UNIVERSITÁ DEGLI STUDI DI PISA
Dipartimento di Ingegneria dell'Informazione
Scuola di Ingegneria
Corso di laurea triennale in Ingegneria Informatica

TESI DI LAUREA

Deployment of an AWS on La Mare Glacier

RELATORI
Prof. Marco AVVENUTI

CANDIDATO
Federico ROSSI

ANNO ACCADEMICO 2016/2017

Abstract

Nowadays it's very important to monitor sensible environments like glaciers in order to keep track of climatic changes or particularly dangerous weather conditions. To pursue this goal, automatic weather station (AWS) are employed. The term "automatic" refers to the ability of the station to remain active nearly without the action of an operator, using energy harvesting techniques to power its component.

The AWS used on the glacier of La Mare is a data logging capable device that sends glacier's environment measurements to a remote server via satellite communication. Then data are redirected to a FTP server, from which they're retrieved,elaborated and represented in a web application. The system has been operative for a few years until January 2015, when data from the device stopped to be sent.

The main goal of this work is to analyse the system in order to obtain more knowledge about it to build a documentation for future maintenance and upgrades of the system.

Contents

1 System introduction

In this section the whole system has been described by its component, without specifying implementation details that will come later. The system is composed essentially by these components:

1. CR1000 Datalogger from Campbell Scientific
2. Michrosat 2403 Satellite Modem from Wireless Innovation
3. Loggernet server by Campbell Scientific hosted by Wireless Innovation
4. Remote FTP server
5. Global Sensor Network (GSN) web application server

1.1 The Datalogger

Basic functions [3] Simplifying, the CR1000 Datalogger can read data from sensors connected to both its digital and analog pins and store them in tables in its flash (EEPROM) memory.

Sensors on board [1] The datalogger has the role to enable sensors by giving them power and acquiring measurements form them. The sensors connected to the datalogger at the moment are:

- Anemometer: Pulse sensor that measures wind speed and direction
- Nivometer: Sonic sensor that measures snow height
- Albedometer: Solar radiation
- Thermo-Igrometer: Air temperature and humidity
- Pirgeometer: IN/OUT long-wavelength radiation
- Thermistors: Temperature from Thermo-Igrometer shields

Programming Actions and table definitions are specified in a source file written in a custom version of BASIC called CRBASIC. Then the program is flashed into the datalogger's memory and it's executed when the device is switched on. The code is shown in appendix A

Energy harvesting The main concern about the deployment of the datalogger is the energy supply management; in fact the device has a limited power storage provided by a 12V battery but it has to be energy autonomous. However the datalogger has the capability to harvest energy from its solar panel. So it's possible to code programs that run on the device and respect the following energy constraint [1]:

$$E_{harvested}(\Delta t) \geq E_{consumed}(\Delta t) \tag{1}$$

1.2 Satellite communication

Iridium The satellite infrastructure is provided by Iridium Satellite Communications that offers a dense constellation of satellites providing an high available service within optimal weather conditions.

Michrosat modem The datalogger can access this infrastructure using a Michrosat 2403 dedicated modem connected to one of its 12V port. The modem is associated with a "phone" number and a SIM able to receive calls. Because of the high power consumption the modem is duty-cycled, being switched on only for a few time during the day. Moreover, the modem is not switched on if the battery voltage level of the datalogger is $\texttt{V}_{\texttt{BATTERY}} \leq \texttt{LOW}$ and not turned on until it's above $\texttt{V}_{\texttt{BATTERY}} \geq \texttt{HIGH}$ using a hysteresis control of the modem. [1]

The other endpoint On the other side of the satellite connection, there is a so-called reference station that makes calls to datalogger's modem number and establishes satellite connection between these two endpoints. Those calls are scheduled using the Loggernet application in order to match duty cycles of datalogger's modem.

1.3 Loggernet remote server

Adding and configuring devices Loggernet provides functions to add and configure a Campbell Scientific device to a network of devices. A device can be added specifying the interface used for the connection (e.g. serial port or radio medium) and its address within the network, then CRBASIC programs can be flashed using Loggernet. Moreover tables stored in device's memory can be selected for remote data fetching.

On the Internet stack If we look at the infrastructure at this point of the analysis we can see the application layer (Loggernet), data link layer (Iridium satellite communication) and physical layer (radio/satellite). The two remaining layers (transport and network) are a custom implementation by Campbell Scientific, called PakBus protocol family.

PakBus protocol [4] The PakBus protocol is similar to TCP/IP. PakBus provides the following services:

- Auto-discovery of the network topology
- Communication between datalogger endpoints (including Loggernet)

Every device can be assigned a `12 bit` address that identifies the datalogger in the PakBus network. Management software are identified by address `>=4000` (particularly Loggernet server has address `4094`).

Every packet has an header containing control information like `SenderAddr, ReceiverAddr and MessageType`, a message body containing data payload and a message trailer used for error checking.

Data fetching Once datalogger is configured into Loggernet, messages can be exchanged between the two endpoints. PakBus protocol's `MessageType` field contains information about the instruction requested by the sending host. Thus Loggernet can ask datalogger to send back its data table definition and the operator can mark one or more of them for data fetch. Now, another message from Loggernet can finally fetch data from remote datalogger.

1.4 FTP Remote Server

Once data are collected from the datalogger, it could be useful to have easier access to them using a common file transfer protocol such as FTP.

Redirecting data to FTP Server Loggernet provides a task scheduling tool called Task Master that can be configured to execute upload tasks to a specific FTP server when data is fetched. Once upload is completed, data table content can be accessed using an FTP client.

1.5 GSN Web Application

What is a GSN [2] A Global Sensor Network (GSN) is a web framework that provides an abstraction layer capable of retrieve data from nearly any kind of sensor/logger and present it in various forms such as chart or human readable tables.

Virtual Sensors From this point of view, every device can be abstracted to a virtual sensor that processes data source inputs and produces an output stream. A virtual sensor can be defined in a XML configuration file where following informations are specified:

- Output structure
- Set of data input streams containing data source informations

In [2] there is the configuration of La Mare virtual sensor, as shown in appendix B

The FTP Wrapper The system described before produces data into a FTP Server, so it's necessary to provide a wrapper that encapsulates data received into the GNS standard data model. Then application logic produces output stream as a row of an SQL table.

Layers	**AWS**	**Wireless Inn, Headend**
Application	CR1000 Program	Loggernet
Transport	PakBus Transport Protocol	PakBus Transport Protocol
Network	PakBus Network Protocol	PakBus Network Protocol
Data Link	Iridium	Iridium
Physical	MiChroSat	MiChroSat

Table 1: AWS to W.I. Headend infrastructure stack

Layers	**Wireless Inn. headend**	**FTP Server**
Application	FTP scheduled by Loggernet	FTP
Transport	TCP	TCP
Network	IP	IP

Table 2: W.I Headend FTP data transfer to remote infrastructure stack

1.6 System structure recap

Figure 1 shows the UML deployment diagram of the whole system.

The infrastructure stack involved in connection between the automatic weather station and the Wireless Innovation reference-station (headend) is shown in table 1.

Standard FTP protocol stack in data transfer between Wireless Innovation headend and FTP remote server is shown in table 2

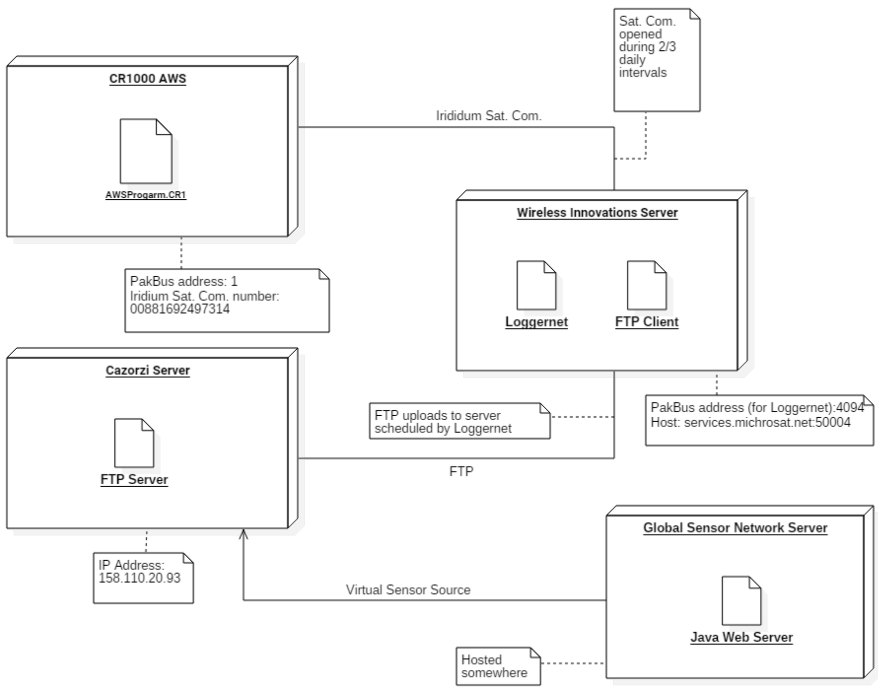

Figure 1: Deployment UML diagram

2 System deployment documentation

In this section more attention has been paid to configuration and deployment aspects of the system. Host addresses and ports are shown by the UML diagram in figure 1. Services' access credentials and information can be found in appendix C

System configuration can be split into following parts:

- Cable management
- Accessing remote host (provided by Wireless Innovation) via RDP
- Configuring Loggernet to communicate with the AWS
- Scheduling data fetch from AWS and FTP upload to FTP server
- Accessing FTP server
- Configuring and deploying the GSN server on Linux host.

2.1 Cable management for power supply

CR1000 In order to give power to CR1000 connect its power port to a 12V DC voltage source.

Michrosat modem Michrosat modem can be powered on both via 12V DC source and CR1000 voltage source ports. First case is useful when attempting signal quality test from pc without the datalogger. The second case is the common deployment scenario. In this case the modem can be connected directly to one of the 12V source ports of the datalogger (high power consumption) or to a switched 12V source port (software-optimized power consumption).

2.2 Cable management for data transfer

PC - CR1000 Connection Use a straight serial rs232 cable (DB9-DB9) to connect PC and CR1000. On the PC side, serial cable has to be connected to a `USB to serial` adapter. On the CR1000 side serial cable has to be connected to the port labeled `RS232 (Not isolated)`.

PC - Michrosat2403 Connection Use a straight serial rs232 cable to connect PC and Michrosat modem, using the same serial to usb adapter seen before on pc side.

CR1000 - Michrosat2403 Connection Use a NULL MODEM male to male serial RS232 cable to connect PC and Michrosat modem. On the CR1000 side, serial cable has to be connected to the port labeled `RS232 (Not isolated)`.

2.3 RDP access to remote Windows host

Connecting to host To access remote host running Loggernet user can use any RDP (remote desktop protocol).
Then the user must provide valid authentication credentials (user-name and password).

Using remote host Once connected, user can launch Loggernet application if not already started. To exit, user can both close RDP window or click on `Disconnect` in Windows start menu. (NOTE: Don't click on `Log-off`).

2.4 Configuring device into local Loggernet

Connection in local environment is accomplished by a serial RS232 serial connection.

Serial link requirements To connect device via serial link user has to install a driver for the serial-to-USB adapter, that can be downloaded from [6].

Adding device to Loggernet Once the device is connected, user can now open Loggernet and click on `Set-up` in `Main` tab. If it's the first time running Loggernet, it will open the `EZSetup Wizard`, otherwise user has to click on `Add` button located in the tool-bar.

1. In `Communication Set-up` section choose CR1000 and go to the next step.
2. Then choose connection type (`Direct` if serial or `Phone Modem` if Satellite)
3. Then follow instructions on program to find the correct COM port (and satellite SIM number if using satellite connection) to connect with
4. Skip `Datalogger settings` by pressing Next, if using standard parameters (potentially set baud rate at `9600`).
5. Check selected parameters in `Set-up summary`
6. Verify parameters in `Communication Test`
7. If test passes press `Finish`

Added device can now be seen by pressing `Set-up` in `Main` tab.

2.5 Configuring device into remote Loggernet

Connection from remote environment is accomplished by using the satellite connection infrastructure explained in section 1.3. Locally, connect CR1000 to modem as shown in section 2.2.

To add a device to remote Loggernet:

1. Open remote desktop connection as shown in section 2.3 and start Loggernet if stopped.
2. Click on `Set-up` in `Main` tab.
3. Click on `Add Root` on the tool-bar and select `ComPort`.
4. In the `Network map` below, click on the just added ComPort and on the right select in `ComPort Connection` `FabulaTech Serial Port Redirector`. Below set `Delay Hangup` at `500ms` and `Communication Delay` at `2s`.
5. In the `Network map` right-click on the ComPort and select `PhoneBase`.
6. Click on the PhoneBase added and set `Maximum baud rate` at `9600`, `Response Time` at `2s` and `Delay Hangup` at `500ms`.
7. Right-click on the PhoneBase and select `PhoneRemote`.
8. Click on the PhoneRemote added and add the SIM phone number of the CR1000 modem with a delay of `500ms`.
9. Right-click on PhoneRemote and select `Generic`.
10. Click on the Generic added and set `Baud rate` at `9600`, `Response time` at `4s`, `Maximum packet size` at `2048` and `Delay Hangup` at `2s 500ms`. In tab `Modem` set `Dial string` to `D10000`.
11. Right-click on the Generic and select `PackBusPort`
12. — to be continued

2.6 Scheduling operations from Loggernet

2.7 Manual data retrieve from FTP server

Connecting to FTP Using any FTP client, user must provide access credential to FTP host.

Files in FTP host Once connected, user can choose files to download:

- `CR1000_2_Status.dat` datalogger status table containing runtime info.
- `CR1000_2_Table1.dat` Table number 1 (see appendix A) data.
- `CR1000_2_Table2.dat` Table number 2 (see appendix A) data.

2.8 Deploying GNS on Linux host

Appendices

A Initial Source Code

Code listed below is from [5]

A.1 Data Table Definition

The following piece of code shows the definition of a data table. As underlined by `DataInterval` directive, the first data table is stored every 15 minutes, while the second is stored every 60 minutes.

```
DataTable(Table1,True,-1)
          DataInterval(0,15,Min,10)
          Average(1,RGup,FP2,False)
          Average(1,RGdown,FP2,False)
          Average(1,AirTC,FP2,False)
          Average(1,RH,FP2,False)
          WindVector (1,Vvento,DirVento,FP2,False,0,0,2)
          FieldNames("Vvento_S_WVT,Vvento_U_WVT,DirVento_DU_WVT
                        ,DirVento_SDU_WVT")
          Average(1,IRup,FP2,False)
          Average(1,IRdown,FP2,False)
          Average(1,IRupc,FP2,False)
          Average(1,IRdownc,FP2,False)
          Average(1,T107_1,FP2,False)
          Average(1,T107_2,FP2,False)
          Average(1,Thmp45,IEEE4,0)
          Average(1,URhmp45,IEEE4,0)
          Sample(1,Status.SW12Volts(1,1),Boolean)
EndTable

DataTable(Table2,True,-1)
          DataInterval(0,60,Min,10)
          Minimum(1,Batt_Volt,FP2,False,False)
          Sample(1,DT,FP2)
EndTable
```

A.2 Swithing modem on

The following code shows how the hysteresis control is applied to decide whether the modem can be swtiched on or not.

```
Dim lowBatTh As Float = 11.50
Dim highBatTh As Float = 13.20

Sub PowerOn
battPowerOnVoltage = Batt_Volt
 If (battPowerOnVoltage < lowBatTh) Then
  powerOnFlag = 0
  Else If (battPowerOnVoltage > highBatTh OR
          ((battPowerOnVoltage >= lowBatTh) AND powerOnFlag)) Then
   SW12(1)
   powerOnFlag = 1
 EndIf
EndSub
```

A.3 Retrieving data from sensors

The following code shows the instructions given to the datalogger in order to retrieve data from sensors. Every 60 seconds (specified by the instruction `Scan` the datalogger read measurements from sensors and store them in data table with the instruction `CallTable`

```
Scan(60,Sec,1,0)
     RealTime(rTime)

     VoltSe(Thmp45,1,mV2500,13,0,0,_50Hz,0.1,-40)
     VoltSe(URhmp45,1,mV2500,14,0,0,_50Hz,0.1,0)
     If URhmp45 >100 Then URhmp45=100

     Battery(Batt_Volt)

     'Pyranometer measurements Solar_kJ and RGup:
                    VoltDiff(RGup,1,mv25,1,True,0,_50Hz,1,0)
                    If RGup<0 Then RGup=0
                    Solar_kJ=RGup*7.46268
                    RGup=RGup*124.378

     'Pyranometer measurements Solar_k_2 and RGdown:
                    VoltDiff(RGdown,1,mv25,2,True,0,_50Hz,1,0)
                    If RGdown<0 Then RGdown=0
                    Solar_k_2=RGdown*7.46268
                    RGdown=RGdown*124.378
```

```
    'CS215 Temperature & Relative Humidity Sensor
            SDI12Recorder(AirTC,1,"0","M!",1,0)


    '05103 Wind Speed & Direction Sensor
            PulseCount(Vvento,1,1,1,1,0.098,0)
            BrHalf(DirVento,1,mV2500,5,1,1,2500,True,0,_50Hz,355,0)
            If DirVento>=360 Then DirVento=0

    'Every 60 minutes read Sonic Ranging Sensor
    If TimeIntoInterval(0,60,Min) Then
    'SR50 Sonic Ranging Sensor
            SDI12Recorder(DT,7,"0","M!",100.0,0)
            TCDT=DT*SQR((AirTC+273.15)/273.15)
            EndIf

    'Wiring Panel Temperature measurement TCR1000:
            PanelTemp(TCR1000,_50Hz)

    'Generic Differential Voltage measurements IRup:
            VoltDiff(IRup,1,mV25,4,True,0,_50Hz,86.881,0.0)
            'Generic Differential Voltage measurements IRdown:
            VoltDiff(IRdown,1,mV25,5,True,0,_50Hz,118.2,0.0)

    '/****************************/
    .......
    '/****************************/

    'Call Data Tables and Store Data
            CallTable(Table1)
            CallTable(Table2)
NextScan
```

B La Mare virtual sensor XML configuration


```
<virtual-sensor name="Lamare_GetFtp" priority="10">
  <processing-class>
    <class-name>gsn.vsensor.BridgeVirtualSensor</class-name>
    <unique-timestamps>true</unique-timestamps>
    <init-params />
    <output-structure>
      <field name="CR1000_2_Table1" type="binary"/>
    </output-structure>
  </processing-class>
  <description>This VS shows files gets from ftp URL</description>
  <streams>
    <stream name="inputFtp">
      <source alias="source" sampling-rate="1" storage-size="1">
        <address wrapper="ftp">
          <predicate key="url">192.168.1.10</predicate>
          <predicate key="username">"username"</predicate>
          <predicate key="password">"password"</predicate>
          <predicate key="path">CR1000_2_Table1.dat</predicate>
          <predicate key="rate">3d</predicate>
        </address>
        <query>
          SELECT source1.TIMED, CR1000_2_TABLE1 FROM source
        </query>
      </source>
    </stream>
  </streams>
</virtual-sensor>
```

C Service access credentials

Service	Host	Username
WI RDP Server	services.michrosat.net:50004	CUSTWILTD/CIRGEO
FTP Server	158.110.20.93	AWS

Table 3: Hosts and user-names of services

References

[1] S. Abbate, M. Avvenuti, L. Carturan, and D. Cesarini. “Deploying a Communicating Automatic Weather Station on an Alpine Glacier”. In: (2013), pp. 22,23.

[2] A. Celli. “Realtime Processing and Presentation of Environmental Data on Global Sensor Network web framework to study climate change on Glaciers”. In: (2013), pp. 10,11,12,13.

[3] Campbell Scientific Ltd. “CR1000 Measurement and Control System - Operator’s Manual”. In: (2006).

[4] Campbell Scientific Ltd. “PakBus Networking Guide”. In: (2008).

[5] S. Pallica. “Adaptive sensing and transmission on an Automatic Weather Station: design of a practical system deployed on a glacier”. In: (2013), pp. 84–103.

[6] Inc. Prolific Technology. *Products: USB to Serial.* URL: `http://www.prolific.com.tw/US/ShowProduct.aspx?pcid=41&showlevel=0041-0041`.