UNIVERSITÀ DI PISA

Scuola di Ingegneria
Dipartimento di Ingegneria dell'Informazione
Corso di laurea triennale in Ingegneria Informatica

TESI DI LAUREA

The automatic weather station of La Mare glacier

RELATORI
Prof. Marco AVVENUTI

CANDIDATO
Federico ROSSI

ANNO ACCADEMICO 2016/2017

Abstract

Nowadays it's very important to monitor sensitive 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 components.

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 analyze the system in order to obtain more knowledge about it to build a documentation for future maintenance and upgrades of the system and re-deploy the system connectivity on the glacier.

Contents

Figure 1: CR1000 Datalogger

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 [2] Simplifying, the CR1000 Datalogger can read data from sensors connected to both its digital and analogical 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
- Therm-Hygrometer: Air temperature and humidity
- Pirgeometer: IN/OUT long-wavelength radiation
- Thermistors: Temperature from Therm-Hygrometer 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; indeed 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 powered by 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 [6] 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 [8] 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 information are specified:

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

In [8] 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 GSN standard data model. Then application logic produces output stream as a row of an SQL table.

1.6 System structure recap

Figure 2 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 (head-end) is shown in table 1.

Layers	**AWS**	**Wireless Inn, Head-end**
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. Head-end infrastructure stack

Layers	**Wireless Inn. head-end**	**FTP Server**
Application	FTP scheduled by Loggernet	FTP
Transport	TCP	TCP
Network	IP	IP

Table 2: W.I Head-end FTP data transfer to remote infrastructure stack

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

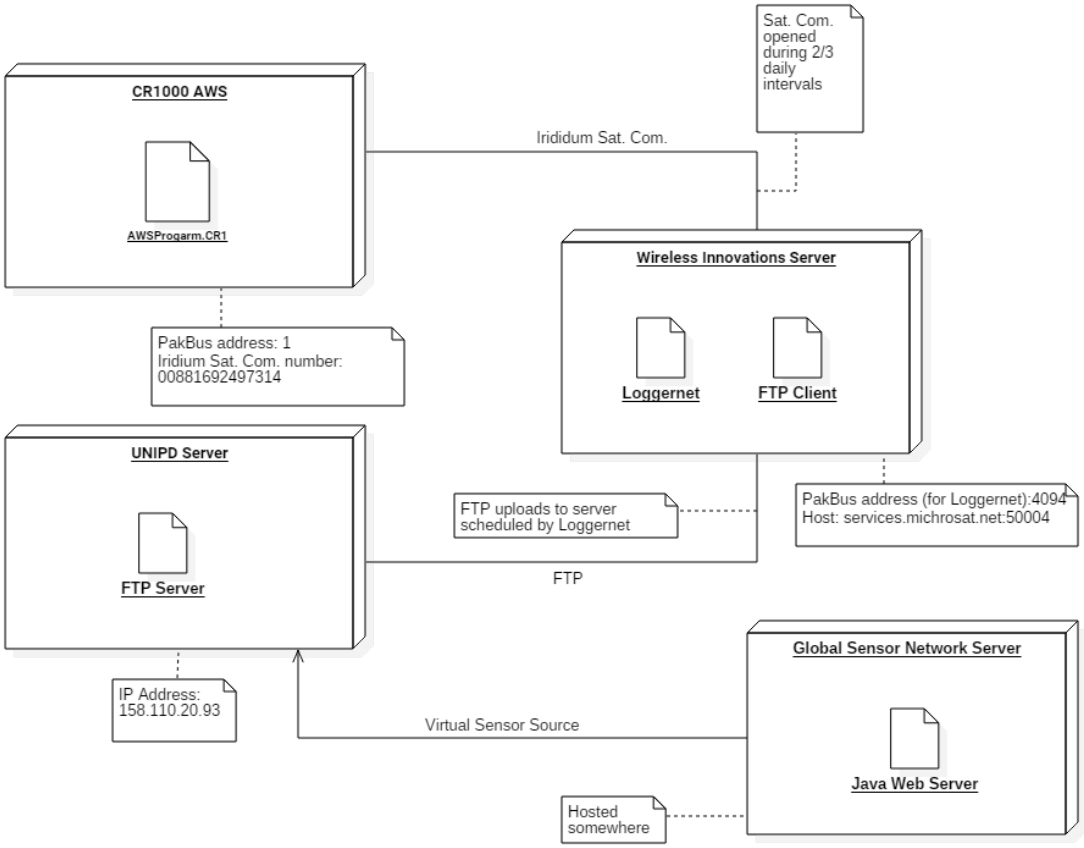

Figure 2: Deployment UML diagram

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 2. 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 (12V+GND in figure 1) (high power consumption) or to a switched 12V (SW12+GND in figure 1 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)` in figure 1.

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

CR1000 - Michrosat 2403 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)` in figure 1. In order to obtain a male to male configuration, use a pair of gender changer on a female to female null modem rs232 serial cable.

2.3 Upgrading datalogger firmware

Download latest firmware Latest firmware for the datalogger can be found and downloaded at [3]. It is an auto-extractable EXE archive that extracts the firmware (.obj file) into the following directory:

`C:\CampbellSci\Lib\OperatingSystems\`

Device configuration utility In order to install the firmware on the datalogger, the software Device Configuration Utility has to be installed. It can be found at [4].

Once both firmware and software are ready:

1. Open Device Configuration Utility.
2. Connect CR1000 to PC as shown in 2.2.
3. On the left, select CR1000 and below the communication port where USB to serial converter is attached.
4. Click connect and wait for the connection to establish.
5. Once connected, select `Backup` from the window menu and click on `Back Up` datalogger.
6. In the new dialog, choose backup location and data from datalogger to backup. Wait for the process to be finished.
7. Once backup is completed, on the right, click on tab `Send Os`.
8. Follow instruction on this tab to perform datalogger upgrade.

2.4 Modem configuration

Michrosat 2403 modem has to be configured in order to answer automatically incoming calls from remote Loggernet:

1. Connect modem to power source (battery or CR1000 12V source) and PC as shown in 2.2.
2. Once connected, check Windows' Device Manager to discover COM port where device is attached, it should display `Prolific USB to Serial adaptor`.
3. Open a serial terminal (e.g. Putty) and connect to COM port discovered above.
4. Check terminal functionality by sending `AT` command. It should answer `OK`
5. Set auto response after the first ring with `ATS0=1`, it should answer `OK`
6. Write configuration in flash memory with `AT&W0` and select it as start-up configuration with `AT&Y0`
7. OPTIONAL: if the antenna is connected, check the signal quality with `AT+CSQ`. It should response with a number between 0 (no signal) and 5 (excellent signal)

2.5 Configuring device into local Loggernet (or PC200W)

Connection in local environment is accomplished by a serial RS232 serial connection. While PC200W is distributed as free software, Loggernet is offered both in a paid version and a 30-day trial version. PC200W can be downloaded from [7]. Loggernet trial version can be downloaded from [5]

Serial link requirements To connect device via serial link user has to install a driver for the serial-to-USB adaptor, that can be downloaded from [11].

Adding device 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 (or PC200W), it will open the `EZSetup Wizard`, otherwise user has to click on `Add` button located in the tool-bar (if using Loggernet, make sure Set-up is running in simplified mode by checking it in the title bar. If not, click on the button `EZ View` on the tool-bar).

1. In `Communication Set-up` section choose CR1000 and go to the next step.
2. Then choose connection type `Direct`
3. Select COM port to connect with (check Windows' Device Manager to discover COM port, it should display `Prolific USB to Serial adaptor`).
4. Set `Baud rate` at `9600` and `PakBus Address` at `1`.
5. Skip the following `Datalogger settings` by pressing Next if not using any type of encryption.
6. Check chosen parameters in `Set-up summary`
7. Verify parameters in `Communication Test` by clicking `Next`
8. If test passes, in the following tab set the datalogger clock by clicking `Check Datalogger Clock`
9. Select and send a program to the datalogger by clicking on `Select and Send program`
10. *Only for Loggernet:* click on `Get table definition` to retrieve table structures from the datalogger as defined in the program
11. *Only for Loggernet:* enable `Scheduled collection` and set the time and interval for automatic data collection.
12. Click `Finish` to close the wizard.

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

2.6 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`).

2.7 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 2.6 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 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 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 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. Click on the PackBusPort and set `Maximum Time On-Line` at `10m`, `Baud Rate` at `9600`, `Beacon Interval` at `1m`, `Extra response time` at `4s` and `Delay Hangup` at `2s 500ms`.
13. Right-click on the PackBusPort and select `CR1000`
14. Click on the CR1000 and set `PakBus Address` at `1`, `Packet Size` at `1000` and `Delay Hangup` at `2s 500ms`.
15. On `Data Files` tab click on `Get Table Definitions`. Loggernet will contact the datalogger to retrieve Tables declared in the program.
16. Once completed, on the left some tables will appear. Click on `Status` and check `Include for scheduled collection`. Set `File Output Option` to `Append to end of file`. Select `Data Logged Since Last Collection` and check `Collect All On First Collection`
17. Repeat step 16 for tables Table1 and Table2.
18. Once finished, click on `Apply` button on the bottom of `Setup` window.

2.8 Scheduling data collection operations from Loggernet

Scheduling data collections can be accomplished by synchronizing modem-on windows on the datalogger and schedules on remote Loggernet. Note that datalogger has clock set on GMT+1, while Loggernet is set on GMT.

Set modem-on windows on datalogger In order to set modem-on windows on datalogger, the program running on CR1000 has to be edited and re-compiled on the datalogger. See appendix A to find more information on the code to be edited.

Set Loggernet's schedules To configure Loggernet's schedules:

1. Open remote desktop connection as shown in section 2.6 and start Loggernet if stopped.
2. Click on `Set-up` in `Main` tab.
3. Select `CR1000` datalogger from the panel on the left.
4. On tab `Schedule` check `Scheduled Collection Enabled`
5. Set `Time` in order to match one of the modem-on windows just configured on datalogger. NOTE: set the time 5 or more minutes after modem-on windows start to permit modem to register on the network.
6. Set `Collection interval` to any value in days. (Typically use 3 days interval between collection)
7. Set `Primary retry interval` at `10m` and `Number of primary retries` at `1`
8. Check `Stay on collect schedule`.
9. On the drop-down below select `Automatically reset changed tables` and check `Enable automatic hole collection`
10. Once finished, click on `Apply` at the bottom of `Setup` window.

Schedule FTP task Once data collection schedule is set, create a task that forwards data to a remote FTP server.

1. Open remote desktop connection as shown in section 2.6 and start Loggernet if stopped.
2. Click on `Task master` on `Main` tab.
3. Click on `CR1000` under `Tasks`, then click on the button `Add after` below.
4. On the right, set `Station Event Type` to `After file closed`.
5. Click on `Configure task...` below, a new window will be opened.
6. On tab `FTP Settings` check `Enable FTP`. Set following parameters according to appendix C (service FTP SERVER). Browse `Remote folder` with button on the right and select preferred folder where data files will be uploaded. Confirm with `OK`
7. Once finished, click on `Apply` at the bottom of `Task master` windows.

2.9 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 run-time info.
- `CR1000_2_Table1.dat` Table number 1 (see appendix A) data.
- `CR1000_2_Table2.dat` Table number 2 (see appendix A) data.

2.10 Deploying GSN on Linux-Windows host

Prerequisites GSN module has the following dependencies:

- Java: JDK and JRE.
- Apache Ant for source compilation.
- MySQL Server.

Source code Source code provided is a custom version of original GSN source code available at [9]

Preliminary Configuration Open a terminal on the root folder and execute the `init.sh` script. It will create default user and database used by the application. The default configuration can be found in `gsn_dev/conf/gsn_conf.xml`

```
1 <storage user="gsn" password="gsnpassword" driver="com.mysql.jdbc.
    Driver" url="jdbc:mysql://localhost/gsn" />
```

Schedule FTP fetch Open a terminal on the root folder and execute the command `crontab -u your_username cronfile`. It will schedule the FTP data fetch from LaMare FTP server. At this point everything is configured.

Running GSN Open a command prompt on the project root `gsn-dev` and execute `ant gsn` to compile and execute the application. Test the application in a web browser at the link `http://127.0.0.1:22001`. You should see the main page showing pre-configured virtual sensors.

3 Software analysis

3.1 CRBASIC program structure

Data-logger's program can be split in three parts:

- Main scan (or main loop)
- Secondary scans (also called slow sequence scans)
- Data table declarations (plus functions and global variables)

Main Scan The main scan is located at the first `Scan` instruction after the `BeginProg` directive. It contains data measurement and processing along with `CallTable <TableName>` instructions that saves data into the table specified.

Secondary scans Secondary scans,also called slow sequence scans, are ordinary loops marked with the keyword `SlowSequence`. In 3.2 more information about scan and task scheduling are provided.

Data table Data tables are declared with the `DataTable` directive. It takes table name and trigger variable (in the considered case trigger is always set to true) as arguments. The size can be specified, but specifying an auto-sizing table (size=0) has to be preferred.

Table columns are enclosed between `DataTable` and `EndTable` directives, together with the "DataInterval" directive, that specifies on which time basis the table has to be saved for output and the `FieldNames` directive, that specifies column names.

Columns are declared by specifying the statistic function to apply to data from `CallTable` directives (e.g Average,Sample,Minimum) and the variable from where desired value will be fetched during `CallTable`

A particular directive, called `WindVector`, allows to store data from pulse sensors like the wind vector used in this scenario.

3.2 Tasks and scheduling

Tasks In a CRBASIC program, main scan, secondary scans and measurement processes are identified as Task. Every task has its own priority and it is arranged in one of three scheduling queues ordered by priority:

1. Measurement tasks: have the maximum priority because they are considered like real-time processes.
2. SDM tasks: device management tasks issued by the CS OS.
3. Processing Tasks: all other tasks.

CPU execution mode Data-logger's program can compiled in Sequential Mode or Pipeline (the choice is done by the compiler itself or can be forced using a directive in the program). Within the first mode instructions are executed in the order they are written in the program. This means there is no reorder of instruction in the CPU, as it happens in Pipeline mode.

Main and secondary scans Among all the scans, the one identified as "Main Scan" has the higher priority. All the other scans can execute once the main scan has finished its internal tasks, or when the main scan is not active. This means that in Pipeline mode,a single scan can be split over multiple executions of the main scan.

4 Conclusion

At the end of the work, the system has regained full functionality that includes remote data retrieval via satellite connection, data forwarding to the UniPd's FTP server and presentation in human readable form of data in a web server powered by Global Sensor Network.

Future work has in view the re-installation of the AWS on the glacier.

Appendices

A Code snippets

Code listed below is from [10]

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.

```
1  DataTable(Table1,True,-1)
2          DataInterval(0,15,Min,10)
3          Average(1,RGup,FP2,False)
4          Average(1,RGdown,FP2,False)
5          Average(1,AirTC,FP2,False)
6          Average(1,RH,FP2,False)
7          WindVector (1,Vvento,DirVento,FP2,False,0,0,2)
8          FieldNames("Vvento_S_WVT,Vvento_U_WVT,DirVento_DU_WVT
9                      ,DirVento_SDU_WVT")
10         Average(1,IRup,FP2,False)
11         Average(1,IRdown,FP2,False)
12         Average(1,IRupc,FP2,False)
13         Average(1,IRdownc,FP2,False)
14         Average(1,T107_1,FP2,False)
15         Average(1,T107_2,FP2,False)
16         Average(1,Thmp45,IEEE4,0)
17         Average(1,URhmp45,IEEE4,0)
18         Sample(1,Status.SW12Volts(1,1),Boolean)
19 EndTable
20
21 DataTable(Table2,True,-1)
22         DataInterval(0,60,Min,10)
23         Minimum(1,Batt_Volt,FP2,False,False)
24         Sample(1,DT,FP2)
25 EndTable
```

A.2 Switching modem on

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

```
1  Dim lowBatTh As Float = 11.50
2  Dim highBatTh As Float = 13.20
3
4  Sub PowerOn
5  battPowerOnVoltage = Batt_Volt
6   If (battPowerOnVoltage < lowBatTh) Then
7    powerOnFlag = 0
8    Else If (battPowerOnVoltage > highBatTh OR
9            ((battPowerOnVoltage >= lowBatTh) AND powerOnFlag)) Then
10    SW12(1)
11    powerOnFlag = 1
12  EndIf
13 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`

```
1  Scan(60,Sec,1,0)
2       RealTime(rTime)
3
4       VoltSe(Thmp45,1,mV2500,13,0,0,_50Hz,0.1,-40)
5       VoltSe(URhmp45,1,mV2500,14,0,0,_50Hz,0.1,0)
6       If URhmp45 >100 Then URhmp45=100
7
8       Battery(Batt_Volt)
9
10      'Pyranometer measurements Solar_kJ and RGup:
11                  VoltDiff(RGup,1,mv25,1,True,0,_50Hz,1,0)
12                  If RGup<0 Then RGup=0
13                  Solar_kJ=RGup*7.46268
14                  RGup=RGup*124.378
15
16      'Pyranometer measurements Solar_k_2 and RGdown:
17                  VoltDiff(RGdown,1,mv25,2,True,0,_50Hz,1,0)
18                  If RGdown<0 Then RGdown=0
19                  Solar_k_2=RGdown*7.46268
20                  RGdown=RGdown*124.378
21
22      'CS215 Temperature & Relative Humidity Sensor
23                  SDI12Recorder(AirTC,1,"0","M!",1,0)
24
25
```

```
26      '05103 Wind Speed & Direction Sensor
27                  PulseCount(Vvento,1,1,1,1,0.098,0)
28                  BrHalf(DirVento,1,mV2500,5,1,1,2500,True,0,_50Hz,355,0)
29                  If DirVento>=360 Then DirVento=0
30
31      'Every 60 minutes read Sonic Ranging Sensor
32      If TimeIntoInterval(0,60,Min) Then
33      'SR50 Sonic Ranging Sensor
34                  SDI12Recorder(DT,7,"0","M!",100.0,0)
35                  TCDT=DT*SQR((AirTC+273.15)/273.15)
36                  EndIf
37
38      'Wiring Panel Temperature measurement TCR1000:
39                  PanelTemp(TCR1000,_50Hz)
40
41      'Generic Differential Voltage measurements IRup:
42                  VoltDiff(IRup,1,mV25,4,True,0,_50Hz,86.881,0.0)
43                  'Generic Differential Voltage measurements IRdown:
44                  VoltDiff(IRdown,1,mV25,5,True,0,_50Hz,118.2,0.0)
45
46      '/*****************************/
47      .......
48      '/*****************************/
49
50      'Call Data Tables and Store Data
51                  CallTable(Table1)
52                  CallTable(Table2)
53  NextScan
```

A.4 Modem-on windows

Following code shows the part of the program that has to edited in order to modify modem-on windows in datalogger.

```
1  ===> Number of daily modem-on windows
2  Const NumWindows = 3
3  /......./
4  'Set true to days you want the modem to be switched on
5  Monday = true
6  Tuesday = true
7  Wednesday = true
8  Thursday = true
9  Friday = true
10 Saturday = true
11 Sunday = true
12
13 ===> First daily window
14 WHourStart(1) = 10 'Ora di inizio della Prima Finestra (0-23)
15 WMinStart(1) = 31 'Minuto di inizio della Prima Finestra (0-59)
```

```
16 WNumMin(1) = 30 'Durata in minuti della Prima Finestra
17
18 ==========> Second daily window
19 WHourStart(2) = 13 'Ora di inizio della Seconda Finestra (0-23)
20 WMinStart(2) = 31 'Minuto di inizio della Seconda Finestra (0-59)
21 WNumMin(2) = 30 'Durata in minuti della Seconda Finestra
22
23 ==========> Third daily window
24 WHourStart(3) = 20 'Ora di inizio della Terza Finestra (0-23)
25 WMinStart(3) = 50 'Minuto di inizio della Terza Finestra (0-59)
26 WNumMin(3) = 0 'Durata in minuti della Terza Finestra
```

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] Ltd. Campbell Scientific. "CR1000 Measurement and Control System - Operator's Manual". In: (2006).

[3] Ltd. Campbell Scientific. *CR1000 OS*. URL: https://www.campbellsci.com/downloads?sb=CR1000+OS&c=9999.

[4] Ltd. Campbell Scientific. *Device Configuration Utility*. URL: https://www.campbellsci.com/downloads?c=9999&d=83.

[5] Ltd. Campbell Scientific. *Loggernet Trial*. URL: https://www.campbellsci.com/downloads?c=9999&d=169.

[6] Ltd. Campbell Scientific. "PakBus Networking Guide". In: (2008).

[7] Ltd. Campbell Scientific. *PC200W*. URL: https://www.campbellsci.com/downloads?c=9999&d=87.

[8] 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.

[9] École polytechnique fédérale de Lausanne. *GSN*. URL: https://github.com/LSIR/gsn.

[10] 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.

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