Wireless Independent Noble Gas Sampler: Software Overview

John Lyle, Ethan Fowler, Ian Sepdham, Hasif Shaikh, Khiloni Shah, Derek Haas

The University of Texas at Austin, 10100 Burnet Rd Bldg 159, Austin, TX 78758, johnlyleiv@utexas.edu

INTRODUCTION

Noble gas detection is an important part of monitoring for underground nuclear explosions. Systems like the Swedish Automatic Unit for Noble Gas Acquisition (SAUNA) use beta-gamma coincidence counting with incredibly low detection limits to collect and analyze gas samples [1]. In order to improve noble gas detection, it is important to understand how noble gases will move through the local atmosphere surrounding a nuclear detonation. To support this goal, the Wireless Independent Noble Gas Sampler (WINGS) was designed and built at the University of Texas. Details on the sampler hardware and previous software design are discussed in a separate publication [2]. Fig. 1 shows a completed WINGS unit with fourteen sample containers, a small air compressor, wiring box, and a 12V battery.

To improve the autonomous sampling capabilities of WINGS, a wireless communications network was needed. The control system was required to be operated from a safe distance, contain an error detection and handling subsystem, and operate on an Arduino board at low levels of power draw. To fulfill these requirements a Blynk control panel was set up on an Arduino Mega using an ESP 8266 Wi-Fi microchip to connect to a mobile Wi-Fi router included in each unit.

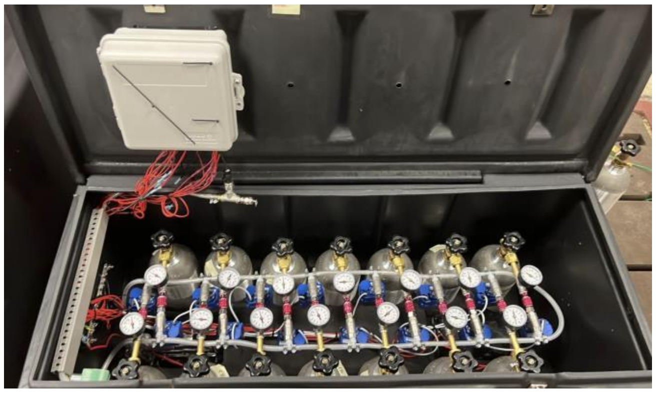
Fig. 1. Completed WINGS unit.

SOFTWARE DESIGN

Blynk is a web interface with modular control panels that allows for full customization of the user interface. Choosing Blynk came from three main needs: controlling a fleet of gas samplers from one device, easy integration with existing Arduino code, and a user-friendly interface that could be operated without any additional software experience. The Blynk interface also displays data which allows for a constant display of pressure levels in the manifold to ensure the sampler is operating correctly. This feature also allows the user to monitor pressure levels on each gas cylinder.

Error Detection and Handling

WINGS was designed to collect pressurized samples using a small air compressor. To mitigate the risk of overpressurization or overheated components, error detection and handling system was needed. The first step in creating this system was to define a safe state for WINGS to exist in as a remedy to any potential errors. This safe state closes all valves to any open tanks preventing the loss of samples, turns the compressor off preventing any further increase in pressure, and opens the vent valve in the manifold to return it to atmospheric pressure. To make sure the gas sampler is in a safe state the gas sampler is constantly checking for errors using multitasking as explained in the next section.

There are three main errors that the error detection system is checking for:

1. The pressure sensor is returning positive values.
2. The pressure must never exceed 160 PSI.
3. The pump should not be on longer than recommended by the manufacturer and operations should follow a 50% duty cycle.

The first error occurs if the pressure sensor reads less than -2 PSI and causes the sampler to enter an error state before returning to the safe state previously described This lets the operator know that the pressure is not returning accurate readings and must be recalibrated as negative gauge pressures should never occur in the sampler. The maximum pressure value for the second error was decided based on the maximum pressure ratings of all components and a detailed pressure safety analysis. If the network fails to cap the pressure at 160 PSI, a pressure relief valve was attached to the manifold and would be triggered at 200 PSI. For the third error, WINGS will return to the safe state if the compressor has been left on for more than 3 minutes and notify the operator of the error.

The error handling system is integrated into the Blynk interface such that any error will be updated onto the interface along with instructions on fixing the error. In most cases a manual inspection is needed to check for broken components or loose connections. The Blynk interface includes a reset button that is pushed by the operator to indicate a manual inspection was performed before standard operations can continue.

Multitasking

For all WINGS operations to run correctly many tasks must be carried out simultaneously. This is not an issue for a modern laptop since each core in a laptop can run its own task. However, this is an issue for an Arduino Mega which only has one core and can therefore only perform one operation at a time. This creates a problem when an operation such as a tank fill which takes a few minutes must be performed. If the Arduino waited for this operation to occur before checking for errors or scheduling additional operations, then most errors would be missed. To work around this a form of simulated multithreading was used. This is called multitasking and allows multiple operations to occur nearly simultaneously.

Multitasking is done by using a global timer with set times that operations are supposed to occur as stored variables. The Arduino then runs a constant loop that checks for error states and then compares the current time to all times where an operation is set to occur. If the time is passed when the operation was supposed to occur, the Arduino then performs that operation. A simple example of this is a tank fill where the selected tank is being filled for a certain amount of time. The obvious way to do this would be to turn the pump on, wait for the time to be reached, and then turn the pump off. However, this method does not use multitasking and would not allow for error checking to occur at the same time. This is undesirable because WINGS would not recognize calibration issues in a pressure sensor while a fill was occurring. To remedy this the following pseudocode can be implemented instead. Here we are trying to fill the selected tank for 10 seconds.

```
Main Loop:
        Error Checking
        Check if timer has hit 10 seconds
    If so turn off pump
    If not repeat loop
```

In the above code the fill has already started, and the loop is running as fast as the Arduino can run allowing the timer to be checked every few milliseconds. This time range is acceptable for every sampler operation and allows WINGS to multitask. Multitasking is not a complete fix since it still requires well-written code. If an operation in the main loop takes too long to occur the rest of the operations have to wait for it to finish running.

Wireless Communications

To support noble gas monitoring, WINGS units would be placed at different locations and must be controllable from a central base station. A few methods were discussed but ultimately a Wi-Fi hotspot was chosen. Wi-Fi hotspots were a significant improvement over the previously used radio transmission since it allowed the WINGS units to be controllable from anywhere with a stable Internet connection. However, this could pose an issue for deployments in remote areas with limited Internet.

RESULTS

The wireless communications network was tested with four WINGS units and was overall successful. The fleet was able to be controlled from one device and samples were successfully programmed and collected. Each WINGS unit could be programmed individually to collect samples at different pressures and at different times. The error handling system consistently detected errors (including loose connections) and would provide instructions to the user as expected. Although WINGS successfully collected samples and showed errors, there were improvements suggested for the next iteration.

The biggest change noted was to allow WINGS to continue fill cycle operations even if the Wi-Fi connection was lost. This was especially important because the current design of the software stops sampling whenever the connection is lost. Before operations can continue, the error must be reset and the previously collected samples must be vented. Another recommended change was to further improve the user-friendliness of the interface and include a display of the battery voltage so the battery could be replaced when needed. Finally, the software operates on central time which would cause difficulties in different time zones. The easiest solution to this would be to update the software to operate on UTC.

SUMMARY

WINGS was designed by a team of students at UT to autonomously collect gas samples for later analysis in support of atmospheric transport modeling. A wireless communications network was created using Blynk and integrated with existing Arduino code. Blynk had a user-friendly interface and could control multiple gas samplers from one device using Wi-Fi. A strong error detection and handling system was introduced to allow for the safe operation of WINGS.

Testing showed that the software functioned as desired, but also revealed shortfalls of the code that could be improved. Improved preliminary testing would find some of the smaller bugs that could have easily been fixed ahead of time. Introducing users to the software would also help them be more familiar with the interface and provide feedback on the usability and friendliness of the interface. Blynk proved effective at programming each WINGS unit to collect gas samples and detecting errors that occurred.

APPENDIX A: SOFTWARE FLOW CHART

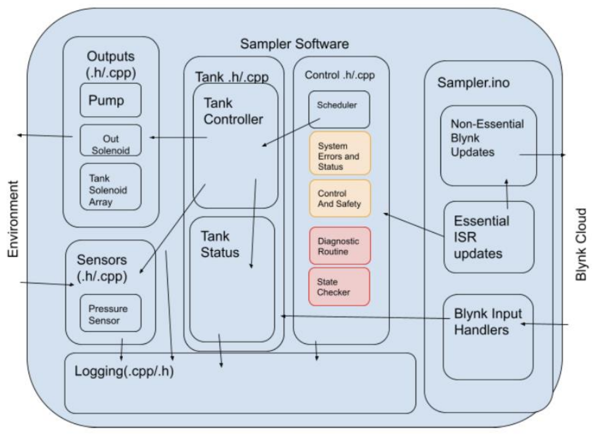

Fig. A.1 Software File System Flow Chart

REFERENCES

[1] A. Ringbom, T. Larson, A. Axelsson, K. Elmgren, and C. Johansson, “SAUNA - A system for automatic sampling, processing, and analysis of radioactive xenon,” *Nucl Instrum Methods Phys Res A*, vol. 508, no. 3, pp. 542–553, Aug. 2003, doi: 10.1016/S0168-9002(03)01657-7.

[2] K. A. Shah *et al.*, “Portable modular gas samplers for nuclear explosion monitoring,” *J Radioanal Nucl Chem*, vol. 331, no. 12, pp. 5305–5310, Dec. 2022, doi: 10.1007/S10967-022-08602-9/FIGURES/5.