SIP Protocol as a Communication Bus to Control Embedded Devices

Ramunas DZINDZALIETA
Institute of Mathematics and Informatics
Akademijos str. 4, Vilnius Lithuania
ramunas.dzindzalieta@gmail.com

Abstract. SIP (The Session Initiation Protocol) is widely used as the signaling protocol for various services in the omnipresent environment. SIP is textual based protocol so the ability to process SIP messages quickly is critical for the performance of consumer electronic devices, such as SIP phone and the various Gateway. Our work enables homogeneous communication between heterogeneous distributed devices. Using SIP protocol communication bus to control widely varying entities, including ZigBEE, serial based (RS232) X10 and other based devices.

Keywords. SIP, communication protocol, embedded system

Introduction

In the network, SIP is implemented in many devices such as the Gateway and various terminals [1]. We propose a ubiquitous service system in network based on SIP, which provides multimedia services and remote control services. It takes use of SIP mobility to preserve service session, even though end-user moves from one computing environment to others. The architecture is implement using SIP and gateway are connected to a sensors environment, such as temperature sensors to realize that at anywhere or SIP phone and home servers connected to Internet. Network device share the ability to interact with physical environment (by sensors or actuators) and have a computing power limited.

The platform of software and hardware is based on the SIP. The main aim is to develop sensor network that will be control, monitored or keep a continuous record of something remotely by smart devices using SIP as a container for collecting data.

1. SIP Project Overview

We want to prove that SIP is good basis to form a communication bus between heterogeneous devices. Extensibility: SIP is request/response protocol, transport-independent and text-based. Like HTTP, SIP is extensible in terms of methods, headers, and message payload. Message payload is format irrespective and SIP supports most kind of data (e.g., SDP, presence information, and SOAP or XML). SIP has provided communication forms, such as commands (RPC-like based on instant messaging), events, and sessions of data streams. SIP platforms are already widely deployed in various forms, because it is standard for IP telephony. SIP protocol become available in various devices such as SIP phone and other including dedicated IP based systems.

2. Building SIP Adapters

We use of SIP as a communication bus between sensors and heterogeneous distributed systems. Let us testing how sensors need to be adapted to connect them to the SIP communication bus.

2.1. Architecture of a SIP Adapter

We must supply entrance to its functionalities via SIP – accepting mechanisms. To entrance to it functionalities are defined of the three interaction modes available in SIP: commands (i.e., control device and status query), events (i.e., event publishing and subscription) and sessions (i. e., invitation to a session of data stream) [2, 3]. Need to request and receive data that may have different formats: command-parameter values (e.g., SOAP), event values (e.g., XML format) and session capability descriptions (e. g., using plain text SDP).

SIP adapters to provide a useful function with an interpreter. The layer obtain from the payload of a SIP message for a given SIP methods, the constituent parts of the corresponding interaction mode (i.e., session, command or event). For example, a SIP request with a MESSAGE method corresponds to a command interaction. The payload interpreter then provides from the request payload a SOAP message, indicating the command name (e.g., *getSensorsParameter*) and the parameter values.

3. Enabling SIP Communication

SIP project consists of two main components:

- a hardware platform made of sensors and actuators connected to Ethernet shield embedded systems with network capabilities;
- specific software on embedded systems implementing the SIP protocol to communicate with the sensors;

The sensor hardware gateway which is based on open source hardware:

- The Atmega board running with I/O connections (USB, serial line) and an Ethernet board to act as a SIP gateway.
- The sensors connected via serial line connection.
- Applications have already been developed to drive sensors on the Atmega board and to collect data from sensor via SIP messages.

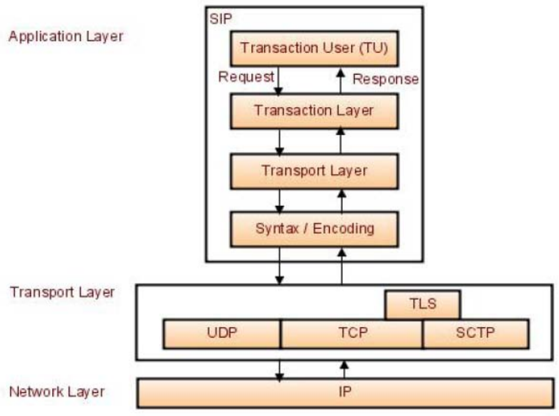


Figure 1. SIP layer protocol

SIP is a layer protocol, comprising the syntax and encoding, transport, transaction, and transaction user (TU) layers [4]. Syntax and encoding layer specifies message format and structure. The transports layer defines how SIP entities send/receive messages over the networks. Above the transport layer is the transaction layer, and the TUs (SIP entities except for the stateless proxies) are in the top layer. SIP transaction layer is the most important layer as it is for request-response matching and retransmission handling.

In SIP, the names found in the fields 'To:' and 'From' are encoded as Universal Resource Locators (URLs). It is useful to define a new type of URL without changing the nature of the protocol to enable better discovery of the device's network address.
The structure of the existing SIP address: <entity>@<location> is maintained even when extended to accommodate the devices' names.

A web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An information resource is identified by a Uniform Resource Identifier (URI) and may be a web page or other piece of content. Hyperlinks present in resources enable users easily to navigate their browsers to related resources. A web browser can also be defined as an application software or program designed to enable users to access, retrieve and view documents and other resources on the Internet.

The board also can connect to a wired network via Ethernet. When connecting to a network, you will need to provide an IP address and a MAC address. The Ethernet Library is fully supported. Using the Ethernet, device will be able to answer a HTTP request. When navigating to your Ethernet shield's IP address, will respond data for a browser to display the input values from all analog pins as Figure 2.

The power pins are as follows:

VIN. The input voltage to the board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V. The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

3V. Supply generated by the on-board regulator. Maximum current draw is 50 mA.
GND. Ground pins.
Digital I/O Pins. 14 pins (of which 4 provide PWM output)

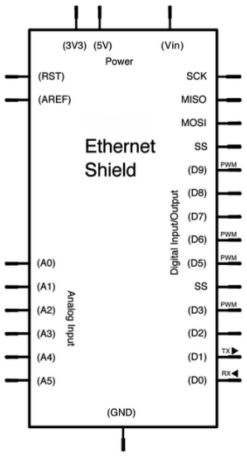
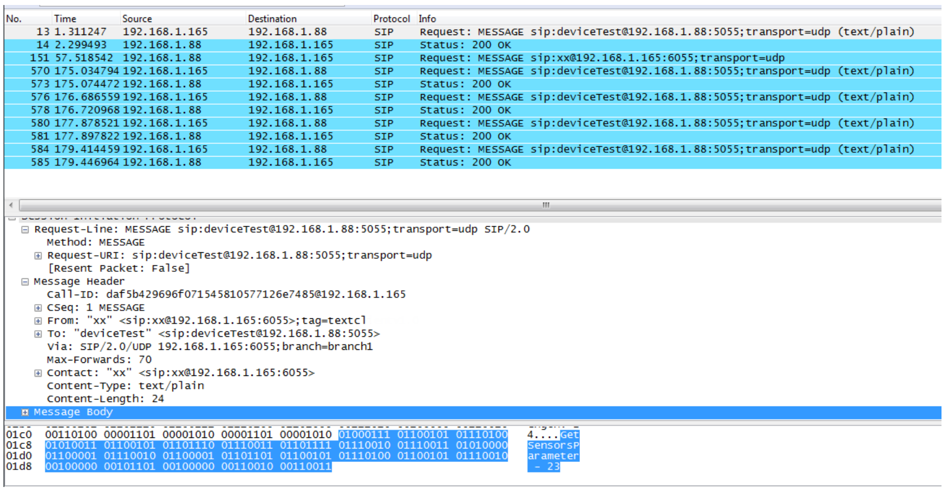


Figure 2. Ethernet shield

The SIP stack that was chosen to develop applications is the oSIP stack to SIP application writing. It is written in C language, is very portable. oSIP supports many transport protocols such as TCP, UDP, and TLS (Transport Layer Security). The GNU oSIP library is written in C and gets no dependencies except the standard C library [5]. oSIP is thread safe and will generally be used in a multi-threaded application. Nevertheless, this is optional. Software was implemented libosip2parser librarary from http://www.gnu.org/s/osip/ has been modified to compile and run on the AtMega board. oSIP is little in size and code and thus could be used to implement IP soft-phone as well as embedded SIP software. oSIP is not limited to endpoint agents, and can also be used to implement "SIP proxy".

oSIP does not intend to provide a high layer API for controlling “SIP Session” at this step [6]. Instead, it currently provides an API for the SIP message parser, SDP message parser, and library to handle “SIP transactions” as defined by the SIP document.



```
No.   Time       Source         Destination    Protocol Info
   13 1.311247   192.168.1.165  192.168.1.88   SIP      Request: MESSAGE sip:deviceTest@192.168.1.88:5055;transport=udp (text/plain)
   14 2.299493   192.168.1.88   192.168.1.165  SIP      Status: 200 OK
  151 57.518542  192.168.1.88   192.168.1.165  SIP      Request: MESSAGE sip:xx@192.168.1.165:6055;transport=udp
  570 175.034794 192.168.1.165  192.168.1.88   SIP      Request: MESSAGE sip:deviceTest@192.168.1.88:5055;transport=udp (text/plain)
  573 175.074472 192.168.1.88   192.168.1.165  SIP      Status: 200 OK
  576 176.686559 192.168.1.165  192.168.1.88   SIP      Request: MESSAGE sip:deviceTest@192.168.1.88:5055;transport=udp (text/plain)
  578 176.720968 192.168.1.88   192.168.1.165  SIP      Status: 200 OK
  580 177.878521 192.168.1.165  192.168.1.88   SIP      Request: MESSAGE sip:deviceTest@192.168.1.88:5055;transport=udp (text/plain)
  581 177.897822 192.168.1.88   192.168.1.165  SIP      Status: 200 OK
  584 179.414459 192.168.1.165  192.168.1.88   SIP      Request: MESSAGE sip:deviceTest@192.168.1.88:5055;transport=udp (text/plain)
  585 179.446964 192.168.1.88   192.168.1.165  SIP      Status: 200 OK

  ⊟ Request-Line: MESSAGE sip:deviceTest@192.168.1.88:5055;transport=udp SIP/2.0
      Method: MESSAGE
    ⊞ Request-URI: sip:deviceTest@192.168.1.88:5055;transport=udp
      [Resent Packet: False]
  ⊟ Message Header
      Call-ID: daf5b429696f071545810577126e7485@192.168.1.165
    ⊞ CSeq: 1 MESSAGE
    ⊞ From: "xx" <sip:xx@192.168.1.165:6055>;tag=textcl
    ⊞ To: "deviceTest" <sip:deviceTest@192.168.1.88:5055>
      Via: SIP/2.0/UDP 192.168.1.165:6055;branch=branch1
      Max-Forwards: 70
    ⊞ Contact: "xx" <sip:xx@192.168.1.165:6055>
      Content-Type: text/plain
      Content-Length: 24
  ⊞ Message Body

01c0  00110100 00001101 00001010 00001101 00001010 01000111 01100101 01110100   4....Get
01c8  01010011 01100101 01101110 01110011 01101111 01110010 01110011 01010000   SensorsP
01d0  01100001 01110010 01100001 01101101 01100101 01110100 01100101 01110010   arameter
01d8  00100000 00101101 00100000 00110010 00110011                               - 23
```

Figure 3. SIP Message between devices

The goal of the project was to make the client (smart phone for examples android) and the servers (our Atmega board) communicate. First of all, the client had to contact the server. Once the communication was established, the server regularly sent the message with temperature data it collected from the sensors. The Android platform can be applied to embedded systems such as control devices with sensors. Mostly embedded systems, C and C++ languages are used as effective languages to control devices, it's very portable and has very low footprint. Embedded devices will be controlled via the Android platform. Developers should create applications by using Java language. But applications written in Java are slower than applications written in native C/C++ languages when the program needs to execute complex operations.

4. Experimental Study

The main aim of this work is to design and implement an automation platform based on SIP. Experiments have been made as in Figure 4. In this environment was introduced by various automation objects, sensors of motion or ranging from smart phone to home devices. This platform is used as a vehicle to experiment with various scenarios. For example, we have developed a ruling application that involves 220V relay, X10 alarms, SIP phones and other sensors as Figure 4. We developed experimental platform which consists: SIP gateway that directly connected with different sensors. The data of sensors is available through the SIP communication bus. For the implementation, we used Atmega board with 16 MHz processor, 4 MB flash memory and Ethernet board.

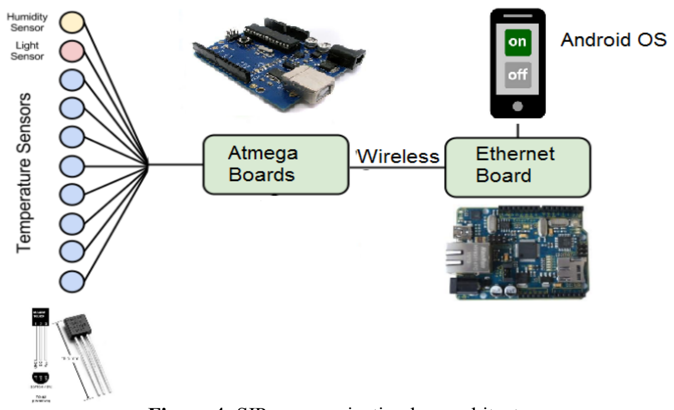


Figure 4. SIP communication bus architecture

5. Conclusions

We presentation homogeneous communications between distributed objects. This presentation relies on the use of SIP as a communication bus for pervasive computing environments. We described a programming support to integrate various objects to the SIP communication. These objects have then been integrated into different of applications for automation systems.

References

[1] B. Bertran, A SIP-based home automation platform: an experimental study. In: *Proceedings of 13th International Conference on Intelligence in Next Generation Networks "Beyond the Bit Pipes"*, IEEE, 2009, 1-6.
[2] B. Campbell, J. Rosenberg, and H. Schulzrinne, C. Huitema, and D. Gurle, *Session Initiation Protocol (SIP) Extension for Instant Messaging*, RFC 3428, The Internet Society, December 2002.
[3] A. B. Roach, *Session Initiation Protocol (SIP) – Specific Event Notification*, RFC 3265, The Internet Society, June 2002.
[4] J. Rosenberg and H. Schulzrinne, *Reliability of Provisional Responses in the Session Initiation Protocol (SIP)*, RFC 3262, The Internet Society, June 2002.
[5] A. Moizard, *The GNU oSIP Library*, 2012. Available from: http://www.gnu.org/software/osip/.
[6] J. Rosenberg, H. Schulzrinne, and G. Camarillo, *SIP: Session Initiation Protocol*, RFC 3261, The Internet Society, June 2002.