OpenAWAM: Open-Source Bluetooth Travel Time System

John Kerenyi (F), T.E., E.E., PTOE, Senior Engineer, City of Moreno Valley

Introduction

Intelligent Transportation Systems (ITS) have many interoperating parts. Arterial management systems require close supervision of traffic signal controllers to gauge performance and respond to incidents. Freeway management systems require incident detection and active ramp meter control. One requirement common to all such systems is the need to know how the system is currently performing: Is traffic flowing well, or stopped? How long will it take to get from here to there? How does today's traffic patterns compare to yesterday or last week?

Recent developments in ITS and computer technology have converged to select one communication architecture as preferred above all others: the Ethernet-based field data communication system. Whether built on fiber optics, copper, or wireless, these networks allow flexible, robust, high-bandwidth communication to a multitude of devices. Such systems are so flexible that once deployed, operators and managers inevitably ask themselves: What else can we do with this system?

Enter the Bluetooth-based travel-time analysis system, which provides detailed traffic flow information using standards-based equipment. Such systems continually scan their surroundings for Bluetooth-enabled devices which are broadcasting their presence (i.e., are "discoverable.") Thousands of such devices traverse heavily-traveled streets daily. Each time they are sensed, the time and place are logged, along with an identifier. When a particular device is logged in multiple locations, travel-time information is easily derived. A network of such devices produces a robust, ever-changing picture of current traffic flow. Historical data is easily made available for subsequent analysis. Such systems have acquired the acronym AWAM, which stands for Anonymous Wireless Address Matching.

AWAM systems have been commercially available for years, at relatively high cost. But a parallel development has been explosive growth in the field of embedded computing: Devices which offer virtually all of the functionality of a desktop computer system, in a tiny package, at very low cost. OpenAWAM is based on the Raspberry Pi, which costs $35 each. Once Ethernet is available to field devices, the cost of each node is roughly $200 including computer, Bluetooth adapter, power assembly, enclosure, and cabling.

The OpenAWAM project seeks to supply the missing component of such a system: The software. OpenAWAM is built on, and consists entirely of, open-source software and thus is free in two senses: It is available at no cost, and is fully customizable by the end user. The OpenAWAM implementer thus acquires a robust, extensible, travel time monitoring system which, due to its open-source nature, can be integrated straightforwardly with other systems, including:

- Freeway and Arterial Transportation Management Systems (FMS/ATMS)
- Dissemination of travel-time information via regional systems such as 511

OpenAWAM Architecture/Concept of Operation

The OpenAWAM system consists of hardware and software for field and central. The following matrix breaks down the system's features in these two dimensions. As in the rest of this document, it is assumed that Ethernet communication is available between the Transportation Management Center (TMC or "Central") and each field location.

Table 1. OpenAWAM System Elements

	Central	**Field**
Hardware	A computer running any modern version of Windows or Linux. Can be x86, x64, or ARM based. Computer need not be dedicated to the OpenAWAM application.	OpenAWAM field node *(refer to **Figure 1** for more information)*
Software	OpenAWAM Central	Any Linux distribution optimized for use with the Raspberry Pi *(OpenAWAM uses a Debian-based distribution)*

The author has tested various installation methods and locations for the field equipment, and has arrived at the following installation recommendations:

- The field node equipment should be mounted at about nine-foot height on an existing traffic signal pole, facing into the intersection (**Figures 2 and 3**), in a weatherproof enclosure. Lower installations are subject to tampering. Higher installations may result in fewer addresses read due to the planar nature of the lobe antenna provided with the recommended Bluetooth adapter: Vehicles passing right under the field node may be missed.
- The field node is connected to the existing Ethernet switch using standard Ethernet twisted-pair cabling (Category 5e or better). The same cable supplies both power and communication, using a low-cost, off-the-shelf Power over Ethernet injector/splitter combination. The unit can be located up to 100 meters (328 feet) from the Ethernet switch, which allows for great flexibility in locating it.

The field installation recommendations are provided as **Figures 2 and 3** (plan view and elevation, respectively). An overall system block diagram is provided as **Figure 4.**

Figure 4 shows the functions of each component of the OpenAWAM system. The field node is powered by, and communicates with, a nearby (within 100 meters—an Ethernet limitation) cabinet housing an Ethernet switch; this cabinet can either be dedicated to ITS components, or can be a regular traffic signal controller cabinet. The OpenAWAM field node needs only enough space in the cabinet to hold a small Power over Ethernet injector.

The existing Ethernet field data network provides a network connection between the field node and the OpenAWAM central computer. Between them is usually a firewall, and minor configuration of that firewall is required to allow the traffic data to pass; in addition to any other functions that may be of interest. For example, it would be prudent to allow SSH traffic (TCP port 22) in order to allow remote logins to the field nodes for the purpose of maintenance and troubleshooting.

The OpenAWAM central computer collects and stores all data from the field nodes. Each time a log entry is made, the central computer performs an update of real-time traffic conditions, and pushes the data out to its subscribers (e.g., Web pages actively viewing the map). Thus, any maps currently being viewed always show the most current traffic conditions.

Field Node Software Stack

Each field node is provided with a slightly customized version of **Debian GNU/Linux,** a popular, robust, and full-featured operating system which is readily available for embedded computers such as the Raspberry Pi. (Another popular distribution which can be adapted for use with OpenAWAM is ArchLinux; the author selected Debian due to his familiarity with that distribution.) Once imaged onto a standard SD card and configured for your network[1], the device will locate your Bluetooth adapter and automatically commence logging of anonymized Bluetooth hardware addresses using an open-source application called Bluelog.

The default OpenAWAM field node configuration anonymizes Bluetooth addresses by performing a CRC-32 cyclic redundancy check on the address in memory, and logging the resulting number. Thus at no time is the system in possession of the actual Bluetooth hardware address.

By default, OpenAWAM logs to **syslog,** a universally available logging system. The critical feature of syslog is that logs can be centralized. Temporary loss of communication only delays transmitting of logged addresses; it does not lose them. Thus, OpenAWAM is tolerant of poor-quality communication links such as wireless: Although real-time data may be impacted due to poor field communication, historical data is highly available.

By default, OpenAWAM continues to log devices that stay near the field node once per minute. This allows the central software the ability to estimate intersection delay, if average delay is sufficiently long. It is also possible to identify and discard data associated with, for example, parked cars, patrons of nearby commercial establishments, and persons waiting at bus stops.

Since each log entry consists only of a timestamp, geographic identifier (in the form of the field node's hostname), and ID, and since log entries only occur if a reportable device is nearby, little bandwidth is consumed, even in the aggregate.

Central Software Stack

The OpenAWAM central software can be run on either Windows or Linux: OpenAWAM is developed in Python, a highly portable scripting language. Cross-platform capability was included at the outset since

[1] Static or dynamic (DHCP) address/netmask/gateway, hostname, and OpenAWAM Central server

Windows is the preferred platform for TMC workstation and server software. Use of Python virtually guarantees full functionality on both platforms with very little, if any, customization required. Python is always included with Linux, and can be easily downloaded and installed on Windows. (The OpenAWAM Python scripts were written from the ground up to be platform-agnostic.)

The OpenAWAM central software operates in three modes, which can be operated independently or concurrently:

- **Real-time traffic monitoring.** A map is continuously displayed showing current travel times.
- **Historic information.** The database of accumulated loggings can easily be queried in a multitude of ways to extract performance information over time to answer questions like: Did anything unusual happen overnight or during the weekend? How did travel times change after I implemented my new timing plans? Is my adaptive system maintaining high mobility? What routes are most commonly driven?
- **Data dissemination.** Travel-time information can be packaged and delivered to your agency's Web site, to other agencies, or to your state DOT or MPO's regional traveler information system.

Bluetooth Technology

Most current Bluetooth devices (including the Bluetooth adapter recommended for use with OpenAWAM) support the Bluetooth Class 1 profile, which allows for communication at distances up to 100 meters or 330 feet. This range ensures robust detection and logging of all Bluetooth devices passing near the node, even if installed at the largest intersections. The 100-meter range together with logging of devices once per minute allows for intersection or link delay to be estimated based on observations at each field node, in addition to the comparison of log entries from multiple locations.

Privacy is maintained by at least two methods:

- Devices which are not set to "discoverable" cannot be logged by this or any other AWAM system.[2] Thus, to be rendered invisible to the system (for instance, due to privacy concerns), all one would need to do is confirm their devices are not discoverable.
- In the default OpenAWAM configuration, the actual hardware address is never logged; instead a proxy is used. Thus, it is not possible to provide, for instance, "all Bluetooth addresses ever logged by the system." The best that can be provided is, "Can you tell me if so-and-so address was recently logged"; and even in this case the answer cannot be known with certainty since the possibility of collisions (i.e., more than one Bluetooth hardware address sharing a checksum) is present. In other words, the system can be queried to yield all logged values that can be associated with a certain hardware address, but this does not guarantee that the logged entry was created by the device in question.

[2] In principle, a Bluetooth device could attempt to discover other nearby Bluetooth devices via a brute-force method, but this procedure is not feasible for AWAM applications since hours are necessary to discover one device, and while operating in this mode the normal scanning function cannot be used.

Present trends in wireless security are such that more and more Bluetooth devices default to not being discoverable. This appears to be offset by both the proliferation of portable Bluetooth devices (i.e. although the proportion of discoverable devices may be going down, the larger total number of devices means plenty of discoverable devices will continue to be available) and by the convenience factor: Some users appear to be overriding the default settings to make their devices discoverable. Nevertheless, the trend toward less devices being discoverable at all times is best combatted by limiting the capital cost of the AWAM system: Another reason to consider OpenAWAM.

Further Development

The system is extensible to provide mode-specific travel-time information, for instance to monitor transit system performance. All that is required is that a known Bluetooth device be installed into the transit vehicle; when transit-related ID's are identified by the system, transit-specific processing can occur. This data can drive, for example, "Next Bus Arrival" message boards, 511 traveler dissemination systems, and transit priority.

The system is capable of being extended to predict travel times on given routes through the use of historic data. This can help avoid the issue of travel time estimation error due to time lag between current conditions and conditions when the trip is actually undertaken.

Conclusions

The OpenAWAM system brings flexible, highly configurable real-time and historic travel-time collection, analysis, and dissemination to the Transportation Management Center for low equipment cost. Every aspect of the system has been designed to minimize capital expenditure.

OpenAWAM is not a turnkey solution. The user will be required to work closely with the maintainer of the field communication network, firewall, and central server to achieve the proper functionality. However, such coordination would be required of any such system that relies on the agency's existing Ethernet communication system, so the additional effort to actually configure the field node and central server (the role which is traditionally performed by the vendor) is small in comparison to the total project effort.

About the Author

Since 2006, John Kerenyi has served as a Senior Engineer for the City of Moreno Valley, a community of 200,000 in Riverside County, California. He is currently the lead engineer for ITS deployment, responsible for design, delivery, and system integration of the City's Transportation Management Center and Arterial Traffic Management System. He is a registered Traffic Engineer and Electrical Engineer in California. Previously he was employed by Kimley-Horn and Associates and KOA Corporation. He graduated with a Bachelor degree in Engineering from Harvey Mudd College in Claremont, California.

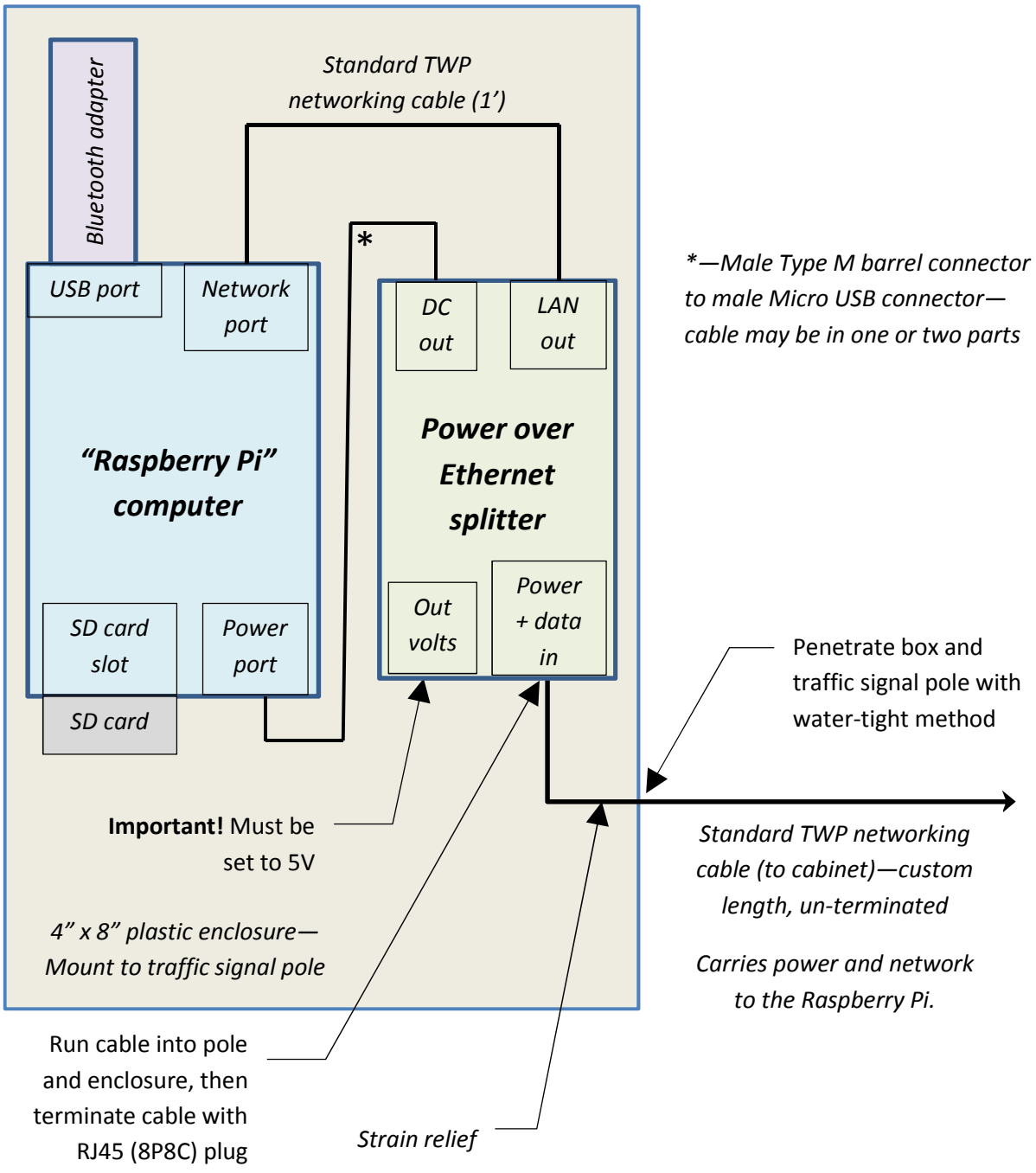

Figure 1. OpenAWAM Field Node Block Diagram

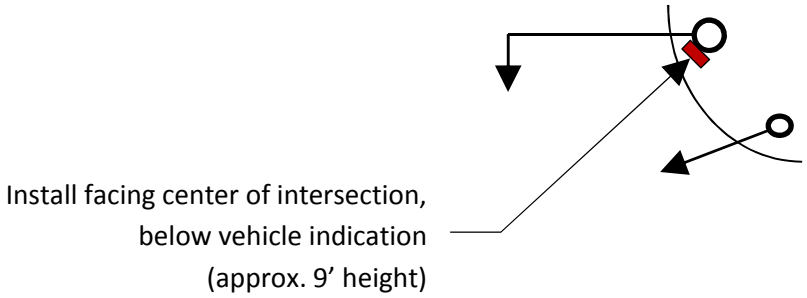

Figure 2. Plan View of OpenAWAM Field Node Installation

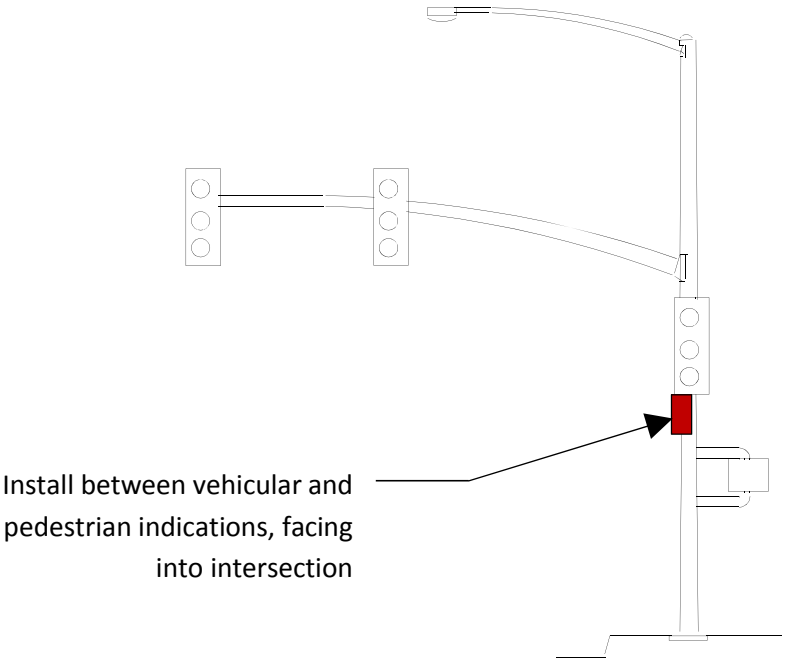

Figure 3. Elevation of OpenAWAM Field Node Installation

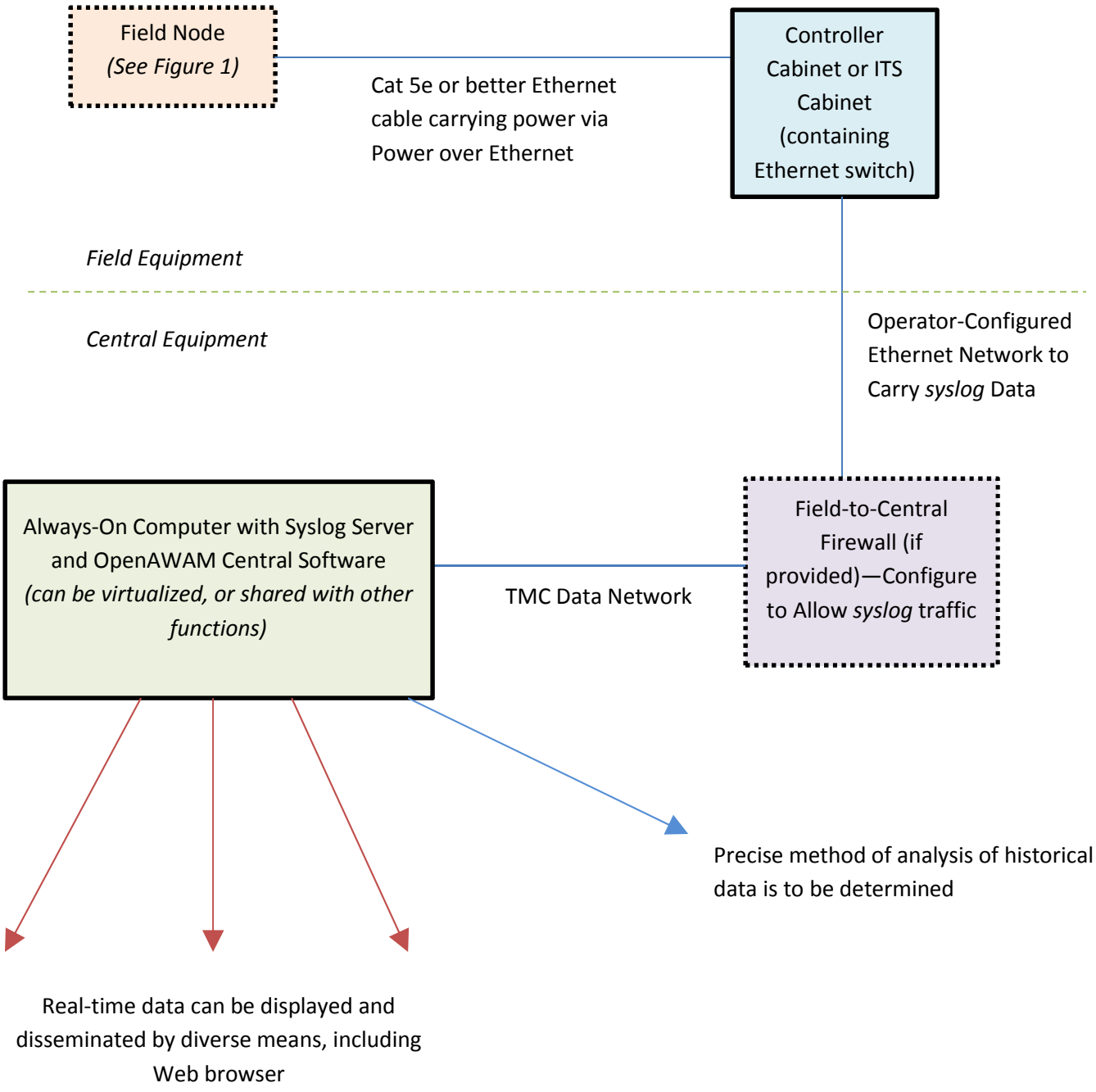

Figure 4. System Block Diagram