Software Project for Aiding ARES, RACES, and MARS

Vance Martin

Software Project for Aiding ARES, RACES, and MARS

There is a significant service component to the amateur radio hobby that takes the form of three organizations, focused on service. ARES, the Amateur Radio Emergency Service, and RACES, the Radio Amateur Civil Emergency Service, are both groups that are deployed during emergencies to assist with communications where traditional communication methods are inoperable, or need supplementing. MARS, the Military Auxiliary Radio System, is a similar group, but is sponsored by the US Army, Navy, and Air Force, and is focused on providing communications support to the military during times of emergency or military need. One of the key tools used by amateur radio operators in these groups are “go-kits” or “go-boxes.” These kits are basically portable, self-contained radio stations that can be set up and operated at emergency locations. The drawback is that the station requires an operator, which is not always safe or practical during an emergency. My project will focus on writing software, and combining it with some commercially available hardware, which would allow one of these self contained stations to be remotely controlled and operated.

The core of this project is to write a software class that provides for the interface between a radio and a computer, so that when the computer is remotely accessed, it can control the functions of the radio. The remote control aspect of the project will be demonstrated using wireless routers that have been flashed to operate under FCC part 97 (Amateur Radio Licensing Rules and Regulations) so that they can simulate a wireless link at a distance for interfacing with the hardware, and the software written for it. Commercial software packages that perform equipment control and remote access are available, but they incorporate heavy Graphical User Interfaces, features for hundreds of models of radios, and features for the convenience of amateur operators that are traditionally operating on a standard desktop PC. These PC’s have larger power requirements, and are therefore impractical in an emergency field situation. The software class I propose to write will be written specifically for the radio that is advertised as being the world's smallest "all-mode" radio, the Yaesu 857d, which also allows for serial RS232 communications. The software will also be implemented on a single board computer (popular versions are the Raspberry PI and the Beaglebone) so that the radio and computer could both be run on a 12 volt battery.

While the interface class will be the core of the project, as the software will need to perform the functions normally performed at the radio's control interface, a simple, lightweight program will also be provided, for the purposes of demonstrating the features of the written class. Both the class, and the user interface, will be written in the Python programming language. Python was chosen because during emergency situations software functions could require changing, and Python's strong points are its readability, reusability, and maintainability – all important in high pressure situations. Additionally, because Python is compiled "on the fly" these emergency field changes could be implemented and tested quite rapidly.

At a minimum, the Python class will provide the following functions to interface with the radio's "CAT" system:

- Locking and Unlocking the keypad of the radio
- Engaging and Disengaging the radio PTT (switching between transmit and receive)
- Setting frequencies and operating modes
- Turning the clarifier function on and off, and adjusting its frequency
- Switching between the radio's 2 Variable Frequency Oscillators
- Turning the split operating mode on and off
- Setting repeater offset frequency

- Turning on and setting CTCSS tones and DCS codes
- Reading frequency and Mode Status information from the radio
- Computerized simulation of controls available on the front of the radio

Lastly, there are 2 data streams (RX status and TX status) that are available from the radio, but are not crucial to the operation as outlined above. If design time permits, these data streams will also be read, and some basic feedback information will also be built into the Python software class.

Ultimately, the core part of this project will be creating a Python software class that interfaces with a Yaesu 857d radio, and will run on a small, low power, single board computer. As an auxiliary to the core, a simple interface program will be written to demonstrate the functions of the class. This demonstration will be done by wirelessly connecting to the single board computer through 2 wireless routers in a mesh network and operating the software from there, to simulate remote control of the radio.