Crab Tracker - Software Requirements Specification

Margot Maxwell, Elizabeth Schoen, Noah Strong, Chloe Yugawa

November 23rd, 2017

Version 1.1

Revision History

Version	Date	Major Changes
1.0	2017-11-22	Initial Version
1.1	2017-11-23	Additions; reformatting

Introduction

Purpose

The “Crab Tracker” product described in this document is a cost-effective solution for tagging and tracking crabs.

Intended Audience and Reading Suggestions

This document is intended to be read by the software and hardware development teams, our client, and other concerned parties such as advisors and instructors.

Project Scope

This product is multi-component system designed to allow for a user to locate crabs in an aquatic environment. The crabs will transmit unique signals, and the software will determine which crab is transmitting that signal and provide relative location information for that crab to help the user find this crab. See the project’s Vision and Scope document for more details.

References (Related Works)

- *Gualtiere et. al.* Active Tracking of Maja Squinado in the Mediterranean Sea with Wireless Acoustic Sensors: Method, Results and Prospectives (2013)
- *Roegner and Fields.* Mouth of the Columbia River: Beneficial Sediment Deposition Project (2014)
- *Toshifumi et. al.* Movement Patterns and Residency of the Critically Endangered Horseshoe Crab Tachypleus tridentatus in a Semi-Enclosed Bay Determined Using Acoustic Telemetry (2016)
- *Bell et. al.* Behavioral responses of free-ranging blue crabs to episodic hypoxia. I. Movement (2003)
- *Wolcott and Hines.* Ultrasonic Telemetry Of Small-scale Movements And Microhabitat Selection By Molting Blue Crabs (Call/Nectes Sap/Dus) (1990)

Overall Description

Product Perspective

The goal of this product is to fill a market need to provide affordable (at the grad student level) technology (hardware and software) that allows researchers to keep track of the locations and movements of marine animals.

User Classes and Characteristics

This product is intended for biology researchers. As only one user is expected for a single installation of the product, there is no need for further distinction between user classes. That is, the researcher should have access to all real-time data gathered by the system, and should have the ability to adjust configuration parameters as needed for given scenarios.

Operating Environment

Hardware platform

This product has multiple hardware components. While the details of the transmitters and receivers are very important, they are being taken care of by our partner in the Electrical Engineering department and are therefore largely outside the scope of this document. As of Revision 1 of this document, we expect them to be piezoelectric

transmitters and receivers tuned to the 40kHz operating frequency, but changes to these details may not be reflected in later revisions of this document.

We expect to have four (4) individual receivers affixed to the boat at known distances from each other. Each one will detect when a transmitter in the vicinity has broadcast a signal and will convey this to our hardware (discussed below) via a digital signal. When a given receiver is actively “hearing” a transmission, its state will be HIGH. When it is not detecting audio in the chosen transmission range, it will be LOW.

The hardware that will detect the rising and falling edges from the aforementioned receivers must be able to detect these changes extremely accurately and precisely in order for the software to be able to effectively and reliable triangulate signals. We have decided to use Arduino Nano micro-controllers for this purpose, as they have a high clock speed and we can read from the input pins with very little overhead.

Each time a rising or falling edge is detected by the Arduino, the state of the pins is recorded and associated with a timestamp. This packet of data is then transferred via SPI to a Raspberry Pi Zero which takes care of triangulation, collision detection, and UID decoding. Most likely, this machine will also host some kind of web server that an external device, such as a tablet, laptop, or smartphone, can connect to. The triangulation and identification data will be communicated to the user via this web interface.

Software platform:

The Arduino code will be written in C++ and assembly. The Pi’s SPI code will be written in C. We can write the rest of the business logic in C as well, though we may want to add a higher-level language to the mix somewhere to facilitate easy communication with the user. For example, we may want to use Node.js to host a server that the user connects to on another device.

Geographical locations of users:

The research that inspired this project will take place in Coos Bay, Oregon. The product needs to work in all types of weather (except snow) and salty water.

Design and Implementation Constraints

Because we need to be able to detect rising and falling edges extremely quickly in order to triangulate accurately, the receiving computer must run a real-time operating system

or be a barebones micro-controller like the Arduino. The Arduino code must be written in C++ and assembly.

We need to communicate between the micro-controller and the Pi Zero. This will be done over the GPIO pins, so we must use a language that has a library that supports SPI over the GPIO pins.

Assumptions and Dependencies

- It is assumed that the user will have access to the crabs and their habitat, as well as a vessel to transport the equipment in the water (such as a kayak).
- It is assumed that the receivers will reliably detect all incoming signals; that is, all the transmitters and receivers will be correctly programmed and tuned, and all receivers will behave identically to each other.

Features

Triangulation/Direction Algorithm

Priority: high

The software must estimate the direction, relative to the user, of any nearby crabs based on incoming signals. This will be calculated based on the time at which each receiver detects an incoming signal.

Unique Identification of Crabs

Priority: high

Each transmitter encodes a unique identification number (UID) in every transmission. The software must decode this UID and report it to the user every time a transmission is detected.

Collision Detection

Priority: medium

The software must be able to identify when one or more transmitters transmit simultaneously. Upon a collision, the software must discard the bad data (unless it is able to confidently glean some accurate information from it) and it may or may not report to the user that a collision occurred.

External Interface Requirements

User Interfaces

The user interface for this product may be on a tablet, where the user will be able to see the direction a crab is based on the data from the triangulation algorithm. Most likely, we will develop a web interface that can be viewed on any connected device.

Hardware Interfaces

The crabs use transmitters to transmit sounds picked up by an array of receivers. The receivers will pass a digital signal along to an Arduino, which will in turn pass it into the Raspberry Pi, where the computation and interaction with the tablet will occur. For a more detailed description of these components, see the Operating Environment in the Overall Description section.

The exact protocol for UID encoding and decoding will be described in a separate document that is a work in progress as of version 1.1 of this SRS document.

Software Interfaces

Most of this project will be built “in-house” and will not have to interface with other products. All inter-device communication will use protocols that are either well-known (such as SPI) or created and documented specifically for this project (such as the UID encoding protocol).

Communications Interfaces

The Raspberry Pi will most likely host a web server that the user can connect to on a device such as a tablet or smartphone. This part of the project is still in a planning phase, so more specifics will be added to this document in later revisions.

Other Nonfunctional Requirements

Performance Requirements

The triangulation and ID decoding must happen quickly enough for the results to still be relevant when they are reported. That is, we don’t want to take so long to calculate a

crab's direction that it is in a different relative position by the time it is reported on screen. The software should take no more than a few seconds to report a location after a transmission completes.

Safety Requirements

The product's intended area of use (in a body of water) is inherently risky. Users must wear lifejackets whenever operating this product and take steps to avoid electrocution when operating this product. The user must use their own best judgement when navigating the kayak; the software does not include any functionality to warn the user of obstacles that the kayak or the receivers could collide with. All creators involved in the making of this product must make sure that all elements of the product are waterproof.

Security Requirements

If ever networking functionality is added to this product, steps will be taken to protect the data accumulated. At no point should the software accept data from any sources other than the attached receivers; that is, an attacker should not be able to inject false or invalid data unless they create and submerse protocol-compliant transmitters in the area where the research is taking place.

Software Quality Attributes

- Everything used in this product must be waterproof and usable in an aquatic environment.
- We must be able to guarantee that our triangulation is always correct, and accurate to within a 1 meter radius.
- We must be able to guarantee that all collisions will be detected and that the system will never report invalid data as correct.

Other Requirements

The software should be built such that other hyrdrophone array configurations are possible. The spacing between each hydrophone should be easily adjustable, either through a user configuration screen or simply through constants defined in the triangulation code. In other words, it is acceptable to have to recompile the code if the receiver array must be adjusted.

Appendix A: Glossary

Hydrophone: a microphone-like device designed to be used underwater. We use this term loosely to refer to the piezoelectric receivers that we'll be using.

Appendix B: Analysis Models

Appendix C: Issue List

See https://github.com/cabeese/crab-tracker/issues/ for an up-to-date issue list.