Software for Aiding ARES, RACES, and Mars

Vance Martin

Introduction

“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 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. This project will focus on writing a piece of the software package and combining it with some commercially available hardware, which would allow one of these self contained stations to be remotely controlled and operated.” (Martin, Project Proposal)

Project Management Approach

As the sole executor and sponsor of this project, Vance Martin will be responsible for all phases of the project. This will include creating and documenting the requirements for the project, creating the scope for the project, performing the necessary coding and code documentation, and creating the user guide for the produced product. Throughout the project, input may be gathered from members of the Red Rose Repeater Association and the Lebanon Valley Society of Radio Amateurs as these groups both have members involved in organizations through the Amateur Radio Service and could be valuable resources when gathering additional input on the features such a software and hardware project should employ.

Project Schedule

This project will have a very condensed time-line, and will therefore be very specific in scope. This scope will be detailed in later sections in this plan. The preliminary schedule of activities for completion of this project are as follows:

Week 1, ending March 30th 2015:

Week one, which was already completed at the time of the preparation of this document, consists of identification of the project and creation of a high level proposal. Week one also included some basic coding experiments to determine if the needed serial connections to the radio could be easily established.

Week 2, ending April 6th, 2015:

Week two will be primarily focused on creation of the detailed Project Plan, and this document to outline it. The project planning will include basic cost and risk analysis, and will also include the project requirements and scope. Week two will also include some coding, based on the the items that will be fundamental to the project, and can be started while the detailed requirements and scope are created.

Week 3, ending April 13th 2015:

Week three will start with final touches on the project plan, but will be primarily focused on writing the code for the project.

Week 4, ending April 20th 2015:

Week four will consist primarily of writing the code for the project. The end of this week will also include review of the in-code documentation for completeness, and preparation of an outline for the final user documentation. If during week four, these tasks are completed in a timely enough fashion, a second code review will be performed to update code for execution speed and compactness.

Week 5, ending April 27th 2015:

Week five will encompass the final creation of the user documentation, and preparation for final project report-out.

Requirements

The primary objective of this project is to create a software object that can be utilized by other amateur radio operators who wish to develop their own user interface to control the operation of a Yaesu 857d model radio. Because the software object is the primary item being developed, the only requirement for the program acting as the user interface is that it must be minimally viable, and able to demonstrate the functions of the object. With this basic outline, the functional requirements for the software object are as follows:

- The object must include functions to establish, connect, and disconnect to the radio via a serial interface.
- The object must include functions to directly implement the CAT commands as available and outlined in the FT-857D Operators Manual. These functions are detailed here:
    - Locking and unlocking the keypad on the radio’s control head.
    - Engaging and disengaging the radio’s push-to-talk circuitry, effectively switching the radio between transmit and receive.
    - Directly setting the operating frequency of the transceiver.
    - Directly setting the operating mode of the transceiver.
    - Engaging and disengaging the radio’s clarifier function.
    - Directly setting the clarifier offset direction and frequency.
    - Toggling between the radio’s two variable frequency oscillators, referred to as VFO-A and VFO-B.
    - Engaging and disengaging the radio’s split operating mode function, allowing transmitting and receiving on two different frequencies.

    - Setting the radio’s repeater offset direction (plus, minus, or simplex.)
    - Directly setting the radio’s repeater offset frequency.
    - Engaging and disengaging the radios CTCSS and DCS encoders and decoders.
    - Directly setting the radio’s CTCSS tone.
    - Directly setting the radio’s DCS code.
    - Reading the current operating frequency and mode from the radio.
- The Object must also include the following functions:
    - Increasing and decreasing the operating frequency by logical “step” amounts
    - Setting up transmit and receive frequencies along with engaging split operation
    - Setting up operation for a known repeater
    - Switching between “bands” and setting their standard modes (i.e. choosing 20 meter SSB changes the frequency to 14.150 and sets the operating mode to USB)
- The Interface Program must provide a basic text menu and interface to provide access to, and implement all of the above functions, to demonstrate their functionality.

In addition to the functional requirements above, there are a number of non-functional requirements that specify items not directly related to the actions the object will perform. First of these are the technical requirements, as outlined below.

- The software must be able to be implemented on a Raspberry Pi single board computer, because they are readily available, inexpensive, and already very popular for use in the Ham Radio Community.
- The programming language chosen for development of the object should be able to work on multiple platforms, but at a minimum must work with the Raspberry Pi hardware and the distribution of Linux that is generally packaged as the default operating system (Raspian).

- The programming language chosen must be one which allows for rapid and simple development, changes, and maintenance.
- The object created must be targeted specifically to the Yaesu 857d radio, to keep physical disk space used to a minimum.
- The implementations that will most likely use this object are ones where the Raspberry Pi running the software will be remotely accessed via either SSH or via a VNC Server instance running on the Raspberry Pi. Due to this fact, the design of the object and interface programing must not preclude the ability to run them in this fashion.
- Wireless remote access will only be via "Broadband-Hamnet" (information available at www.broadband-hamnet.org) to prevent non-licenses individuals from controlling the radio. Therefore, the design of the object and interface program must not preclude the ability to operate in this manner.

The users of the created software object and interface program will be other developers, and amateur radio operators. For this reason, and due to the technical requirements above, the usability requirements for the object and interface program are minimal. The software needs to be usable from a command line interface, and use terms and methodologies that can be understood and used by someone with a minimum of a General Class Amateur Radio License.

The implementation of this software, and the hardware it is intended to run on, is specified so that it can easily be used "in the field" and as such, reliability and maintainability are critical. The programming language must be chosen such that changes to the software can be made in the field, and implemented immediately. The code written must also be clearly documented, and clearly written, so that an individual with reasonable knowledge of the chosen languages can read, understand, and maintain the software. Lastly, a technical manual must be provided that describes the use, operation, and functions of both the object and the interface program.

Scope

The scope of this project includes the development and design of the software object as outlined in the preceding requirements. It also includes creating a minimally viable interface program, implemented on a minimally viable hardware platform, such that the functional and non-functional requirements can be demonstrated.

To meet the requirements listed for the programming language used to create the object, Python3 will be used. As a cross platform language, it will work excellently with Linux and the single board computers, and is designed to allow for rapid development and deployment. The written code will include in-code documentation to make clear what each function is doing. Each function definition within the class will be followed by a description of what the function does, a description of the arguments it takes, and a description of any object returned. Due to the condensed time-line of this project, the design focus of these functions will be on speed of developments.

The interface class will be a simple text based, menu-driven class, that will give the user access to all of the functions listed above, so that they can be demonstrated. Due to the time constraints of the project, and the fact that the interface software is only intended to be minimally viable, the interface software will not incorporate in-depth input verification, and will assume that the user will, for the most part, be inputting proper values, where vales are requested. The purpose of the interface software is only to demonstrate the functions of the created object, not to provide a full-featured software package.

The non-functional requirements for this project are mostly related to the hardware that the object needs to operate on, and in conjunction with. To meet these requirements, the software will be developed and implemented on a Raspberry Pi single board computer, with the minimum amount of hardware needed to interface with the radio. The Broadband-Hamnet connection to the Raspberry Pi will be provided through a Linksys WRT54G router, flashed to operate using the Broadband-Hamnet protocol. The Raspberry Pi will use an Ethernet connection to this router. The remote wireless link will be demonstrated by wireless connection to a second Linksys WRT54G. The second router will have an Ethernet connection to the laptop that will serve as the remote-control station.

The following items are not included in the scope of this project:

- The software, hardware, or implementation of streaming audio from the remote PC to the Raspberry Pi, or from the Raspberry Pi to the radio. Audio streaming in these arrangements is usually provided via pre-packaged software, and will not be included in this project.
- Creation of an actual "go-kit" or "go-box" radio station. This project is focused on the software needed to implement these functions in a "go-kit" or "go-box". It is not intended to actually create one of these stations.
- Access to the devices via the "Internet" or any publicly accessible network. Because radio operation is a licensed activity, access to the device will only be via the Broadband-Hamnet mesh network.
- Display or interface hardware to directly interact with the Raspberry Pi. Because the focus of this project is on remote access, human interface devices for the Pi are not required. They may be provided for demonstration purposes, but are not part of the scope of this project.

Cost Analysis

Costs incurred for this project will be minimal, as many of the items required are already on-hand. As it is an academic project, the only costs will be for the hardware required to demonstrate the non-functional requirements of the project, and to perform some basic testing and interface verification. In the event someone desires to completely replicate this project however, a breakdown of all of the costs that would be incurred is provided below. Most radio operators wishing to implement this object would however, have much of this hardware already, and the additional costs would be minimal.

Item	Approximate Cost
Raspberry Pi	$30
TX/RX to DB9M RS232 serial host adapter	$20
Yaesu CT-62 CT cable	$23
2 Linksys WRT54G routers (no longer in production, available used)	$20
Ethernet Cables	$15
Yaesu FT-857D transceiver	$850
Basic Laptop (Running Linux.) Old used models are fine for this application	$300

While no costs other than these are expected, any required or incidental costs will be the responsibility of Vance Martin.

Risk Analysis

Due to the condensed time-line of this project, the primary risk facing the project is timely completion. To mitigate the risk of not completing the project on time, the scope of this project has been kept narrow. This is the reason the project is focusing primarily on the software object as its core deliverable, and has excluded streaming audio to the radio as a part of this project. A secondary risk, while unlikely, has the potential to be catastrophic if it occurs. This risk would be a hardware failure involving the transceiver, which would prevent completion of the project and would be quite costly to rectify. Even though the risk is very small, steps such as connecting dummy loads during testing (to prevent potential transmitter damage) are being taken.