Milestone 2: Software Design and Engineering

Chris Charlebois

RaspberryPi Remote sensor

*Briefly describe the artifact. What is it? When was it created?*

This is a RaspberryPi running a Python script that reads data from a light sensor and a temperature and humidity sensor. It saves this data locally and sends the data to a remote database. The RaspberryPi uses a Seeed Grove Hat to handle the Seeed peripherals and the interrupts. The script is written in Python 2.7 and uses the requests library to handle the HTTP Post method. It was created as part of a course on embedded operating systems.

*Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?*

I selected this artifact as part of my portfolio because it is the single most useful and complete product from my education at SNHU. Embedded systems provide a huge opportunity to implement information technology systems in ways that have real world effects. In particular the various peripherals can be used in a number of interesting ways. I enhanced this project by using the requests Python library to establish communication so that the sensor data is sent to a central repository.

*Did you meet the course objectives you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?*

This project required the systems design to collect the data and to export the data to the central repository. This also required error handling and error messages. In addition to recording the data locally and reporting to the server, the system also queues readings in memory if the communication fails and sends the entries when communication is restored.

*Reflect on the process of enhancing and/or modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?*

Error handling was both the most challenging and enjoyable part of this project. I had to make several attempts at catching the error and identifying the problem. I really liked the queue that I created to account for communication errors. At first, I thought of using a second file used only to temporarily hold the data that could not be sent. But using the file system to store temporary data is not efficient and not easy withing Python. It is also not necessary for this purpose. I am already saving the data locally, so in case of a script error or system crash, the recordings will be recoverable in some way. And storing the queue in memory made it very simple to store the readings and attempt later when connection is restored. Especially since the reading has a time stamp so the order in which the entries are uploaded is not critical.