An Open Interface for Vexilar SonarPhone Fish Finders

Michael Scherer

Abstract. This document describes a method to interface with Vexilar SonarPhone products. Communication is accomplished by connecting to a T-Pod device over WiFi in the same way that the manufacturer's phone application might connect to it. This method enables the device to be used without the app, allowing for users to implement control and imaging on a variety of alternative systems. Protocol details identified include: controlling range; and feedback on sonar range, status, battery level, serial number, and the water column.

1.Introduction

Fish finders provide a low-cost option for individuals seeking to image a water column and floor of a water body. Although depth sounders are readily available that provide a standardized means of communication, such as NMEA0183 and NMEA2000, there are not any low-cost options for sonars providing a complete water column which can be interfaced with. Ordinarily, such devices are shipped in conjunction with a chart plotter and provide no interface to interface with a user's own software. Such an interface is desirable for the development of autonomous systems, and open source chart plotters.

1.Experimental Setup

The manufacturer sells three products which are branded as SonarPhone devices that share a common smart phone application. Those products are the T-Pod SP100, SP200, and SP300. The SP100 was chosen because it was the cheapest option, and was the most portable. Due to the fact that the SP100 only has a single beam, there are likely some distinctions in in the protocol used by it versus the other products which were unable to be determined in this setup. It is postulated, however, that since they share an application,

they likely closely share a protocol.

	“T-Pod” SP100	SP200	SP300
Maximum Range	120 ft	240 ft	240 ft
First Beam	125 kHz, 30°	200 kHz, 20°	200 kHz, 20°
Second Beam	n/a	83 kHz, 40°	83 kHz, 40°
Mounting	Towed / Casted	Permanent Fixed-mount	Removable Fixed-mount
Retail Cost	129.95 USD	149.95 USD	199.95 USD

Table 1: Differentiation of SonarPhone products

“SonarPhone by Vexilar” Android app, version 1.9 (2014-11-04 Update)
“tPacketCapture” Android app, version 1.7 (2014-06-06 Update)
Android smartphone running Android 4.4.2
Five gallon bucket, filled with water
SP100

The SP100 is turned on when its contacts become wet. The device was placed in the bucket. The phone was then set to connect to the device’s WiFi (identified as “T-POD-37A”). Following this, the tPacketCapture application was enabled and logging. Subsequently, the manufacturer’s application was run, and various settings were manipulated while logging the time of such manipulations.

The logs were then transferred to a computer and inspected using Wireshark, and compared against the hand-written log of setting manipulation.

1.Network topology

The device itself acts as an access point and DHCP server, with a fixed IP address of 192.168.1.1. The device responds to ping messages. The device was probed for open network ports using nmap, identifying several ports were open.

Port Number	Application Protocol
80	HTTP
5000	UPNP

Table 2: Open TCP ports on the SP100

1.UPNP? Server

Connecting to port 5000 with TCP gives an immediate response from the device:

Server "T-POD"

1.HTTP Server

When the device is connected to with a web browser on port 80, the user is prompted to enter a user name and password. Several trivial combinations were attempted but it was not cracked. The investigation of this device did not suggest that the page was integral to the operation of the SP100.

Connecting with wget to display headers indicates that the device may be running an apache-based web server (httpd).

```
HTTP/1.1 401 Unauthorized
Server: httpd
Date: Thu, 01 Jan 1970 00:07:27 GMT
WWW-Authenticate: Basic realm="T-POD"
Content-Type: text/html
Keep-Alive: 3
Connection: keep-alive
```

It is postulated that this server may be used by the manufacturer to configure the device before shipping.

1.Sonar Protocol

By inspecting the packet captures it was determined that the sonar data and control is communicated via UDP. A connection is made to the device on UDP port 5000 and commands are sent periodically to keep the unit alive and transmitting. When these commands stop for a period of **XX** seconds, the device times out and stops transmitting ping messages. Other than this timeout, the rate that control messages are sent from the master does not appear to affect the ping rate of the device.

The device responds with UDP messages on whichever port the master transmitted on. Thus, if the master transmits from UDP port 1234, the device will send all responses to the master's host on port 1234.

1.Master Commands

The master issues two types of commands, both with the same structure.

Byte Index	Description	Possible Values
0-1	Request Type	"FC" or "FX"
2		0x15
3		0x00
4-5	Shared Magic 1	0x2C:01 or 0x00:00
6	Upper Range (feet)	0x00 - 0xFF
7	Unknown1	0x00
8	Lower Range (feet)	0x00 - 0xFF
9	Unknown1	0x00
10-18		0x00:01:00:04:00:00:00:00:00
19	Checksum, sum of bytes 0-18	0x00 - 0xFF
20		0x01
21-26	Shared Magic 2^2	0x58:a2:b5:78:ef:f5
27-28		0x00:00

[1]Postulated to be the upper bits of the range value assuming a little-endian byte ordering. Because the maximal range of any device is 240 feet (0xF0) this could not be confirmed, but it is reasonable to assume that those bytes might be reserved for such a purpose.
[2]This magic blob appears in the ping responses in addition to commands.

1.Server Messages

Server messages have a common framing as described below.

Byte Offset	Description	Possible Values
0-3	Sync Bytes1	0xFF:00:FF:00
4-5	Message size (little endian)1,2	10, 32, 340
6-9	Status code	“REDY” or “BUSY”
10-N	Message Body	

[1]As a UDP protocol, it is unusual that these features are present in the messages. It is postulated that this protocol is designed to be run on other link protocols
[2]This is equal to the size of the entire packet, including the sync bytes.

In practice, the message size appeared to be the most stable indicator of type. Three distinct messages were identified: busy (10), handshake (32), and ping (340) messages. The status code as described is likely a part of the body of the message and not the framing, however no message was observed which did not contain a status code. All body contents

1.Busy (10)

Busy messages simply contain a status code that says “BUSY”. This message appears to indicate an error state. The message was most readily observed when transitioning between air and water, and when the water was too shallow. Busy messages contain no body.

1.Handshake (32)

Sent in response to a handshake message from the master.

Byte Offset	Description	Possible Values
10-11	Response Type	“FX”
12-15		0x16:00:01:00
16-25	Serial Number (ASCII)	
31-36	Shared Magic 2	0x58:a2:b5:78:ef:f5

1.Ping (340)

Ping messages are ...

1.Handshake

When initially connected, a handshake is exchanged between the master and the device. From investigation, this handshake appears to be entirely optional and not necessary to retrieve sonar pings. However, the handshake does provide the master with information not transmitted in the ping messages, such as the device’s serial number.

The master issues its standard command message with the request type of 0xFX. Bytes 3-18 are all zeros. The checksum is calculated as normal, and then the remaining 9 bytes are additionally zeros.

The server responds with its handshake message as follows: