Judicael Briand Djoko

CS2310 – Final Project Report

Description

The SIS allows health officials to monitor patients through a GUI. Vital signs such as temperature, blood pressure, EKG could be viewed and alerts could be set. Following the objectives set by the established T1 project; the goal is to extend its features to provide the physician with more information regarding his context. T1 allows a doctor to continuously monitor the temperature of a patient using the SIS GUI. However, from my perspective, capturing the patient's temperature alone might not be enough regarding his context. In this light, I decided to implement another temperature sensor, one that monitors the patient's environment. Patients, especially the elderly, are sensitive to their environment. Therefore, having information regarding its state might help the physician or health official take more informed steps in caring.

Implementation

Below are the assumptions taken by the project:

- There would be two temperature sensors: Ambient and Patient.
- Both sensor readings would continuously be output to two files.

With these, the project's architecture is as such: A main component, PatientAmbientTemperature, reads the output from both sensor readings and communicates with the SIS through messages. As it is, the PatientAmbientTemperature component has 4 messages:

i. **CreatePatientAmbient.xml**: Creates the main component.

ii. **GUIToPatientAmbientTemperature.xml:** Simulates the reading request from the GUI to Component.

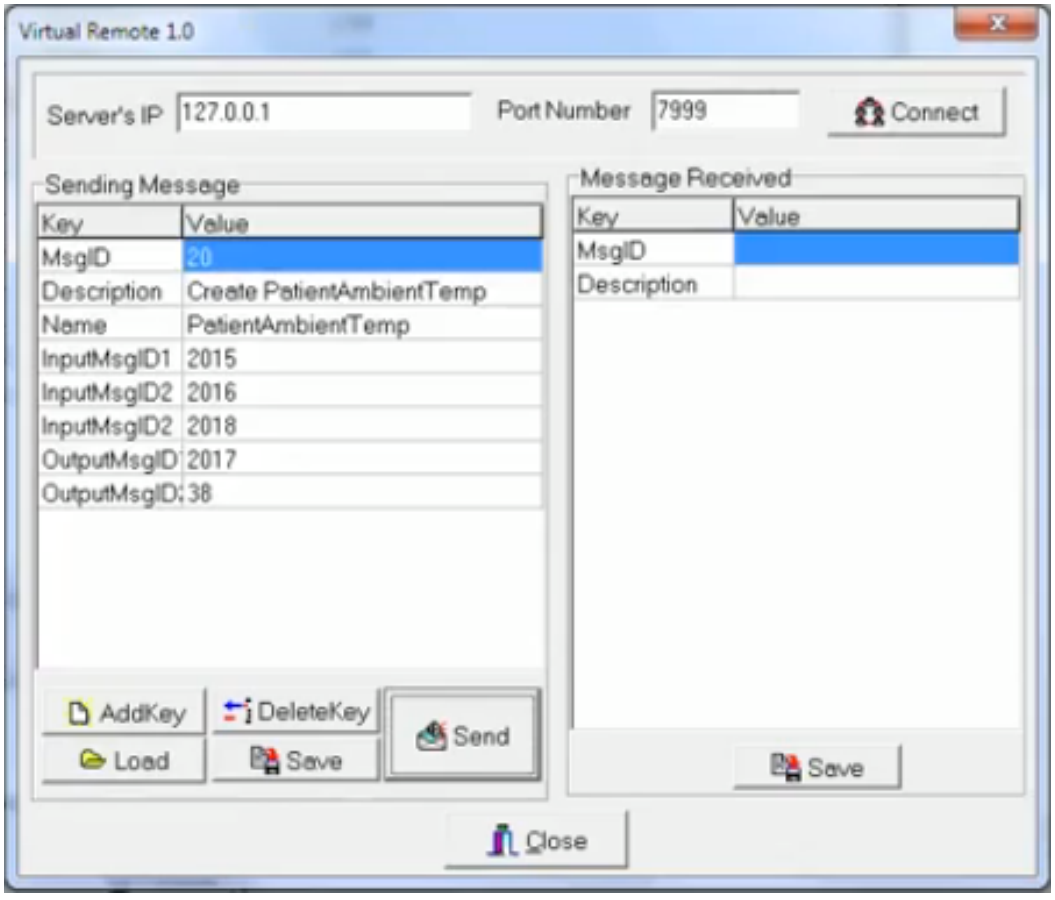
iii. **ShowPatientAmbientGUI.xml:** Displays Component GUI controls.

iv. **StopPatientAmbientGUI.xml:** Hides the GUI of the Component.

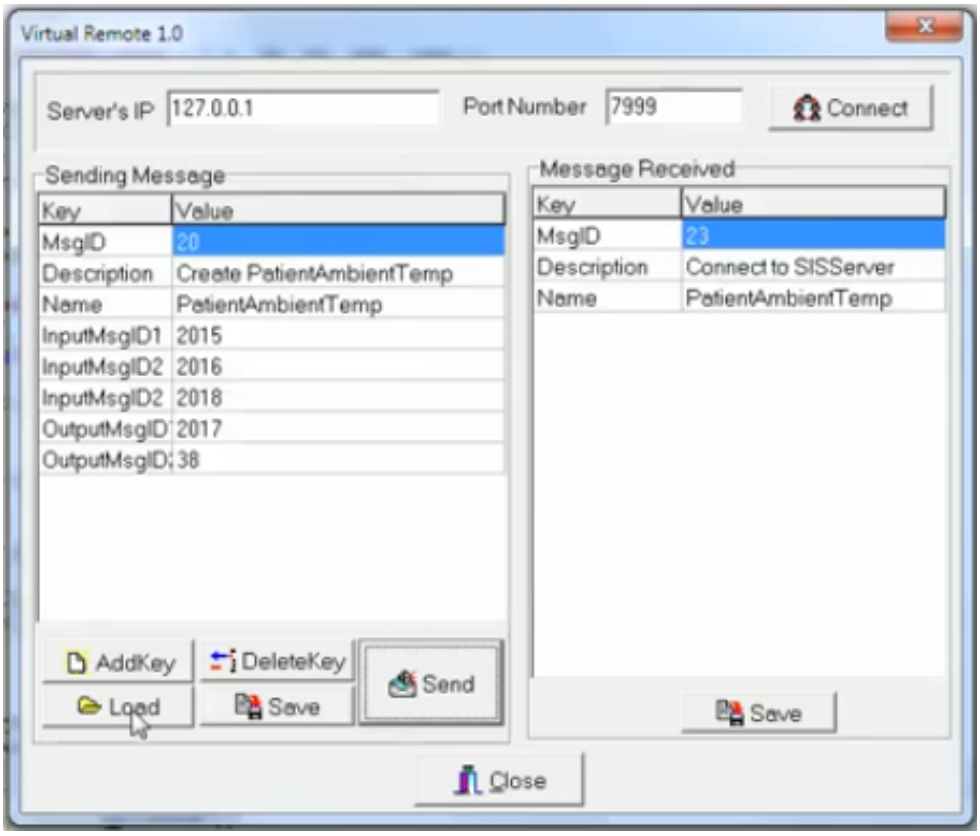
These four messages together with the alert message to the uploader describe the communication of the Component with the SIS system.

To start the system, one needs to have the SISServer running. This is the brain of the SIS system; it broadcasts the messages to all the components in the system. Next, the prjremote program, capable of sending XML messages to the SIS server needs to be launched. This allows the simulation of different scenario as it allows for the crafting of custom messages before transmission.

First load the CreatePatientAmbient.xml in prjremote to start the component

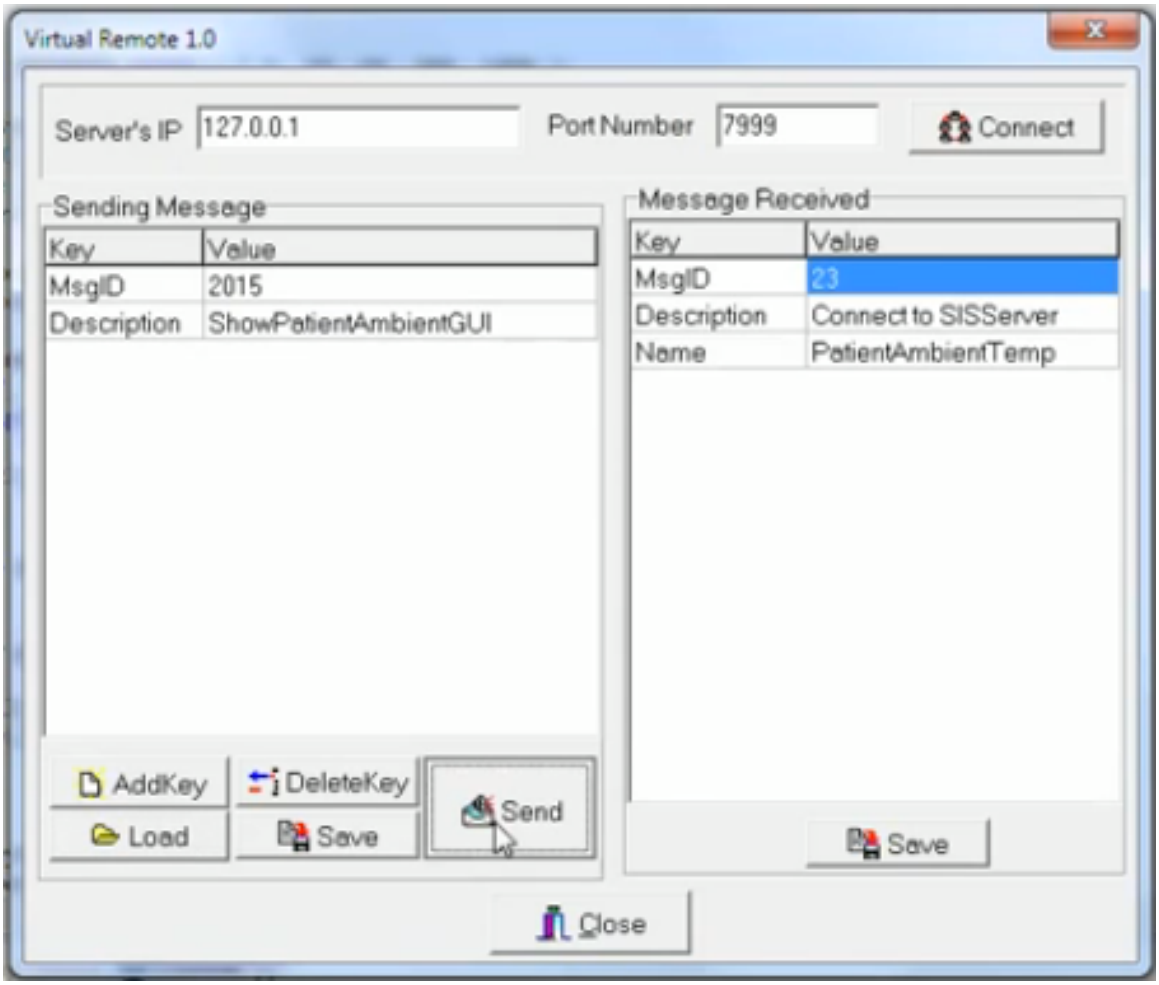


As you can see from the component description, the PatientAmbientTemperature expects to send 3 messages and receive 2 (it includes the alert). Next, start the application; this would connect to the SIS server and report it has been successfully connected. Since prjremote runs in "promiscuous mode" (a phrase from networking), it would display the response from the created component.

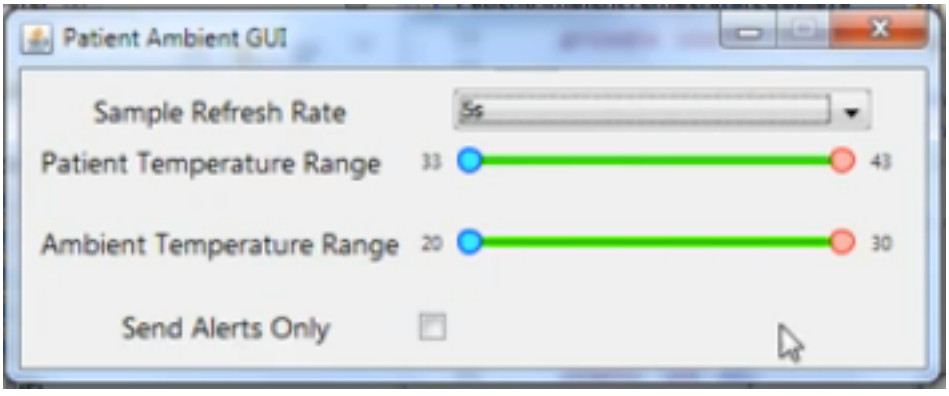


With the component registered to the SIS server, we're now able to receive messages from it. By default, the PatientAmbientTemperature sends no messages unless prompted by the SIS GUI or configuration set in its own GUI. To activate its message sending, one is to send either the ShowPatientAmbientGui or GuiToPatientAmbientTemperature messages.

Continuing on the same procedure as we did in initializing the component, we load the xml message into prjremote and send it to the SIS Server.



This message when received by the PatientAmbientTemperature component would create the GUI used to control the component.



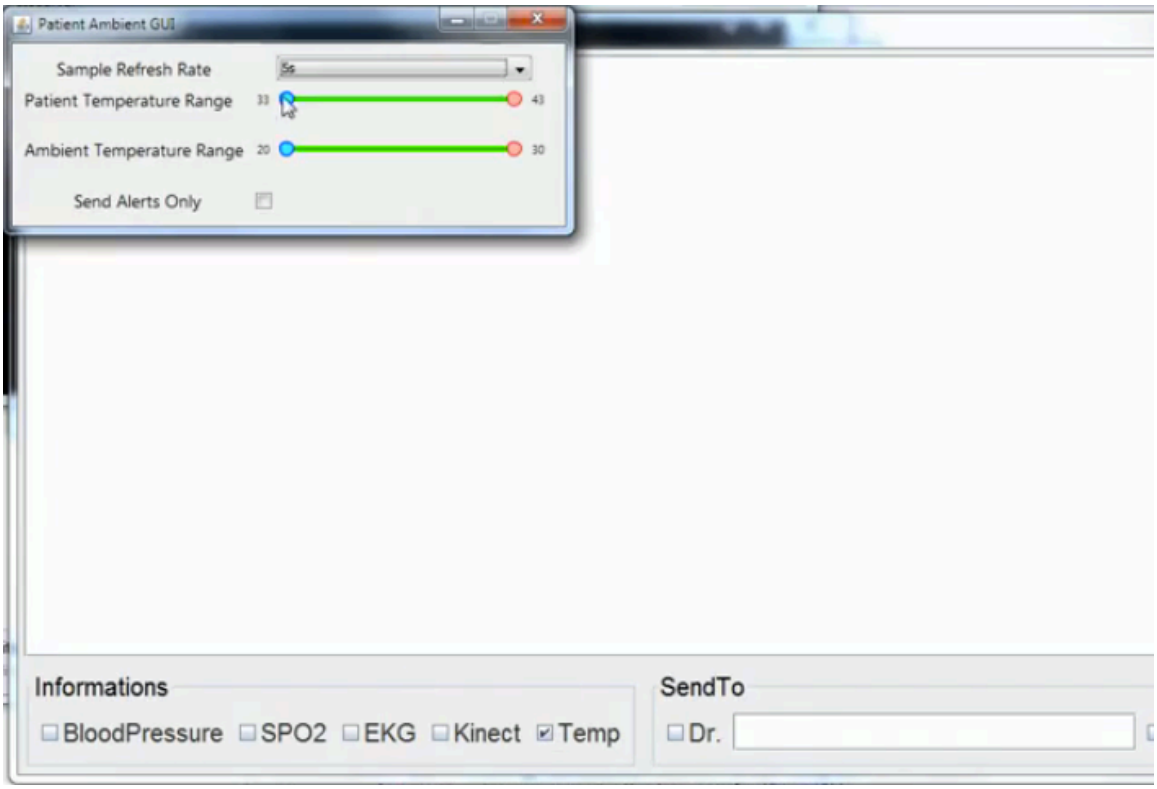
Above is the GUI of the PatientAmbientTemperature. It allows the control of 4 different parameters:

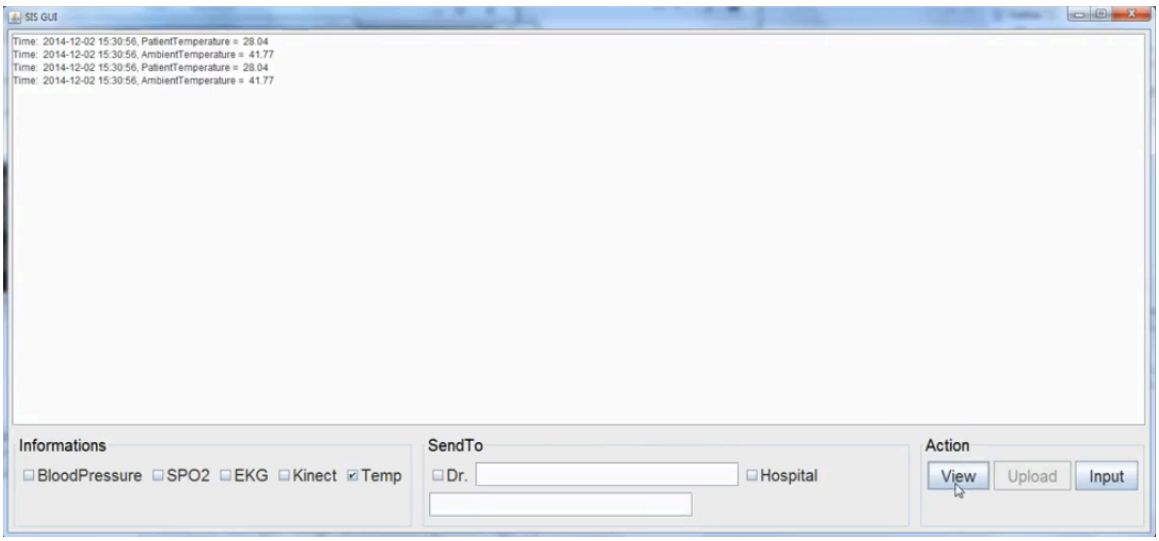
1. Sample Refresh Rate: This is the rate at which the component has to go fetch the content of the temperature output and return its value. As stated before, we assumed the reading of the sensor is continuously fetched and written to a file. Provided with the SIS is the tempver program. This program is in charge of communicating with the temperature sensor and writing the output to a file. It writes its format in cvs whose fields are an ID (autoincremented number), the date of the reading and the value of the reading. To the best of knowledge, Tempver writes to the file every second. So, when setting the Sample Refresh Rate in PatientAmbientTemperature component, it waits for the desired time before reading from the file. It then returns the last reading as the "current" one.
2. Patient Temperature Range: Sets the range at which the patient's temperature is considered safe. Out of this range would issue an alert to the health official about the state of the patient.
3. Ambient Temperature Range: Same principle as in the patient temperature. If the reading of the Ambient temperature is out of the specified scope, an alert is triggered and a message is sent to the SIS server.
4. Send Alerts: At a high sampling rate, the output message from the PatientAmbientTemperature GUI could be overwhelming. The user can selectively choose to send alerts only. In our case, the doctor would only receive messages when the patient or ambient temperatures fall out of range.

Setting any of the values immediately has effect on the workings of the component.

As this is all message based, the component could be integrated with the existing SIS GUI. To demonstrate this principle, I made some slight changes to the full SIS testbed. For this, I made some slight modifications to the CreateGUI component to send messages to PatientAmbient component instead of the native Temperature component. Here're the specific changes

- When clicking on the Temp checkbox, it launches the PatientTemperature GUI. Unchecking it closes it
- Clicking on the "View" returns the current readings from the temperature sensors.





The CreateGUI component was also modified to accept messages from the PatientAmbientTemperature component. This allows the SIS GUI to be extended to integrate with the PatientAmbientTemperature more tightly.

Conclusion

This proof of concept could be further advanced. More functionality could be added to provide the health official with more insight with the patient's context. An idea that comes to mind is the "trace mode"; the component sends the temperatures of the past hour whenever there's an alarm. It helps in restricting the output to a "meaningful" set. In addition to this, finer controls could be added to the component GUI.

Part of the submit package is a temperature generator. Since the Tempver program only supports one sensor at a time, there was need to implement this for testing.