete hardware time synchronization system which comprising hardware circuits and software algorithms, based on the open-source solution of VersaVIS [6]. Compared to VersaVIS, our system supports Cameras, IMUs, LiDARs and GNSS modules, making it suitable for most SLAM applications. The hardware time synchronization system and its applications are shown in Fig. 1. Our contributions can be summarized as follows:

This work was supported in part by the National Natural Science Foundation of China under Grants 62073066, in part by the Fundamental Research Funds for the Central Universities under Grant N2226001, and in part by 111 Project under Grant B16009. (Corresponding author: Zheng Fang.)

The authors are all with the Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110819, China. Xuankang Wu and Zheng Fang are also with the National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Northeastern University, Shenyang 110819, China and and also with the Key Laboratory of Data Analytics and Optimization for Smart Industry, Ministry of Education, Northeastern University, Shenyang 110819, China. Email:{2202067, 2202058, 2202066}@stu.neu.edu.cn, fangzheng@mail.neu.edu.cn.

[1]https://github.com/NEU-REAL/EverySync-Hardware-Suite

- We optimized the synchronization scheme for Cameras and IMUs in the system, enabling support for Trigger-Only Cameras through software compensation of exposure time. The system also supports the integration of high-precision IMUs that can be externally triggered for synchronization.
- We proposed a time synchronization algorithm that supports synchronous operation with multiple LiDARs. The algorithm has been tested for compatibility on LiDAR brands such as OUSTER, RoboSense, and Livox, demonstrating its universality. Additionally, we incorporated the PPS signal provided by GNSS/RTK receivers into the system to synchronize with the GNSS/RTK receivers.
- Extensive experimental validation was conducted on real-world datasets, and compared with other approaches, our synchronization system exhibited greater generality and versatility. By conducting comparative experiments using open-source SLAM algorithms, we demonstrated that the hardware time synchronization SLAM system built with this proposed scheme achieves superior robustness and accuracy.
- Our method is divided into hardware circuit design, hardware structure design, synchronization program. The complete system solution is open source and available on GitHub.

II. RELATED WORK

In this section, we introduce recent works on multi-sensor time synchronization and existing sensor synchronization framework.

Multi-sensor time synchronization frameworks include passive synchronization algorithms [13] and solutions like TriggerSync [14], which do not require additional hardware synchronization frameworks. Lu et al. [15] proposed a network protocol-based synchronization system, which is a strictly time-synchronized trigger-based multi-sensor system.

We categorize the general sensor hardware time synchronization framework into three types: Camera-IMU, LiDAR, and GNSS/RTK.

A. Camera-IMU Time Synchronization

To achieve camera-IMU hardware time synchronization, Nikolic et al. [7] proposed the VI-Sensor, a vision-inertial time synchronization system implemented through FPGA preprocessing and data forwarding. The advantage of this system is its ability to perform precise time synchronization between the camera and IMU without utilizing CPU resources. However, the drawback is that the solution is relatively inflexible, making it difficult to reproduce, and it is currently unavailable due to its high cost.

To address the aforementioned issues, Tschopp et al. [6] proposed VersaVIS, an open-source sensor suite that supports hardware time synchronization for multiple camera-IMU setups. This suite utilizes a low-cost microcontroller unit (MCU) and achieves camera-IMU hardware time synchronization through timestamp exchange and hardware triggering between the host and the MCU. The solution demonstrates good scalability. However, it lacks a universal solution for external clock synchronization and LiDAR integration.

B. LiDAR Time Synchronization

For common 3D LiDARs, they typically support various modes of time synchronization, such as PTP (Precision Time Protocol), GPS, and serial port synchronization. In general, in vehicle sensor systems, they are often paired with RTK or GNSS sensors. The PPS pulse and NMEA (National Marine Electronics Association) time signal provided by such sensors can serve as the timing signals for the LiDAR. The LiDAR clears the current frame timestamp upon receiving the PPS (Pluse Per Second) signal and assigns the Unix timestamp converted from the NMEA timestamp to the current frame's LiDAR data. Meanwhile, it continues counting using the internal clock until the next PPS signal and NMEA timestamp are received. This type of timing method is widely used in open-source solutions [16], [17].

However, the GNSS-based timing method has certain drawbacks. For instance, in indoor or tunnel environments where GNSS signals are blocked or in densely populated urban areas with interference, the unstable GNSS signals cannot provide a stable timing reference for the LiDAR. To address this issue, an auxiliary processor can be added to maintain a stable clock source for timing. Faizullin et al. [18] simulated GNSS signals using a microcontroller to ensure a stable timing signal.

C. GNSS module Time Synchronization

GNSS/RTK sensors are used to receive GNSS positioning and timing information from satellites. GNSS satellites utilize atomic clock timing and achieve time synchronization through local recovery, often serving as the time reference for the overall system. In many solutions, system synchronization is achieved by directly or indirectly accessing the GNSS clock's PPS signal and NMEA signal [19], [20]. However, such approaches perform poorly in GNSS-denied environments and require signal strength discrimination and additional processors for clock maintenance and real-time alignment to maintain accurate time synchronization [21].

III. SYSTEM DESIGN AND IMPLEMENTATION

A. System Overview

EverySync system consists of core circuit, sensors, firmware on MCU and hardware driver on host computer.

The core circuit of EverySync system is based on the open-source solution VersaVIS [6], using the ATSAMD21G18A-MU chip as the main controller for the circuit board. Its core is compatible with the Arduino-Zero [22] series. Communication is achieved using the ROS-serial [23] program natively as a ROS node. The main frequency of the circuit board is up to 48 MHz, and the core is based on ARM Cortex M0+, providing excellent performance and a wide range of expansion interfaces such as SPI, I2C, UART, etc.

Fig. 2 illustrates system overview as an example. The hardware selection list for the system is as follows: The

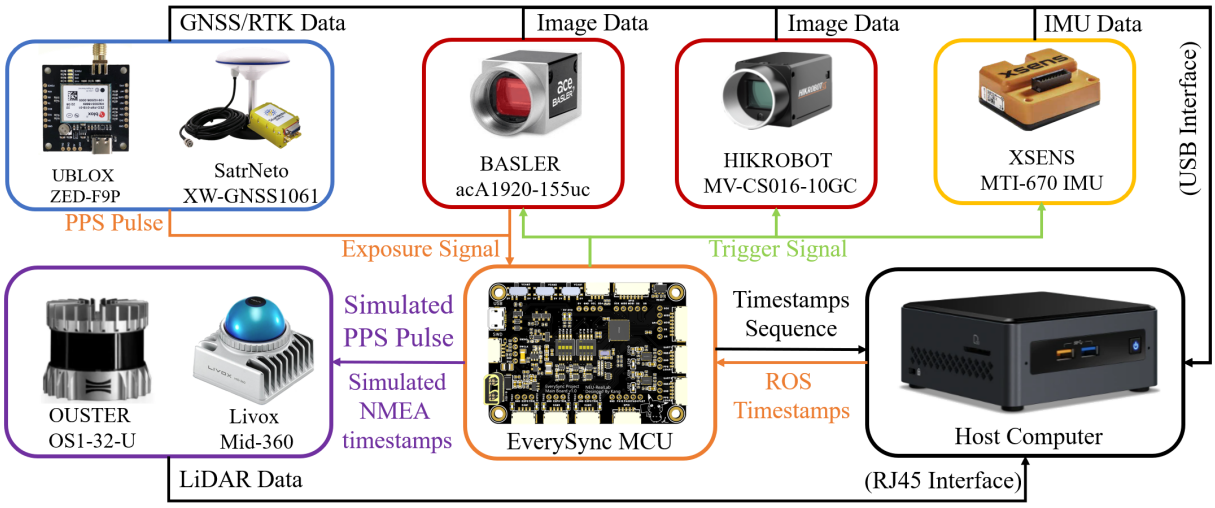

Fig. 2. The EverySync System Overview: The EverySync MCU sends trigger signals to Triggeralbe-IMU and Cameras. Sensor data is directly transmitted to the host and combined with the timestamp from the MCU within the host. The IMU data is received directly by the host and utilizes the IMU-triggered sampled timestamp. LiDARs achieves GNSS synchronization by utilizing the simulated Pulse Per Second (PPS) signal and NMEA timestamp provided by the board. GNSS or RTK receivers, serving as external clock, receive the PPS signal and NMEA timestamp from the atomic clock of satellites and passe them to the host for computing external time deviation.

cameras employed are Basler acA1920-155uc and Hikvision MV-CS016-10GC. The IMU chosen is the XSENS MTI-600 series. For LiDAR synchronization, the system supports a full range of LiDARs including Robosense, Ouster, and Livox, which are capable of GNSS-based timing. The GNSS/RTK receviers selected is UBLOX ZED-F9P. The system operates under the ROS system, supporting multiple sensor outputs with pulse triggers, recording and calculating timestamps.

The host computer runs the synchronization board driver and synchronization program to accomplish the pairing and reassembly of data sequences from multiple sensors.

The MCU firmware ensures that it can trigger at the correct timing or capture external triggers, and capture the timestamps of sensor measurements triggered.

B. Main Workflow

The EverySync MCU is used to trigger sensor measurement at regular intervals, set timestamps, and send the timestamp sequence to the host. Additionally, the MCU sends trigger pulses to the camera to initiate image exposure which applies to both standalone cameras and stereo cameras. After successful exposure, the MCU reads the exposure time by monitoring the cameras' exposure signal, enabling it to perform exposure compensation and set mid-exposure timestamps. The timestamps, along with strictly increasing sequence numbers, are sent to the host. As a result, image data is directly transmitted from the camera to the host, avoiding the need to transfer large amounts of data through the MCU. This allows the usage of high-resolution cameras even with low-performance MCU.

As common sensors are usually individual, sensor data from each sensor is unsynchronized, which is shown in Fig.**??**. The genneral workflow of EverySync system is devided into initialization procedure and regular procedure, which are detailed described in Sec.IV. The initialization procedure is used to find every sensor's sequence bias and start at the same time in EverySync system. After that, entering the regular procedure, the host computer merges the sensors' timestamps received from the MCU with the corresponding sensors' messages based on the sequence numbers.

IV. MULTI-SENSORS SYNCHRONIZER DESIGN

This section provides a detailed classification of time synchronization methods for commonly used sensors in SLAM systems. Different synchronization schemes are employed for different sensors based on their specific requirements. In the end, all sensor timestamps in the system are unified under a consistent host ROS system clock [24], enabling the direct utilization of sensor data within the ROS system.

We divide the design of the multi-sensor hardware time synchronizer into initialization procedure and regular procedure.

- Initialization Procedure:

After the triggering sensors and trigger board are activated, the first step is to match the sensor serial numbers with the trigger board serial numbers. This is done by slowly triggering (1 Hz) the sensors to obtain the closest timestamp and find the corresponding sequence. If the trigger period is significantly longer than the expected delay or if there is inconsistency in the corresponding serial numbers due to data fluctuations on the host, the initialization process is restarted until all sensors have been initialized. The detailed illustration is shown in Fig.3. The *sensor* here is classified into Cameras and IMUs supported by the system, which is marked as *cam* or *imu*. According to Equation 1, n_{sensor} and n_{sync} are confirmed until multiple consecutive adjacent time intervals meet the maximum time threshold T_{max} in

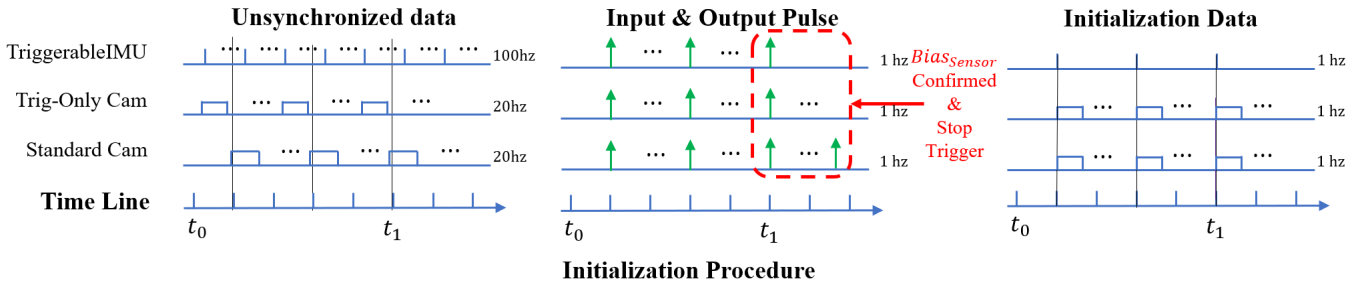

Fig. 3. The schematic diagram of trigger signals in the EverySync System while initialization procedure. The green and purple upper arrows represent the output pulses of the MCU, the orange lower arrows represent the input pulses, and the blue lower arrows represent timing data reading.

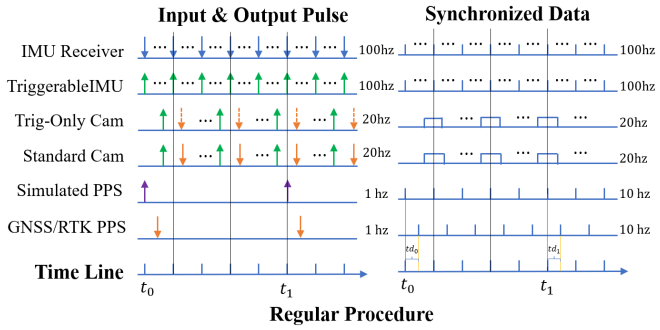

Fig. 4. The schematic diagram of trigger signals in the EverySync System.The green and purple upper arrows represent the output pulses of the MCU, the orange lower arrows represent the input pulses, and the blue lower arrows represent timing data reading.

Equation 2.

$$n_{sensor}, n_{sync} = \underset{n_{sensor}\in\mathcal{N}_{sensor}, n_{sync}\in\mathcal{N}_{sync}}{\arg\min} \left| t_{sensor}^{n_{sync}} - t_{sync}^{n_{sync}} \right| \tag{1}$$

$$\left| t_{sensor}^{n_{sync}} - t_{sync}^{n_{sync}} \right| \leq T_{max} \tag{2}$$

where $\mathcal{N}_{sensor}$ represents the collection of each sequences of sensor and $\mathcal{N}_{sync}$ represents trigger timestamp sequence from board starting from the triggering moment. $t_{sensor}^{n_{sync}}$ and $t_{sync}^{n_{sync}}$ represent the timestamp of sensor data sequence and trigger sequence corresponding to n_{sync}, respectively.

$$bias_{sensor} = n_{sensor} - n_{sync} \tag{3}$$

Once the offset $bias_{sensor}$ between the triggering sensor data sequence and the trigger board timestamp sequence is determined, which is shown in Equation 3. System enters the regular procedure, where synchronized triggering is performed using the specified frames.

- Regular Procedure:

In the regular procedure, the design of trigger pulses and synchronization schemes is tailored to different categories of sensors, which can be classified into four main categories: Camera, IMU, LiDAR, and GNSS/RTK. Within each category, specific synchronizer classifications and introductions are conducted. The schematic diagram and corresponding sensor data are shown in Fig. 4.

A. Camera

In this system, we classify cameras capable of triggering image capture and providing exposure pulse signal outputs as *Standard Cameras*. Conversely, cameras capable of triggering image capture but lacking the ability to output dedicated exposure pulse signals are categorized as *Trig-only Cameras*.

1) Standard Camera: Standard camera is defined as a sensor capable of being triggered by signal pulses and with a non-zero data measurement time, which corresponds to the image exposure time. Additionally, the sensor is required to provide an exposure signal indicating the sensor's exposure state, commonly known as a strobe signal. While both trigger pulses and timestamps are created on the microcontroller unit (MCU) based on its internal clock, the image data is directly transmitted to the host via USB or Ethernet. To establish the correct association between image data and timestamps on the host, EverySync assigns independent sequence numbers, represented as n_{sensor} and n_{sync}, to the image data and timestamps, respectively. The mapping between these sequence numbers is determined during the initialization period.

The mid-exposure compensation is mentioned in Furgale et al. [25] and is beneficial for image-based state estimation. Instead of using periodic triggering of the camera, the approach proposed by Nikolic et al. [7] is adopted, which is also used in VersaVIS [6]. The idea is to trigger the camera's periodic mid-exposure timestamp by advancing the exposure time by half.

$$t_{img} = (t_{trig} + t_{expo})/2 \tag{4}$$

where $\mathcal{N}_{sensor}$ represents the sensor trigger timestamp is t_{trig}, the image synchronized timestamp is t_{img}, the exposure timestamp is t_{expo}.

2) Trig-only Camera: Unlike standard cameras, Trig-only Cameras only support triggering but do not have output for exposure signals. Therefore, it is not possible to determine the median exposure timestamp by measuring the exposure duration. In the timed exposure mode, we publish the timed exposure duration T_{expo} along with the camera image sequence. During the message reassembly process on the host, the timed exposure duration is compensated to the trigger time.

In automatic exposure, our approach involves estimating the exposure time compensation by assuming the average value of multiple samples of image transmission time. Assuming the sensor trigger timestamp is t_{trig}, the image message timestamp is t_{recv}, the estimated transmission time is t_{trans}, then the estimated image exposure time t_{expo} can

be obtained as follows:

$$T_{expo} = t_{recv} - t_{trig} - t_{trans} \tag{5}$$

This method can effectively accomplish trigger control and exposure compensation for this kind of camera.

B. IMU

For SLAM systems, IMU is an essential component. Due to its high-frequency measurement capabilities and high-precision inertial measurements, the IMU provides important information for the high-frency output of the SLAM system.

1) IMU-Receiver: Based on the original system, this system introduces support for the BMI088 and ICM42688 series IMU. IMU data is read through an MCU using SPI or I2C communication. IMU messages are received and forwarded with minimal information on the trigger board, and the message reconstruction and consolidation are performed on the host. The IMU data is then published in the form of ROS topic.

2) Triggerable IMU: For IMUs that support data retrieval through USB interface, which offer pulse triggering to acquire sample data for multi-sensor time synchronization, such as the Xsens MTi-600 series. In this system, IMUs of this kind achieve multi-sensor time synchronization by connecting to synchronized trigger signals and transmitting data through the USB interface. Treating these IMUs as cameras that do not require exposure time, the trigger time for each frame satisfies the following equation:

$$t_{imu} = t_{trig} \tag{6}$$

C. LiDAR

Most commonly used LiDARs support GNSS time synchronization. In this system, we achieve LiDAR time synchronization by simulating GNSS time synchronization using an MCU. This approach enables simple and efficient synchronization and frame alignment of multiple LiDARs, as well as time synchronization between the LiDAR and other sensors in the system.

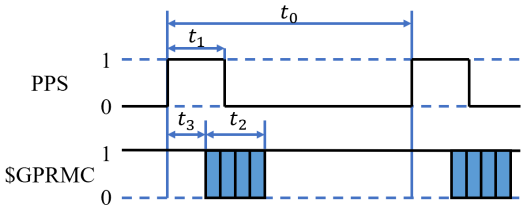

Fig. 5. PPS pulse timing diagram. In this diagram, t_0 represents the interval between adjacent rising edges of the PPS signal, which has an effective duration of 900ms-1100ms. t_1 represents the high-level duration of the PPS signal, which is effective for 5ms-900ms. t_2 represents the transmission time of the GPRMC data (baud rate: 9600bps), which has an effective duration of 70ms. t_3 represents the delay between the start of GPRMC data transmission and the rising edge of the PPS signal, with an effective duration of 0ms-900ms.

Referring to the data requirements for GNSS time synchronization with Livox LiDAR [26], as depicted in Fig. 5. The specific implementation is achieved as follows:

(1) The microcontroller configures a 1Hz timer trigger using an independent clock and generates a general PPS pulse. In this system, it is designed to generate a square wave pulse signal with a high level of 50ms duration at a frequency of 1Hz.

(2) At the beginning of the timer trigger, specifically at the rising edge of the PPS signal, the MCU retrieves the ROS time from the host, which represents the system's Unix time at that moment. The MCU calculates and converts this time into the NMEA signal format of the GPRMC protocol. Once the calculation is completed, the MCU sends the simulated GPRMC NMEA signal to the LiDAR for timing synchronization. In the simulated GPRMC NMEA signal, only the time-related information and the checksum need to be calculated, while the remaining bits are set as character constants to reduce computational load.

D. GNSS/RTK

Considering that SLAM systems are often used in GNSS-denied or intermittent conditions, it leads to intermittent or discontinuous PPS and NMEA signals. If the GNSS clock is used as the system clock, during the intervals, the time needs to switch to the host time of the system for clock maintenance. When it resumes, there is a possibility of time jumps due to time deviations. Therefore, in this system, the GNSS module is synchronized with the hardware time synchronization system using a real-time updated time deviation parameter, denoted as td_n. The synchronization is achieved through the following steps:

(1) Once the GNSS receiver confirms the reception of a valid GNSS signal and positioning information, it will transmit the positioning information along with the PPS time t_{Pre} of the next frame.

(2) The MCU is connected to the PPS pin of the GNSS receiver, and captures the PPS pulse using an external interrupt. When the rising edge of the pulse is detected, an interrupt is triggered, and the current system time is recorded as t_{Trig}.

(3) The time deviation parameter td_n is calculated as follows:

$$td_n = t_{Pre} - t_{Trig} \tag{7}$$

where t_{Pre} is the broadcasted time from the GNSS receiver and t_{Trig} is the time of the external interrupt trigger.

V. Experiments And Evaluation

In this section, we demonstrate the reliability and system time accuracy of our system through experiments involving multiple sensor precision experiments and SLAM application experiments.

A. Precision Experiment

1) Camera-Camera: Multi-camera time synchronization primarily involves assessing whether the captured images from multiple cameras are consistent in terms of exposure timing and evaluating the pulse trigger errors at the same moment. To evaluate this, we conducted long-duration recordings of the high-refresh-rate screen (165Hz) using multiple

cameras in the system, shown in Fig. 6. We utilized an external trigger pin of another MCU to record the timestamp of the rising edge of the trigger pulse. In a dataset over 10 minutes, 500 image pairs were checked randomly with no error, the time synchronization errors among the cameras are presented in Table I. The multi-cameras synchronization accuracy is less than 0.2ms.

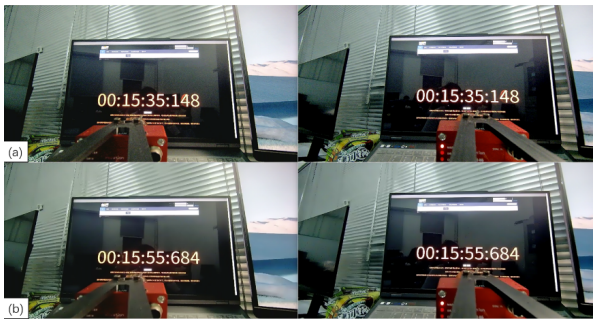

Fig. 6. Precision Experiment for multi-cams. (a) Image pair captured by different cameras at the same time. (b) A later image pair.

2) LiDAR-LiDAR: To evaluate the synchronization relationship between multiple LiDARs, we deployed two identical OUSTER-OS1-32 LiDARs for testing. By collecting 10 sets of data using the synchronization system, each set is recorded for 5 minutes, timestamp data from the Http API of each LiDAR were simultaneously collected to evaluate the time deviations between the LiDARs. The evaluation of the timestamp errors among the LiDARs is presented in Table I, its synchronization accuracy is less than 0.1ms.

TABLE I

DETAIL TIME OFFSET OF EVERYSYNC SYSTEM.

Sensor Type	Time Offset[ms]	
	Mean	Std
Cam-Cam	0.136982	0.000697
Lidar-Lidar	0.077916	0.000339

3) Camera-IMU: The synchronization relationship between the multi-camera and IMU was evaluated by performing N sets of experimental calibration using the Kalibr framework [25]. The results were compared with two commercial-grade VIO sensor platforms, namely Realsense D435i [27] and Viobot [28], as shown in Table III.

The reprojection error represents the fitting degree between the lens and distortion model and the actual lens, as well as the consistency between the camera's motion and the IMU's motion after calibration. Both EverySync and VersaVIS exhibited consistently low reprojection errors, indicating that the camera and IMU converged to consistent extrinsics during the calibration experiments, and the measurements from both sensors were highly consistent.

B. Application Experiment

In Fig. 7, we deploy the developed system on a mobile robot platform and evaluate the performance of the multi-sensor system in SLAM algorithm experiments with and without time synchronization. The dataset used in our experiments was collected in campus environment. The mobile

TABLE II

KALIBR BENCHMARK OF DIFFERENT CAMERA-IMU

Sensor	Reprojection Error [pixel]		Gyroscope Error [rad/s]	
	Mean	Std	Mean	Std
EverySync	0.101342	0.086469	0.004938	0.000628
Versavis	0.100078	0.064657	0.009890	0.006353
Realsense	0.1701585	0.400515	0.008595	0.012385
Viobot	0.126749	0.019900	0.010715	0.004876
Sensor	Accelerometer error [m/s2]		Time offset [ms]	
	Mean	Std	Mean	Std
EverySync	0.018008	0.002581	2.591619	0.034513
Versavis	0.143162	0.191270	1.552360	0.034126
Realsense	0.057778	0.045611	6.44265	0.047525
Viobot	0.053249	0.013572	19.563170	0.042293

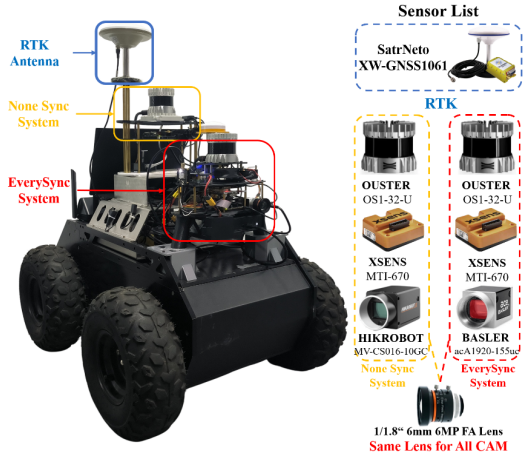

Fig. 7. The real-world experiment platform: A mobile robot base carries both Sync and None Sync multi-sensor system.

robot platform is equipped with Basler acA1920-155uc and Hikrobot MV-CS016-10GC cameras, using the same camera lens, with an image frequency of 20 Hz. The Xsens MTi-670 MEMS IMU has a sampling frequency of 100 Hz, and the OUSTER-OS1-32 LiDAR provides 10 Hz point cloud data. The ground truth of the dataset is collected using the SatrNeto XW-GNSS1061, with an RTK positioning accuracy of 2 cm.

TABLE III

REAL-WORLD EXPERIMENT BENCHMARK

	RMSE	Mean	Max	Min
EverySync System	1.70135	1.60448	3.38127	0.290129
None Sync System	3.18945	2.9209	5.4323	0.775926

Fig. 8 illustrates the Tightly-coupled Sparse-Direct LiDAR-Inertial-Visual Odometry algorithm FAST-LIVO [4] used in this experiment. Table III presents the experimental results of the FAST-LIVO algorithm with the same parameters on the dataset in terms of the APE metric for synchronized and non-synchronized systems.

The RTK reference trajectory length in the experiment is 1347.79m. EverySync demonstrates a trajectory that is closer to the ground truth and exhibits smaller drift. In contrast, the non-synchronized system using only ROS timestamp shows larger drift on the map, especially at multiple turning points. We attribute this to poor pose estimation caused by the lack of data synchronization in the multi-sensor fusion

Fig. 8. The RTK ground truth and LIVO trajectories from different sensor suite are plotted on Google Maps.

process. Moreover, compared to the None Sync system, the EverySync system exhibits smaller end-to-end error. This experiment demonstrates that a rigorous hardware time synchronization system offers improved accuracy and robustness for tightly-coupled multi-sensor fusion SLAM systems.

VI. CONCLUSION

In this paper, we propose a comprehensive and versatile hardware time synchronization system for multi-sensor fusion. The proposed system provides a stable and accurate hardware synchronization clock source for multi-sensor fusion SLAM systems that require precise time measurement and timestamps. Through extensive experimental validation, the system demonstrates the capability to synchronize commonly used sensors in the SLAM domain, including Cameras, IMUs, LiDARs, and GNSS/RTK.

In the future, we will continue to optimize the hardware time synchronization system, enhance its compatibility.

REFERENCES

[1] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[2] S. Cao, X. Lu, and S. Shen, "Gvins: Tightly coupled gnss–visual–inertial fusion for smooth and consistent state estimation," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2004–2021, 2022.

[3] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[4] C. Zheng, Q. Zhu, W. Xu, X. Liu, Q. Guo, and F. Zhang, "Fast-livo: Fast and tightly-coupled sparse-direct lidar-inertial-visual odometry," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4003–4009.

[5] X. Liu, S. Wen, Z. Jiang, W. Tian, T. Z. Qiu, and K. M. Othman, "A multi-sensor fusion with automatic vision-lidar calibration based on factor graph joint optimization for slam," *IEEE Transactions on Instrumentation and Measurement*, 2023.

[6] 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.

[7] 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.

[8] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[9] Intel Corporation., "Intel ® realsense ™, Tracking Camera T265," 2023, https://www.intelrealsense.com/tracking-camera-t265/, Last accessed on 2023-9-10.

[10] B. Pfrommer, N. Sanket, K. Daniilidis, and J. Cleveland, "Penncosyvio: A challenging visual inertial odometry benchmark," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3847–3854.

[11] D. Honegger, T. Sattler, and M. Pollefeys, "Embedded real-time multi-baseline stereo," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5245–5250.

[12] Z. Zhang, S. Liu, G. Tsai, H. Hu, C.-C. Chu, and F. Zheng, "Pirvs: An advanced visual-inertial slam system with flexible sensor fusion and hardware co-design," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3826–3832.

[13] 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.

[14] 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.

[15] L. Lu, C. Zhang, Y. Liu, W. Zhang, and Y. Xia, "Ieee 1588-based general and precise time synchronization method for multiple sensors," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 2427–2432.

[16] Zhuang Yan, "hard-sync-dut, Available online," 2019, https://github.com/ZhuangYanDLUT/hard_sync_dut, Last accessed on 2023-9-10.

[17] Hui Liu, "sync-gps-lidar-imu-cam, Available online," 2021, https://github.com/nkliuhui/sync_gps_lidar_imu_cam, Last accessed on 2023-9-10.

[18] M. Faizullin, A. Kornilova, and G. Ferrer, "Open-source lidar time synchronization system by mimicking gnss-clock," in *2022 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. IEEE, 2022, pp. 1–5.

[19] H. Guo and P. Crossley, "Design of a time synchronization system based on gps and ieee 1588 for transmission substations," *IEEE Transactions on power delivery*, vol. 32, no. 4, pp. 2091–2100, 2016.

[20] A. Carta, N. Locci, C. Muscas, F. Pinna, and S. Sulis, "Gps and ieee 1588 synchronization for the measurement of synchrophasors in electric power systems," *Computer Standards & Interfaces*, vol. 33, no. 2, pp. 176–181, 2011.

[21] W. Yao, Y. Liu, D. Zhou, Z. Pan, M. J. Till, J. Zhao, L. Zhu, L. Zhan, Q. Tang, and Y. Liu, "Impact of gps signal loss and its mitigation in power system synchronized measurement devices," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 1141–1149, 2016.

[22] Arduino Inc., "Arduino zero, Available online," 2023, https://store.arduino.cc/arduino-zero, Last accessed on 2023-9-9.

[23] Ferguson, M.; Bouchier, P.; Purvis, M., "rosserial, Available online," 2023, http://wiki.ros.org/rosserial, Last accessed on 2023-9-9.

[24] 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.

[25] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1280–1286.

[26] Livoxtech Inc., "livox-wiki-cn, Available online," 2023, https://livox-wiki-cn.readthedocs.io/zh_CN/latest/tutorials/other_product/timestamp_sychronization.html, Last accessed on 2023-9-10.

[27] Intel Corporation., "Intel ® realsense ™, Depth Camera D435i," 2023, https://www.intelrealsense.com/depth-camera-d435i, Last accessed on 2023-9-9.

[28] Zichuan-tech Inc., "Viobot, Multi-Sensor Pose Fusion Component," 2023, https://www.zichuan-tech.com/list/77.html, Last accessed on 2023-9-9.