This document is subject to change. The latest version may be found at https://github.com/ashawnbandy/cecs491/tree/master/docs

10/9/2012

CSULB Marine Biology Department Software Project

*Requirements Analysis (Preliminary rev 3)*

Contents

[Introduction 2](#_Toc336946503)

[Purpose of the System 2](#_Toc336946504)

[Objectives 2](#_Toc336946505)

[Definitions 2](#_Toc336946506)

[Functional Requirements 2](#_Toc336946507)

[Non-Functional Requirements 3](#_Toc336946508)

[Usability 3](#_Toc336946509)

[Reliability 3](#_Toc336946510)

[Safety 3](#_Toc336946511)

[Security 3](#_Toc336946512)

[Performance 3](#_Toc336946513)

[Supportability 4](#_Toc336946514)

[Implementation – To Be Determined 4](#_Toc336946515)

[Interface 4](#_Toc336946516)

[Packaging 4](#_Toc336946517)

[Statistical Data – To Be Determined 4](#_Toc336946518)

[Legal – To Be Determined 4](#_Toc336946519)

[Use Cases 5](#_Toc336946520)

[Change to real-time mode 5](#_Toc336946521)

[Connect to server 5](#_Toc336946522)

[Setup email alert(s) 5](#_Toc336946523)

[Start/stop recording data 5](#_Toc336946524)

[Start/stop recording data 6](#_Toc336946525)

[Open and use a text console for direct interaction with receiver hardware 6](#_Toc336946526)

[Add and configure new receiver 7](#_Toc336946527)

[Access/query/save recorded data 7](#_Toc336946528)

[Begin Data Streaming 7](#_Toc336946529)

[Appendix 1 – System Overview Diagram (updated 9/25/2012) 9](#_Toc336946530)

[Appendix 2 – GUI Mock-ups (Added 10/02/2012) 10](#_Toc336946531)

# Introduction

## Purpose of the System

The CSULB Marine Biology department collects data from marine life that has been tagged with an acoustic transmitter. Data will be collected by a receiver located off of Manhattan Beach Pier (MBP), which will record the ID number of tags, associated sensor data, date, and time of detection. The receiver in turn will be connected to a computer through a serial port connection. The purpose of the system will be to interface with this computer remotely in order to control the receiver and receive data from it.

## Objectives

1. Ability to connect remotely to the computer managing the receiver
2. Ability to control the receiver remotely through that connection.
3. Ability to stream real-time data from the receiver.
4. Send email alerts when a detection meeting user-defined criteria is observed.
5. Archive recorded data and recording metadata.
6. Extensibility with an eye toward Phase II clients.

## Definitions

1. “The system” will refer to the software being created by this project, and not the firmware on the receiver equipment.
2. Software connecting to the remote sites (e.g. Manhattan Beach Pier) will be referred to as the “server” or “backend”.
3. Software running locally by an end-user will be referred to as the “client” or “front-end”.
4. In general, a collection of server and clients will be referred to as the “application network” with each site as “nodes”.
5. “Phase I” refers generically to the software previous defined.
6. A “user” has limited access to system, with primary interest the reading and exporting of approved data.
7. An “administrator” has access to all user related functions in addition to all functions with restricted user-access.
8. Real-Time-Mode (RTM) refers to the presentation of data as it comes into the server from the receiver and may or not utilize Real Time Mode 0 on the receiver hardware.

# Functional Requirements

1. Connect to receivers located remotely (e.g. MBP)
2. Start and stop recording data from receiver to server for later retrieval
3. Provide connection to receiver through console
4. Parse and aggregate data sent from receiver
5. Access and query recorded data
6. Stream status information from acoustic receiver using “Real Time Mode”
7. Email alerts sent out when the system detects user defined parameter
8. Transfer “Real Time” data to a remote SQL database.

# Non-Functional Requirements

## Usability

* The application will minimize network configuration.
* Additional receiver nodes with up to *N* receivers will also require minimal configuration by a remote operator.
* The application will allow more than one client to connect to the server.
* The front-end user interface will follow familiar design practices.

## Reliability

* The server should be continuously available. To this end, the system should detect critical faults and reset without end-user administration.
* Non-critical faults will be either logged or reported to a connected user.

## Safety

There are no known safety requirements.

## Security

* Access to the server will be limited to authorized users through the use of configurable access control lists.
* Data identified as sensitive will be encrypted when transmitted over open networks (e.g. the internet, the CSULB network, etc.)

## Performance

* Commands to receivers and their effects should be sent and received in near-real-time.
* Data from the receiver may be buffered by the server to extend storage capacity and facilitate lower latency data transfers.

## Supportability

* Sufficient documentation will be provided to the customer to allow for future bug fixes by a third-party.
* The design will be modular and make use of simple declarative language control files (i.e. json) to allow for future changes or additions to the system.

## 

## Implementation

* The software will be written in C# for deployment on Microsoft Windows machines.
* Client-Server Communication will be implemented using Simple Object Access Protocol (SOAP) which is a platform/OS/language independent framework for exchanging complex objects over the network.
* Network communication between remote sites and the server will be handled by serial-over-ethernet hardware device.

## Interface

* The system will interface with the firmware (current at time of implementation) on each receiver.
* The system will generate data consistent with existing output formats (e.g. CSV, SQL).

## Packaging

* + Server software will be installed at a remote site.
  + All software will also be packaged in a manner that facilitates additional installations.

# Use Cases



|  |  |
| --- | --- |
| M | |
| Name | Connect to server |
| Actor(s) | Administrator, user |
| Pre-conditions | Available network connection to the server. |
| Flow-of-Control | 1. The user specifies a server to connect to and credentials to connect with. 2. The client connects to the server and is authenticated, or denied. |
| Post-conditions | The user is connected to the server, or an error is returned. |

|  |  |
| --- | --- |
| Y | |
| Name | Setup email alert(s) |
| Actor(s) | Administrator |
| Pre-conditions | 1. Network connection to server. 2. Administrator has logged with valid credentials |
| Flow-of-Control | 1. The user selects Settings 🡪 Email Alert from menu. 2. The user enters an email address on the text box and clicks add button to enter the email on the email list. 3. The user setup filter for the email alert(s) based on location/receiver/specific tag/time etc. 4. The user clicks save to save the email alert setup. 5. The user clicks red X button on the right side upper corner to close the email setting window. |
| Post-conditions | The server machine sends out email alert(s) to the email addresses on the email list when the trigger condition occurs. |

|  |  |
| --- | --- |
| Y | |
| Name | Start/stop recording data |
| Actor(s) | Administrator |
| Pre-conditions | 1. Network connection to server. 2. Administrator has logged with valid credentials |
| Flow-of-Control | 1. The user chooses a location and a receiver to look the data on the Real-Time-Mode window. 2. The user click start button to start recording data. 3. The user click stop button to stop recording data. 4. The user specifies a location/folder to save the data. 5. The user specifies a name for the data file. 6. The user click save button to save the data on the computer. 7. The user click X button on the right side upper corner to close the window thereby ending Real-Time-Mode. |
| Post-conditions | 1. The user has the recorded data on his/her local machine. |

|  |  |
| --- | --- |
| Y | |
| Name | Start/stop recording data |
| Actor(s) | Administrator |
| Pre-conditions | 1. Network connection to a new receiver. 2. Administrator has logged with valid credentials |
| Flow-of-Control | 1. The user chooses a location and a receiver to look the data on the Real-Time-Mode window. 2. The user click start button to start recording data. 3. The user click stop button to stop recording data. 4. The user specifies a location/folder to save the data. 5. The user specifies a name for the data file. 6. The user click save button to save the data on the computer. |
| Post-conditions | The user has the recorded data on his/her local machine as a text file. |

|  |  |
| --- | --- |
| S | |
| Name | Open and use a text console for direct interaction with receiver hardware |
| Actor(s) | Administrator |
| Pre-conditions | 1. Network connection to server with a receiver in serial-ready mode. 2. Administrator has logged with valid credentials. |
| Flow-of-Control | 1. Administrator selects Connect to Receiver from menu. 2. The Administrator selection receiver from list and selects gesture to begin console session. 3. A text console with the current status of the receiver is displayed before the prompt. |
| Post-conditions | User has direct access to receiver hardware through the text console. |

|  |  |
| --- | --- |
| S | |
| Name | Add and configure new receiver |
| Actor(s) | Administrator |
| Pre-conditions | 1. Network connection to a new receiver. 2. Administrator has logged with valid credentials. |
| Flow-of-Control | Associate Administrator defined data with receiver hardware. |
| Post-conditions | Receiver is added to pool of active receivers and will be polled during the next cycle. |

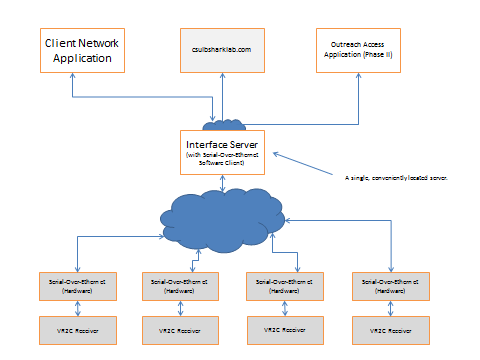
|  |  |
| --- | --- |
| D | |
| Name | Access/query recorded data |
| Actor(s) | Administrator, User |
| Pre-conditions | 1. Network connection to server 2. Administrator or User has logged in with valid credentials. |
| Flow-of-Control | 1. User enters all required fields and selects all necessary checkboxes. 2. User selects “search” button. 3. System returns query results, displays it in GUI |
| Post-conditions | Query results sent to the GUI. |

|  |  |
| --- | --- |
| D | |
| Name | Save recorded data |
| Actor(s) | Administrator, User |
| Pre-conditions | 1. Network connection to server 2. Administrator or User has logged in with valid credentials. |
| Flow-of-Control | 1. User selects file 🡪 Save/Save as from menu 2. User selects directory to save data on local computer. |
| Post-conditions | If user chooses to save their queried data, data is saved into user directory. |

|  |  |
| --- | --- |
| D | |
| Name | Add User |
| Actor(s) | Administrator |
| Pre-conditions | 1. Network connection to server 2. Administrator has logged in with valid credentials. |
| Flow-of-Control | 1. User selects file 🡪 Add User from menu. 2. User adds new username and password and optionally selects to add new user to administrator group. |
| Post-conditions | New user may log into system with newly created credentials |

|  |  |
| --- | --- |
| D | |
| Name | Remove User |
| Actor(s) | Administrator |
| Pre-conditions | 1. Network connection to server 2. Administrator has logged in with valid credentials. |
| Flow-of-Control | 1. User selects file 🡪Remove User from menu. 2. In list of users, selects user from whom to remove credentials. 3. Confirms action on pop-up. |
| Post-conditions | Removed user is no longer able to validate on system. |

# Appendix 1 – System Overview Diagrams (updated 9/25/2012)



# 

Main Service Loop

Responds to changes in COM port status.

**VR2C / serial-over-ethernet/COM port**

**VR2C / serial-over-ethernet/COM port**

Encoder

Note: Objects between common dashed lines represent serial executions.

Front-End Client

SQL Databases (e.g. csulbsharklab.com)

Parse/Decode Module

Client Communication Module (SOAP)

Persistence Module

Email Notification Module

Event Queue/Dispatcher

Distribute intra-server messages.

Receiver Objects

Sets up and receives data asyncrhonously from VR2C receivers

## Appendix 2 – Data Flow Diagrams (Added 10/08/2012)

# “Data Browser” Motif

Serial Port Stream

Advantages:

* Incoming data available in database in near-real-time.
* Less labor intensive than the “Gatekeeper”

Disadvantages:

* Real-world “noise” leaks into canonical data store.
* If editing records is enabled, the potential for catastrophic data loss from misuse or mistake exists.

Questions:

* How sensitive to unexpected (“bad”) data in the database will the end-users be?
* Should records in the database ever be edited?
* Could a reasonable set of filters in our “Persistence Module” cut down unexpected data sufficiently?

“Packaged”

VR2C Format

Intra-server messaging

SOAP

Client Com Module

Front-End

SQL

C# Object

Receiver Objects

Database

Parser Decoder

Persistence Module

# “Gatekeeper” Motif

Questions:

* Given the expected rate of transmitter/status events flowing through the server, is there sufficient man-power to reasonably verify/edit each record?
* Is the loss of “Real Time” data significant?

VR2C Format

# Appendix 3 – GUI Mock-ups (Added 10/02/2012)

C# Object (Intra-server format)

Note: The previous diagram shows the path that all data (whether historical or real-time) takes before being displayed on the Front-End GUI. This diagram does not show previously approved data nor the path it would take to the Front-End.

Advantages:

* Tighter control over quality of data being stored.

Disadvantages:

* More manual intensive than the “Data Browser”
* Increased complexity in the front-end.
* Loss of “near-real-time” data flowing into the csulbsharklab.com database.

Pending Data 🡪

🡨 Edited and Approved Data

SQL

Database

Persistence Module

Client Com Module

Front-End

VR2C Format

Receiver Objects

Parser Decoder