Designed for Flexibility: Design and Implementation of the Agricultural Sensor User Interface

Isaiah Snell-Feikema, Devin Moody, Miles Gase, Yi Liu, Wei Wang

Department of Electrical Engineering and Computer Science

South Dakota State University

Brookings, SD 57007

isaiah.snellfeikema@jacks.sdstate.edu, devin.moody@jacks.sdstate.edu, miles.gase@jacks.sdstate.edu, yi.liu@sdstate.edu, wei.wang@sdstate.edu

Abstract

Producers of environmentally sensitive crops such as ice wine require time sensitive readings of important information such as temperature and moisture to produce quality products. Traditionally, this can be costly as it requires dozens of workers remaining on-call during potential harvest nights.

To improve the quality and reduce the cost associated with the production of environmentally sensitive crops the Energy Harvesting Wireless Sensor Network (EHWSN) system was developed. The EHWSN system uses an ad-hoc wireless network to relay sensor information to a hub connected to a USB port of a PC. The Agricultural Sensor User Interface (ASUI) is developed to store and present readings and network information from the EHWSN. The ASUI presents this information in a clear manner to allow the data to be easily monitored and analyzed, reducing production cost and increasing product quality. This paper focuses on the design and implementation of the ASUI.

1 Introduction

Producers of sensitive crops can spend a large sum of money on workers who are monitoring the temperature and moisture of their crops. Often, this is cost inefficient since some nights workers are on standby and end up not being needed but still have to be paid for their time. To solve this problem, the Energy Harvesting Wireless Sensor Network (EHWSN) was developed. This system can be used to wirelessly relay readings from the sensors in the field. However, this data is not easily accessible or useful to producers of sensitive crops. Therefore, the Agricultural Sensor User Interface (ASUI) is developed in order to present the data in an easily accessible way. ASUI uses the readings from the EHWSN to present real time information, a logical network map of the sensor nodes, and historical reading information. The goal of ASUI is to reduce cost while also improving the quality of the producer's crops. This paper discusses background information in section 2, the architectural design in section 3, and implementation along with how the implementation supports architectural design in section 4, and gives a conclusion in section 5.

2 Background

The ASUI system is useful to producers of delicate crops. For example farmers who grow grapes to make a particular type of wine called ice-wine. Ice-wine must be an exact temperature for a certain amount of time before it is harvested in order to qualify as ice-wine. Many times farmers have to call in their staff in the middle of the night in the winter (because that is when the ideal temperature is usually reached) only to find out that the temperature is actually not quite cold enough yet. So all the workers are sent home and paid for three hours of work. The ASUI system solves this problem by using the EHWSN and allowing the farmer to quickly check the moisture and temperature readings from the nodes and pull up a recent history of each node before calling in workers.

2.1 Sensors

This project uses sensors from the EHWSN. The sensors use an ad-hoc wireless network in order to communicate information. Nodes will broadcast their information to other nodes that will relay the information until the readings reach the hub. These sensors can relay information such as temperature, moisture, and battery life of the sensor. Data is passed using two different string formats. Using regular expressions, the strings will match one of two formats:

```
NODE: (?<MessageOwner>\d+) is following NODE: (?<FollowedID>\d+)
```

```
NODE: (?<MessageOwner>\d+) BATTERY: (?<Volt>\d+) TEMP: (?<Temp>\d+)
DAMPNESS: (?<Mois>\d+)
```

Through the use of these regular expressions, it is easy to add more reading types to the EHWSN nodes because more readings can be easily added to the regular expression.

2.2 Overview of Requirements

The ASUI allows a user to see the data from each sensor in a GUI. There requires a logical view of the sensors in a list that displays sensor data. A timeline trend of data should also be available along with the ability to save the graph as a picture. Network maps for the sensors are automatically populated and hovering over a sensor will display information relating to that sensor. Figure 1 shows the ASUI system perspective.

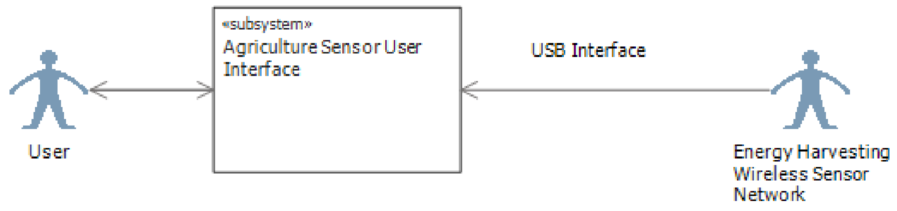

«subsystem»
Agriculture Sensor User Interface
USB Interface
User
Energy Harvesting Wireless Sensor Network

Figure 1: System Perspective

3 Architectural Design

This section describes the architectural design of the system. First, motivating design issues are presented. Next, design decisions are derived from the design issues. Finally, the design decisions are used to create a flexible and maintainable architectural design.

3.1 Design Issues

Based on interviews and workshops with the customer the developers identified the two most critical issues with the greatest influence on the architectural design of the system:

1. *Input Format and Source*
   Currently, the readings can be read from the EHWSN via strings written to the serial port. However, the format of the strings is likely to change in the future if more readings are added or the EHWSN system is revised or replaced. For instance, wind speed readings might be added, the EHWSN might provide readings via a more compact bit stream rather than a character string, or the EHWSN might be modified to no longer use the serial port to write data.

2. *Sensor Reading Types*
   Reading types are likely to be added or removed in the future. For instance, solar irradiance readings might be added. Additionally, some sensors may produce different reading types. For example, a special purpose sensor might only produce wind speed readings, while other sensors produce temperature and moisture readings.

From the issues enumerated above it can be seen that flexibility and maintainability are the foremost concerns that should shape the architectural design. The system must allow for variable types of readings and changes to the form of the input from the EHWSN.

3.2 Design Decisions

The design decisions directly follow from the design issues.

1. Readings will be abstracted into a discrete numerical value that is a function of sensor id, reading type and time.
2. The input from the sensor will be converted into discrete readings.
3. Readings will be accessible via sensor id, reading type and time.

3.3 Design of the ASUI Architecture

To address the design decisions to support further requirements changes, the likely changes are encapsulated into modules in the system architecture [3]. An abstract interface [1] is used to capture the behaviors of each module based on the design decisions. When the change happens, a new implementation of the abstract interface will be provided for that specific module that is related to the change while other modules will not be impact.

The entire ASUI system is divided into two subsystems, GUI subsystem and Sensor subsystem, as shown in Figure 2.

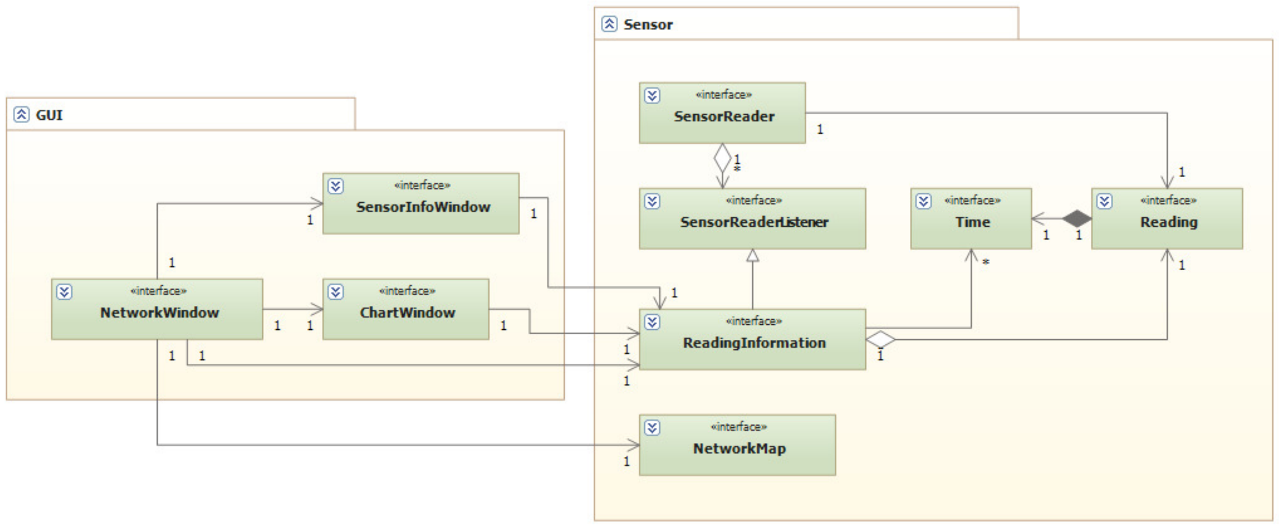

Sensor
GUI
«interface» SensorReader
«interface» SensorReaderListener
«interface» Time
«interface» Reading
«interface» ReadingInformation
«interface» NetworkMap
«interface» SensorInfoWindow
«interface» NetworkWindow
«interface» ChartWindow
1
*

Figure 2: High-Level ASUI Architecture

3.3.1 Sensor Subsystem

The sensor subsystem handles the capture, storage and retrieval of sensor readings. SensorReader encapsulates the process of reading. Readings are considered to be numerical values that correspond to a certain sensor id, reading type, and time stamp. The SensorReader handles the reading of input data from the sensor network via the serial port and then notifies SensorReaderListeners of each new reading. Although the SensorReader may read multiple sensor readings at a time, the readings separated and outputted from the SensorReader separately. This enables other modules to more easily process and display the readings. The method of reading storage and retrieval is encapsulated by the ReadingInformation module. It receives readings from the SensorReader and handles the storage and retrieval of sensor data, such as sensor readings and the current network graph. Figure 3 illustrates the details of each module.

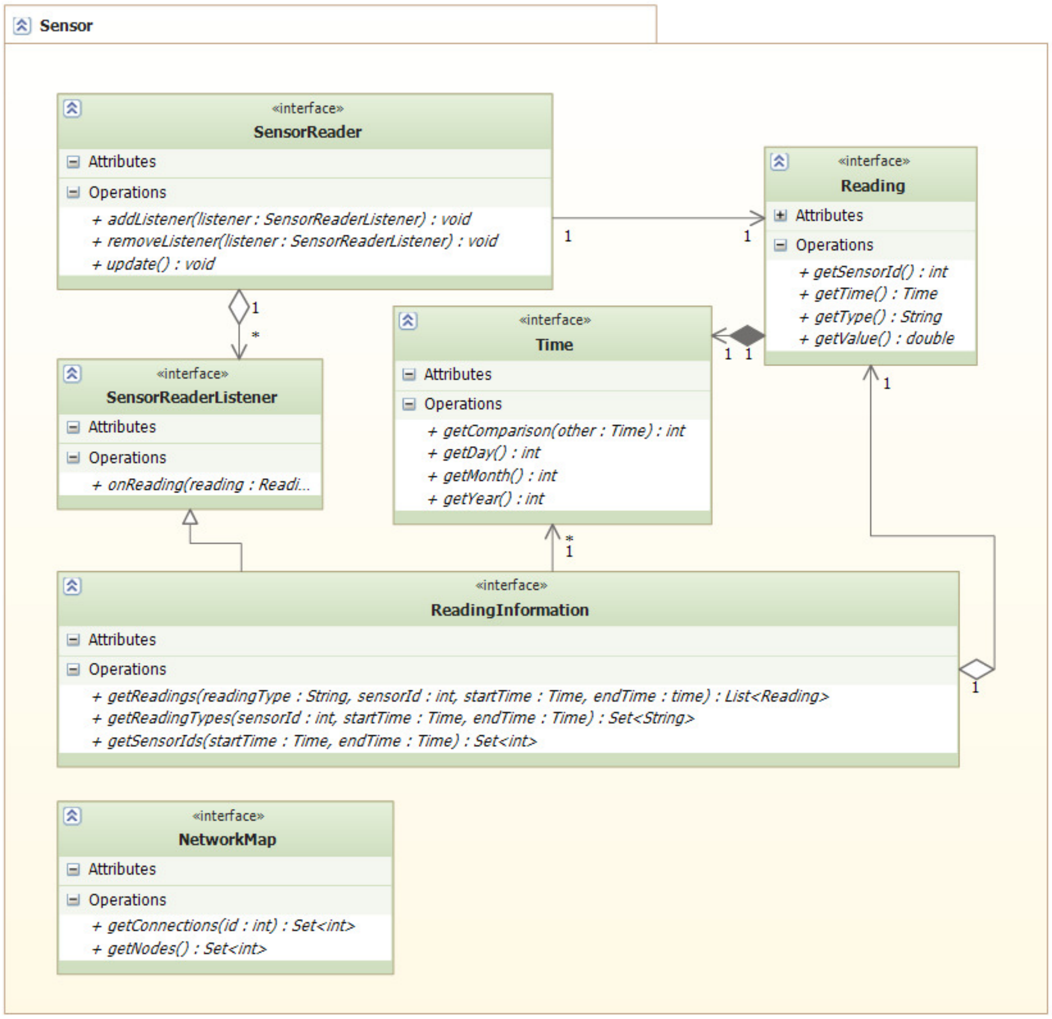

Sensor
«interface»
SensorReader
Attributes
Operations
+ addListener(listener : SensorReaderListener) : void
+ removeListener(listener : SensorReaderListener) : void
+ update() : void
«interface»
Reading
Attributes
Operations
+ getSensorId() : int
+ getTime() : Time
+ getType() : String
+ getValue() : double
«interface»
Time
Attributes
Operations
+ getComparison(other : Time) : int
+ getDay() : int
+ getMonth() : int
+ getYear() : int
«interface»
SensorReaderListener
Attributes
Operations
+ onReading(reading : Readi...
«interface»
ReadingInformation
Attributes
Operations
+ getReadings(readingType : String, sensorId : int, startTime : Time, endTime : time) : List<Reading>
+ getReadingTypes(sensorId : int, startTime : Time, endTime : Time) : Set<String>
+ getSensorIds(startTime : Time, endTime : Time) : Set<int>
«interface»
NetworkMap
Attributes
Operations
+ getConnections(id : int) : Set<int>
+ getNodes() : Set<int>
1
1
1
*
1
1
1
*
1
1

Figure 3: Sensor Subsystem

3.3.2 GUI Subsystem

The GUI subsystem handles interaction between the user and the system. The GUI modules dynamically determine the reading types and obtain readings by querying the ReadingInformation module in the Sensor subsystem. SensorInfoWindow displays up to date reading information for a particular sensor. NetworkMapWindow displays logical map of sensors and current sensor readings. GraphWindow is used for viewing historical sensor data as a graph. The details of the design of the GUI subsystem are shown in Figure 4. Figure 5 shows a screenshot of the network map window.

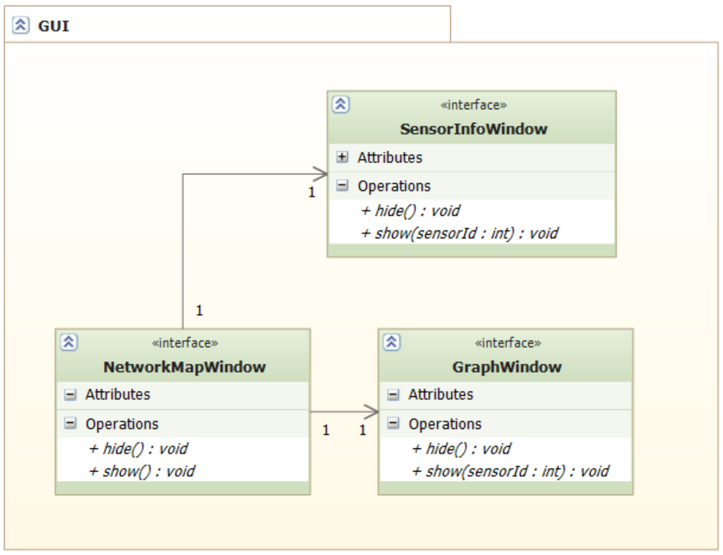
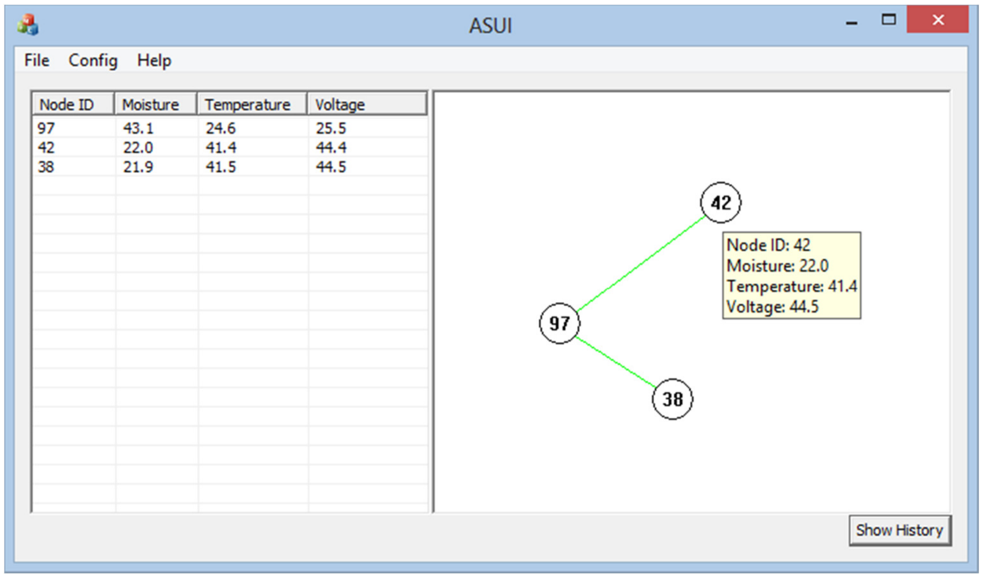


Figure 4: GUI Subsystem



Node ID	Moisture	Temperature	Voltage
97	43.1	24.6	25.5
42	22.0	41.4	44.4
38	21.9	41.5	44.5

Node ID: 42
Moisture: 22.0
Temperature: 41.4
Voltage: 44.5

Figure 5: Network Map Window

4 Implementation

The ASUI is required to run on Windows, be written in C++ and use Microsoft Foundation Classes (MFC) [2] for the GUI subsystem. C++ is the institutional language of South Dakota State University and MFC is likely to be used by future maintainers of the system. The system is designed for Windows because Windows is the currently the most popular operating system for single users.

The SensorReading module runs as a daemon thread behind the GUI. This ensures that no input from the sensors is missed. Input strings from the sensors are broken into multiple Reading objects as they are read. Figure 6 shows the IReading interface. The Readings are placed in a blocking queue and disseminated to the attached SensorReaderListeners when the update method is called.

```
class IReading {
public:
	virtual ~IReading();
	virtual std::unique_ptr<IReading> copy() const = 0;
	virtual const int getSensorId() const = 0;
	virtual const ITime* getTime() const = 0;
	virtual const std::string getType() const = 0;
	virtual const double getValue() const = 0;
};
```

Figure 6. The IReading Interface.

The IReadingInformation interface is shown in figure #.

```
class IReadingInformation : public virtual ISensorReaderListener {
public:
   virtual void update() = 0;
   virtual std::list<std::unique_ptr<IReading>> getReading(const int sensorId,
       const std::string readingType, const ITime& start, const ITime& end) const = 0;
   virtual std::set<std::string> getReadingTypes() const = 0;
   virtual std::set<int> getSensorIds(const ITime& start, const ITime& end) const = 0;
};
```

Figure 7. The IReadingInformation Interface.

To support the design objective of flexibility, GUI components are dynamically populated by querying the IReadingInformation interface to determine the number and type of readings for a specified time interval. Figure 7 and 8 show how the GUI components query the IReadingInformation interface.

```
std::set<int> ids = readingInformation->getSensorIds(startTime, endTime);

for (std::set<int>::iterator idIter = ids.begin(); idIter!= ids.end(); idIter++) {
   ...
   for (std::set<string>::iterator readingTypeIter = readingTypes.begin();
        readingTypeIter != readingTypes.end();
        readingTypeIter++) {
      std::string readingType = *readingTypeIter;
```

```
        std::list<unique_ptr<Sensor::IReading>> readings =
               readingInformation->getReading(currentId, readingType, startTime, endTime);
        ...
    }
    ...
}
```

Figure 8. Iterating Readings in the GUI

5 Conclusion

The ASUI is a great tool for producers of sensitive crops. The ASUI makes it quicker and easier to monitor and analyze soil readings pertinent to the crops. This increases the overall productivity and quality of the crop while reducing the costs. The ASUI is also flexible enough that the sensors are able to be modified to take different readings without requiring changes to be made to the ASUI. This flexible design also makes the ASUI useful for other applications, such as environmental monitoring, since the appropriate sensors would just need to be added to the EHWSN nodes with slight modifications to the network code.

References

[1] Britton, K. H., R. A. Parker, and D. L. Parnas. A Procedure for Designing Abstract Interfaces for Device Interface Modules. In *Proceedings of the 5th International Conference on Software Engineering*, pp. 195-204, March 1981.

[2] Microsoft Foundation Class Library. http://en.wikipedia.org/wiki/Microsoft_Foundation_Class_Library. Last accessed: 3/22/2013

[3] Parnas, D. L. , On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, *15*(12), 1053-1058.