Development of Omni-SLR System: (3) Timing/software subsystem

Yusuke Yokota(1), Toshimichi Otsubo (2), Hiroshi Araki (3), Takehiro Matsumoto (4), Kenji Kouno (1)
(1) Institute of Industrial Science, University of Tokyo, Tokyo, Japan; (2) Hitotsubashi University, Tokyo, Japan; (3) National Astronomical Observatory, Tokyo, Japan; (4) Japan Aerospace Exploration Agency, Tsukuba, Japan

Omni-SLR Project is a Hit-U, NAOJ, U-Tokyo, JAXA joint project aiming to build a small SLR system. A small PC such as Raspberry-Pi is arranged for each part of the timer part that acquires time data, the laser part/detector part that is the transmission controller of the laser, the mount, and the software parts.

Omni-SLR uses Swabian Instrument's Time Tagger (TT) for the timer part. For the detector part, we conducted in situ ranging experiments for several candidates. Using this setting, the outdoor background noise was acquired with a noise rate of around MHz through an ND5 filter. We also confirmed the operation in the cold-room (-30 degrees Celsius) environment in the NIPR.

With high-frequency lasers above kHz, the amount of received data, that is, the amount of communication from the timer part to the PC, increases together with the background noise. We are experimenting with arbitrary waveform generators and receivers to see how much data can be processed in real-time.

Received data is introduced to the PC side via a USB3.0 cable in binary format. In the case of MHz rates, saving each time data acquisition and ascii conversion cause processing delays. Therefore, we are considering a design in which a virtual reception channel is set for the real reception channel of TT, and a channel that can automatically record the round-trip time is installed in TT. This virtual channel can support various uses such as conventional transmission mode with about Hz, burst mode, and operation to targets other than satellites by shifting only the timing. We are also considering GUI for real-time measurement. The system is designed to be controllable via web browsers, on-site or remotely, as the user interface part is constructed using Streamlit and Flask.

Acknowledgement: This research was supported by JSPS KAKENHI Grant Number JP20H01993