In this project, the undergraduate will be tasked with designing, implementing, and testing a graphical user interface (GUI) to control a custom wireless sensor network for geophysical data collection. In this white paper, we explain the desired functionality and design requirements for the GUI enabled base station.

The wireless sensor network will consist of many wireless nodes and a single base station computer with a GUI interface (e.g., Figure 1). The GUI interface will be used to control the wireless sensor nodes in many ways. First, the GUI will have buttons to allow the user to start and stop recording data from the wireless nodes. This will allow geophysicists to acquire “snapshots” of data from, for example, a wireless array of geophone sensors. For this to work seamlessly, the undergraduate must address implementation issues of packet collision avoidance and time synchronization. Fortunately, there has been much work in the wireless sensor network community on such topics, e.g., time-delay multiple access (TDMA) for collision avoidance [1] and message-based time synchronization to align wireless node clocks [2]. Additionally, much of the underlying wireless node and base station software for data collection, collision avoidance, and time synchronization has already been implemented; it will be the undergraduate’s responsibility to incorporate such existing functionality into the GUI enabled base station. To clarify, the undergraduate will not be responsible for any necessary changes in wireless node software. Instead, the undergraduate will seek help from other undergraduate and graduate student experts in specific wireless node software.

Figure 1: The overall design of the GUI enabled wireless sensor network. Using the GUI interface, the user shall be able to start and stop data collection, control acquisition parameters such as sampling rate and gain, and visualize the raw data.

The second functionality of the GUI interface will be to allow users to control data acquisition parameters of the wireless nodes. At a minimum, this includes the ability to adjust, in near real-time, the sampling rate (e.g., 1 - 500 Hz) and signal amplification (e.g., 1-128X gain). If time permits, the GUI should also control the desired input channel (e.g., channel 1, 2, or 3) and the calibration modes (e.g., self, system). Note, again, that the undergraduate will not be tasked with writing the wireless node software to enable such real-time changes; instead, the undergraduate will work with other undergraduate and graduate students to implement the matching wireless node software (much of which has already been written).

The third functionality of the GUI interface will be to visualize the raw data coming in from one of the wireless nodes. This will allow users to see the data as it comes in and make any necessary adjustments to the sensor locations and/or node parameters (e.g., gain). To enable this feature, the undergraduate should use existing visualization tools such as OpenFrameworks or Processing. In other words, the undergraduate will be discouraged to work with low level graphics platforms like OpenGL or X11.

To reach these design requirements, the undergraduate is strongly encouraged to first implement the GUI software in a high-level language such as Processing or Python. Once the overall logic and functionality is validated, the undergraduate must port the software to a low-level, faster language such as C++ (with OpenFrameworks) or Java. Additionally, such software must be cross-platform enabled and should work on Windows, OS-X, and Ubuntu Linux operating systems.

References
