Wireless Remote Control Station for Acoustic Underwater Modems

Daniel Stolpmann, Leonard Fisser, Murat Arisli
Hamburg University of Technology

Introduction

Motivation
- Research group smartPORT develops a low-cost, low-power acoustic underwater modem
- Current communication tests are error prone and time consuming

Project Goals
- Simplify test procedures by making multiple modems controllable from a single point via wireless links
- Improve workflow by real-time visualization of test results

Solution Approach

Each modem is connected to a Wi-Fi module via a serial link
- Infrastructure mode Wi-Fi network via access point
- Single notebook runs backend services and user software

Hardware

- Circuit board stacks on top of modem using pin headers
- Designed around the ESP8266 development board D1 Mini Pro
- 32 Bit microcontroller with integrated IEEE 802.11b/g/n Wi-Fi
- Universal Asynchronous Receiver Transmitter (UART) link between modem and ESP
- Voltage regulator for ESP and power pass-through to modem including fuse, reverse polarity protection and additional battery monitor via ESP
- Options to use separate batteries for modem and ESP or to power ESP by modem

Performance Measurement

- Packet loss test using 1000 Internet Control Message Protocol (ICMP) ping messages:
  - 50m and 100m → 0% packet loss
  - 150m and 200m → 3% packet loss
- The number of packet losses can be considered small enough to be correctable by TCP

Summary

- Extended existing toolchain to control multiple modems at the same time via Wi-Fi instead of connecting to each modem individually via a serial link
- Development of extensible Web UI for easy debugging and performance analysis

Software Architecture

- Communication via Message Queue Telemetry Transport (MQTT)
- Publish/subscribe protocol with central message broker on notebook
- Protection against packet loss by Transmission Control Protocol (TCP) connections
- Connection between web browser and MQTT broker via WebSocket

ESP8266 Firmware

- Based on Mongoose OS and implemented in embedded JavaScript (mJS)
- Forwards raw modem packets between MQTT and UART connection
- Adds/removes escape sequences for UART communication to/from packets
- Publishes discovery message with modem information (ID and software version)

User Software

- Implemented MQTT support into existing Python toolchain as an alternative to using a direct serial link to the modem
- MQTT communication is fully transparent to the user
- Extended existing test procedures to control all modems by a single script

Web UI

- Client-side application written in Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript
- Sidebar with information about discovered modems
- Console with packet dissection for debugging
- Predefined test procedures for quick tests with JavaScript graphs
- MQTT API for visualizing the test procedures of the Python toolchain

Future Work

- Extend Web UI and Python toolchain with more test procedures
- Setup permanent test environment on jetty in Finkenwerder accessible over the Internet
- Make test runs independent of notebook by running backend services on separate server