# Demo of the Medical Device Dongle: An Open-Source Standards-Based Platform for Interoperable Medical Device Connectivity*

Philip Asare
Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, United States
asare@seas.upenn.edu

Danyang Cong
Santosh G Vattam
Computer and Info. Science
University of Pennsylvania
Philadelphia, United States
{cdanyang, vattam}@seas.upenn.edu

BaekGyu Kim, Oleg Sokolsky, Insup Lee
Computer and Info. Science
University of Pennsylvania
Philadelphia, United States
{baekgyu, sokolsky, lee}@seas.upenn.edu

Shan Lin
Dept. of Computer Science
Temple University
Philadelphia, United States
shan.lin@temple.edu

Margaret Mullen-Fortino
University of Pennsylvania
Health System
Philadelphia, United States
margaret.fortino-mullen@uphs.upenn.edu

## ABSTRACT

Emerging medical applications require networked coordination between medical devices. However, most of the medical devices in use today do not support wireless communication or network connectivity. Currently, hospitals interested in coordinated medical care would have replace existing devices with expensive new devices. We believe that existing medical devices can be extended to support interoperable network connectivity. We demonstrate the Medical Device Dongle (MDD), an open-source platform that enables such extensions to current medical devices. The MDD can attach to any device that has a data output interface (RS-232 or USB) and enables it to connect wirelessly and in an interoperable manner for various applications. We show how multiple medical devices, including pulse oximeters and infusion pumps, can be connected and controlled together using an open-source platform, standards-based connectivity protocols, and model-driven software. The demo setup consists of medical devices attached to an MDD Agent, an MDD Manager device, and a mobile phone running monitoring applications. The MDD components can communicate over Bluetooth, WiFi and Ethernet.

*This research was supported in part by NSF CNS-0834524, NSF CNS-0930647, NSF CNS-1035715, and NIH 1U01EB012470-01. POC: Insup Lee, lee@cis.upenn.edu


*Wireless Health '11,* October 10 - 13, 2011, San Diego, USA


## Categories and Subject Descriptors

J.2 [**Computer Applications**]: Life and Medical Sciences; D.2.12 [**Software Engineering**]: Interoperability; H.4.3 [**Information Systems Applications**]: Communications Application

## General Terms

Design

## 1. INTRODUCTION

Emerging medical applications require coordination between medical devices [2]. In [2], for example, a smart alarm running on the central patient monitor needs data from all vital sign monitors connected to a patient in order for the alarm to correctly notify the practitioner of safety risks. Other applications may require medical devices to communicate directly with each other. Regardless of how coordination is done, all devices must be capable of network connectivity and support the same set of protocols to ensure interoperability. Unfortunately, many of the existing medical devices in use were not originally designed for network connectivity [3]. Many of these devices, however, provide a communication port (usually RS-232 or USB) for data exchange with a single computer. Such devices can adapted for network connectivity by providing a peripheral that connects to their output ports.

Such peripherals do exist [4], and even though they use a standard protocol like IEEE 11073-PHD [1], these peripherals are more likely to support personal health devices that typically are not used in the coordination applications we are interested in. Also, the implementation of the protocols are closed source making such peripherals unsuitable as testbeds or platforms for research in interoperability. The aim of the Medical Device Dongle (MDD) is therefore two-fold: (1) to enable existing devices connect and communicate using a standard protocol to allow and support development of distributed medical applications, (2) to provide a platform and testbed for research in medical device interoperability.

## 2. MDD OVERVIEW

Applications of interest typical require that medical devices connected to a patient must interact with a central point (usually called the supervisor). This supervisor manages the coordination between the devices and runs applications that use these connected medical devices. There are, therefore, two versions of the the MDD, designed to fit into this *multiple-devices-single-manager* architecture: one for devices connected to the patient (the agent MDD); and another for the supervisor (the manager MDD). The MDD can be implemented physically as a peripheral that connects to a device or logically as software components running on the device (with access to the device's network interface). The main part of the MDD is this collection of software components, which we call MDDWare. This demo shows the agent MDD implemented as a physical peripheral and the manager MDD implemented is a physical device in one case and logical software components in another.

Rather than develop our own interoperability protocol, we based the MDD on the IEEE 11073 protocol. The MDD is also designed to support other medical device and interoperability protocols, especially those geared towards maintaining patient safety. We chose 11073 because it is an IEEE standard and supports the multiple-devices-single-manager model. The network architecture is based on the older point of care (PoC) standard, while the connectivity and communication protocol is based on the personal health devices (PHD) protocol. We chose PHD for the main protocol because it has better and more recent documentation than the previous PoC standard. It is important to note that we did not implement the full standard, but only those parts we deemed sufficient for medical device connectivity and communication. In particular, we implemented: 1) manager and agent finite-state machines for maintaining connectivity and providing GET and SET services for message exchange and command execution; 2) the Medical Device System (MDS) object from the domain information model (DIM) for device description; 3) the Medical Device Encoding Rules (MDER) from the communication model for encoding messages. We call this implementation 11073-MDD. The MDD turns any medical device with a data output interface (RS-232 or USB) into an 11073-MDD-compliant device.

## 3. DEMO SCENARIO

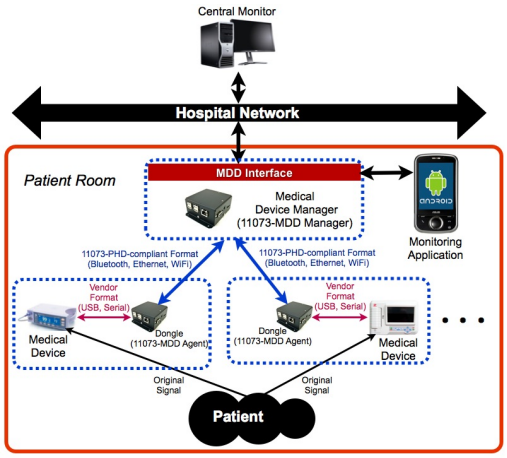


**Figure 1: Device Connectivity Architecture**

Supposing the patient room is an Intensive Care Unit (ICU), medical devices are connected to the patient to measure vital signs such as pulse oximetry ($SpO_2$), heart rate or to administer treatment such as insulin infusion. Figure 1 shows the scenerio of our demo, which consists of three main parts: 1) *Mutiple medical devices*: the data source for the manager; 2) *Agent MDD*: the intermediary between medical devices and the MDD manager; and 3) *Manager MDD*: An embedded device or smartphone. The relevance of MDD in such a scenario is highlighted as follows:

**Agent MDD.** The agent MDD behaves like a translator between the manager MDD and the medical device. It translates any data from the device into IEEE 11073-PHD compatible formats and transmits the data to the MDD Manager over a network interface.

**Manager MDD.** The medical device manager is in charge of coordination of different devices. It monitors the connection of the supervisor to agent devices and is responsible for sending and receiving messages on behalf of medical applications such as a smart alarm [2] running on the supervisor directly connected to the manager as shown in Figure 1.

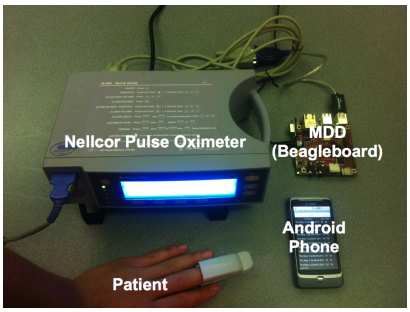


**Figure 2: MDD Implementation Setup**

**Setup.** Figure 2 shows one basic demo setup with a Nellcor Pulse-oximeter, the MDD Agent (Beagleboard) and an Android phone running a logical MDD Manager and a medical application. Another demo (not shown here) has the Android phone communicating with a stand-alone manager device using the MDD Interface (as shown in Figure 1). Each MDD (agent or stand-alone manager) is a BeagleBoard running Linux (Ubuntu 10.04 LTS)and is capable of Bluetooth and WiFi communication.

## 4. REFERENCES


[1] ISO/IEC/IEEE Health informatics–personal health device communication–part 20601: Application profile–optimized exchange protocol. *ISO/IEEE 11073-20601:2010(E)*, pages 1 –208, 1 2010.

[2] A. L. King, A. Roederer, D. Arney, S. Chen, M. Fortino-Mullen, A. Giannareas, W. Hanson, III, V. Kern, N. Stevens, J. Tannen, A. V. Trevino, S. Park, O. Sokolsky, and I. Lee. GSA: a framework for rapid prototyping of smart alarm systems. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI '10, pages 487–491, New York, NY, USA, 2010. ACM.

[3] K. Lesh, S. Weininger, J. M. Goldman, B. Wilson, and G. Himes. Medical device interoperability-assessing the environment. In *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability*, HCMDSS-MDPNP '07, pages 3–12, Washington, DC, USA, 2007. IEEE Computer Society.

[4] C.-Y. Park, J.-H. Lim, and S. Park. ISO/IEEE 11073 PHD standardization of legacy healthcare devices for home healthcare services. In *Consumer Electronics (ICCE), 2011 IEEE International Conference on*, pages 547 –548, Jan. 2011.