SenseBox: A Low-Cost Smart Home System

Joseph Taylor*, H M Sajjad Hossain*, Mohammad Aril Ul Alam*, Md Abdullah Al Hafiz Khan*,
Nirmalya Roy*, Elizabeth Galik†, Aryya Gangopadhyay*
*Department of Information Systems, University of Maryland Baltimore County
†University of Maryland School of Nursing
{j.taylor, hmsajja1, alam4, mdkhan1, nroy, gangopad}@umbc.edu, Galik@son.umaryland.edu

***Abstract*—Smart home technologies are getting acclaimed for providing a wide variety of functionalities - security, appliance control, HVAC control and remote monitoring. Installation cost and interoperability issues have restricted the adaptability of these technologies. In this demo paper, we demonstrate the design of an interoperable prototype of our smart home system, *SenseBox* using low-cost embedded device as part of an NSF I-Corps project. We integrated an off-the-shelf commercially available inexpensive ($12) hardware (PogoPlug). We paired commercially available PIR sensors, door sensors and moisture sensors with our smart home system prototype to monitor human activities. We deployed our prototype and collected real-time activity data traces from a retirement community center. We articulated the SenseBox development, integration and testing challenges and insights along with real deployment experience.**

I. Introduction

Activity monitoring using smart home technologies has enabled a great deal of opportunities in pervasive and ubiquitous computing environment. Current smart home technologies lack the ability to recognize human activities unobtrusively. Activity enabled smart home systems can provide meaningful insights to developers and end-users by distinguishing between multiple individuals in physical spaces and providing context-specific automated actions like HVAC adjustment, reminder of pills intake, or timely commercial advertising. Remote activity monitoring can also help to enable profiling functional and behavioral health of those being monitored longitudinally. Another issue with current smart home technologies is that they are often prohibitively expensive. In this paper, we demonstrate the development of a low-cost unobtrusive smart home system prototype for activity monitoring in a retirement community environment. Identifying a cost performance equilibrium was critical for our application. We found that computational needs were too great for eight and sixteen-bit micro-controllers, but embedded x86 hardware was too costly. After much experimentations, we determined that ARM-based hardware may likely be the best choice. We tested ARMv5 and ARMv6-based devices, both without any hardware floating-point capabilities, and postulated either would likely be adequate, with the former potentially requiring some more careful optimization to handle tasks that are more computationally-intensive. We have developed an early prototype using the Raspberry Pi [1] but we planned to develop a more resource optimized and low-cost platform. We selected the inexpensive ($12) Cloudengines PogoPlug [2] as a hub to test our prototype. This hardware configuration bears some similarities to the Marvell DreamPlug, which has been used in sensor network deployments as described in [3]. Table I shows the properties and cost of different potential devices we have tested. In this demo, we demonstrate the hardware and software architecture of SenseBox along with how our system helps to capture and process data in real time and detects activities of daily living.

TABLE I. Device properties

Device	CPU	Storage	Memory	Cost
Lab Workstation	2.9 GHz	1 TB	8GB	$ 1200
Raspberry-Pi	900 MHz	256 MB	N/A	$ 20
PogoPlug	700 MHz	128 MB	128 MB	$ 10
Arduino	16 MHz	512B	2-256 KB	$ 15
DreamPlug	1.2 GHz	512 MB	4GB	$200

II. System Protocol

One of our goals in this SenseBox development project is to address the problem of interoperability which open source automation software OpenHab [4] addresses quite intensively. But it appears in our case that we are too limited in computational resources to design a complete system like OpenHab. Even in spite of this, we wanted to make our design easy to integrate in to other systems. In pursuit of this, we first evaluated a few protocols that are common in industry in order to identify a standard way to pass messages from sensors to other application threads. We planned to do this in a fairly standard way, using normal IP packets on a standard IPv4 or IPv6 network. In selecting the protocol, we evaluated two popular standards: CoAP and MQTT. From our evaluation, we found that either of these protocols works fine, but in CoAP the sensors act as servers instead of clients. This means agents listening for events (databases, software analyzing events looking for state changes, etc.) need to be connected to the sensors. This leads to some limitations in network architecture that we waned to avoid. MQTT sensors

978-1-5090-4338-5/17/$31.00 ©2017 IEEE

communicate to a MQTT server ("broker") and this traditional architecture is more amenable to things like NAT. A MQTT server (or "broker") accepts updates from sensors, and forwards those updates to clients who want them. As MQTT forwards and forgets messages so we implemented a listener and developed a database to listen and store messages respectively, and retrieve them later for storage or analysis. We used a MQTT "wildcard" listener and a lightweight database like Berkeley DB for this purpose. Bluetooth messages from the sensors to our hub were received and forwarded to the MQTT broker.

A. *Hardware*

The PogoPlug software is covered under the GNU General Public License (GPL), and CloudEngines publishes the build changes to the kernel and userland software in tarballs on its public website. The PogoPlug has an exposed USB port (potentially expanded to many ports by use of a hub,) an internal universal asynchronous receiver/transmitter (UART) and a gigabit Ethernet port. An SD slot is present on the device, but I/O to the device is "bit-banged" so file transfers are CPU-intensive, and too slow for many applications. For a general-purpose sensing hub, we required 3 wireless interfaces: 802.11 wireless, Bluetooth 3, and Z-Wave wireless. We identified a $20 combined Bluetooth/WiFi dongle that met the first two requirements without the use of a USB hub. Z-wave needs were met by a device that can interface with the internal UART, reducing costs and eliminating the need for a USB hub which gives a more integrated appearance.

B. *Software*

The PogoPlug ships with a Linux 2.6 kernel [5] and a modified U-Boot bootloader. The device has 128MB RAM and 128MB flash memory, formatted as UBIFS. Most userland tools are provided by Busybox, including *init*. Some additional packages like Dropbear SSH and FFMPEG are available on the stock device. Our first task was to rebuild the kernel to include support for several devices that were omitted in the stock configuration. The PogoPlug ships without support for 802.11 devices, V4L webcams and NAT/iptables firewalling, all of which during the prototyping phase required in our smart home deployment scenario and were frequently encountered during the prototyping phase.

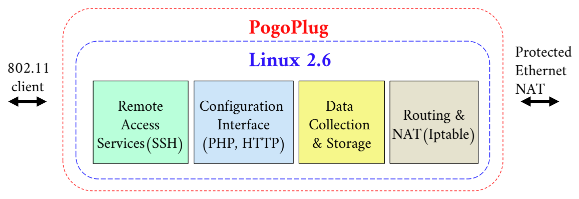
PogoPlug
Linux 2.6
802.11 client
Remote Access Services(SSH)
Configuration Interface (PHP, HTTP)
Data Collection & Storage
Routing & NAT(Iptable)
Protected Ethernet NAT

Fig. 1. Software architecture

III. Implementation

A toolchain was built using Buildroot, and the 2.6 source distributed by Cloudengines was rebuilt to support the additional components we required. Some trimming was necessary to make the kernel fit in the reserved kernel space; support for File Allocation Table (FAT) and New Technology File System (NTFS) was removed along with some other features we did not require. Some additional miscellaneous tuning was done to meet the preferences of developers, like changing the I/O scheduling engine from Completely Fair Queuing (CFQ) to deadline. Several different software packages were built and shown to work well, including the Lighttpd web server, autossh for tunnel maintenance, and RTL-SDR for collecting RF data from sensors or the local environment. Some tools in support of the additional kernel components were also built, including the userland components of iptables and tools for video capture and streaming for ground truth collection purpose. For machine learning applications, some additional tools were required. We investigated several different options and settled on MLPACK [6], which supports most learning algorithms in the context of activity discovery and recognition. Additionally, a web-based interface was built to observe the status of the device and attached sensors using the Klone web server [7] and later with Lighttpd+PHP [8]. Tools for remote access and administration that alleviated NAT issues were also explored, since NAT is a common issue we faced when deploying in end-user environments.

IV. Demo Description

We were able to use our prototype system in a real-world deployment. We installed our prototype in 20 residents apartments to monitor the activities of daily living - laundering clothes, using the telephone, sweeping, taking out trash, drinking water, etc [9] - IRB (#HP-00064387). The apartments were not equipped with Ethernet connectivity, offering only Wi-Fi internet access. We are currently using a proprietary sensor suite, which requires Ethernet connectivity to communicate activity data. We utilized our PogoPlug-based system and custom kernel to create a NAT accessible on the PogoPlug Ethernet port, and then routed the traffic on to the wireless network using a USB 802.11 adapter configured as a client as shown in Figure 1. Once configured this was observed to be very stable, no issues were encountered and network connectivity was not interrupted during any of the 20 test deployments. During the test deployments, our system was left in the volunteers residence for 24 hours. Because there was no observer during the majority of that time, it was critical of fault-tolerant remote access system in place in case of remote diagnostics or shutdown, if necessary. Our SSH-based system allowed us to do remote troubleshooting and diagnostics, subverting the NAT and providing a secure, encrypted tunnel with an authentication scheme that prevents attacks like man-in-the-middle and renders packet sniffing ineffective. Using this tunnel we are able to securely run code on the remote device, and collect information about its current status. While remote diagnostics were never necessary as the device remained stable during all deployments, these facilities were still tested. Logging systems were developed to automatically notify the group when a node changed status (e.g., on/offline) while also periodically storing logs from the remote device.

A. Demo Showcase Plan

In this demo session, we will demonstrate our low-cost smart home system with some sensors. The sensors will capture the movement of the participants and the system frontend will show the detected events. Also we will log the state (CPU/memory utilization, status of encrypted tunnel, uptime etc) of our SenseBox system in the frontend. We will need a small area to setup the sensors and we will bring our own hardwares.

V. Results

To evaluate our prototype we focus on three specificities - Memory, CPU and Disk space.

A. Memory

Under normal conditions, memory consumption remains at about 20%, but it deviates a lot. Activities like diagnostic web page generation or file transfer increase this marginally, but rarely beyond 30%. When training or running MLPACK models, memory consumption increases appreciably. Models and parameters have to be selected carefully, as we run the risk of memory exhaustion which will cause undesired behavior.

B. CPU

When not running or training models, the device is almost entirely idle. When a model is being evaluated, CPU usage predictably increases to near 100% for the duration of execution.

C. Disk Space

PogoPlug uses Linux's memory technology device (MTD) framework [10] to provide what appears to applications as a standard disk drive. The stock PogoPlug device uses UBIFS, and we felt this was a reasonable choice. While incurring some overhead in usable space, the wear-leveling provided by UBIFS should increase the lifespan of our device. The layout of the partitions in flash are like the following

```
dev:    size   erasesize  name
mtd0: 00200000 00020000 "u-boot"
mtd1: 00300000 00020000 "uImage"
mtd2: 00300000 00020000 "uImage2"
mtd3: 00800000 00020000 "failsafe"
mtd4: 07000000 00020000 "root"
```

Additionally, a temporary, in-memory filesystem is created on boot to store files that are modified frequently, again with the intention of reducing wear to flash memory. Examples of files stored in the volatile temporary partition include logs and the on-disk list of DHCP leases.

VI. Conclusions

The presence of home automation devices and sensors are getting more prominent but for efficient exploitation these systems need to be context aware which can infer different human contexts on the fly. In this demo, we described an efficient and optimized design of an expert smart home system, *SenseBox*. Data collected by our prototype has been used in our work [11] [12] [13]. As of now we have not fully implemented our activity recognition algorithm using the PogoPlug and MLPACK, but as device prototyping and base system sotfware testing is complete, implementing activity recognition is our next goal. Careful tuning of the model will be required to deal with the limitations of our platform. We have also found that off-the-shelf sensors are fairly expensive and do not perform to our expectations, so we have begun testing our own Bluetooth Low Energy-based sensors with promising results.

References

[1] Raspberry Pi 2 Model B. https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

[2] CloudEngines PogoPlug. https://pogoplug.com/.

[3] P. Dawadi, D. Cook, and M. Schmitter-Edgecombe. Automated Cognitive Health Assessment from Smart Home-Based Behavior Data. *IEEE J Biomed Health Inform*, Aug 2015.

[4] Openhab. http://www.openhab.org.

[5] PogoPlug mobile 2.6.31.8 kernel. https://pogoplug.com/opensource.

[6] MLPACK. http://www.mlpack.org/.

[7] Klone Web Server. http://www.koanlogic.com/klone/.

[8] Lighttpd+PHP. https://wiki.ubuntu.com/Lighttpd%2BPHP.

[9] Umbc-umb partnership awards a catalyst for collaboration: Research at umbc news feb 17 2015. http://research.umbc.edu/umbc-research-news/?id=49901.

[10] Memory technology device. http://www.linux-mtd.infradead.org/doc/general.html.

[11] H M Sajjad Hossain and et al. Active learning enabled activity recognition. *IEEE PerCom*, 2016.

[12] Mohammad Arif Ul Alam and et al. CACE: exploiting behavioral interactions for improved activity recognition in multi-inhabitant smart homes. In *36th IEEE ICDCS 2016, Nara, Japan, June 27-30, 2016*, pages 539–548, 2016.

[13] M. A. U. Alam, N. Roy, and et al. Automated functional and behavioral health assessment of older adults with dementia. In *2016 IEEE CHASE*, pages 140–149, 2016.