Introduction

What follows is a description of an “Internet of Things” inspired, do-it-yourself, background radiation monitor. The implementation below places a high priority on “open source” software and hardware. Avoiding proprietary, closed source software allows for much greater flexibility. This flexibility facilitates ease of deployment and scale-out on a large variety of open source platforms, and a large variety of end user applications.

Prerequisites for building and deploying the DIY radiation monitor include: soldering, ability to follow electrical hookup guides, knowledge of Arduino technology, Linux system administration, and some knowledge of the C and Python computer languages. The open source software provided should be sufficient for most applications with only minor modifications. The server side software has been designed around the file structure and system configuration found in most common Linux distributions. In most cases, the majority of modification will be in the HTML, customizing the look and appearance of the web page presentation to the end user.

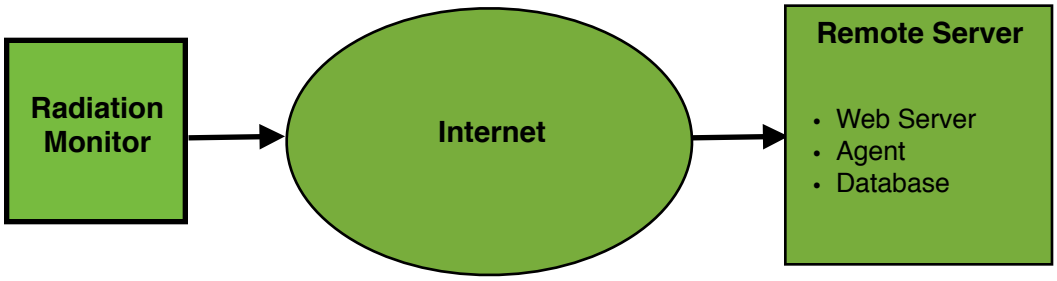

Figure 1. Overall conceptual view showing radiation monitor, network, and server components.

Radmon System Overview

Referring to figure 1, the DIY radiation monitor consists of three major components. They are the radiation monitor, the network connecting the radiation monitor to the remote server, and the remote server. The network connecting the two together, including wireless access points, switches, routers, etc., will be considered background infrastructure beyond the scope of this project description.

The radiation monitor, itself, consists of several hardware components, as well as a software component running on the Arduino micro-controller. The remote server hardware can be anything from a full up Linux server to a Raspberry Pi. The remote server software consists of two components: an agent for processing data from the radiation monitor and HTML documents for displaying the data in a web page.

The remote server agent periodically sends an HTTP request to the radiation monitor. The monitor replies with a Java Script Object Notation (JSON) formatted text string. The agent parses this string, converts and reformats the data, updates a database, and generates graphical charts of the data. The agents sends the reformatted data to the HTTP server by writing to a JSON formatted file, which Java Script, embedded in the HTML documents, can read and display in web pages.

Component Descriptions

Radiation Monitor Hardware

Referring to figure 2, the radiation monitor consists essentially of only two components: an Arduino Uno with attached Ethernet shield and a Mighty Ohm Geiger counter. Both components are mounted in the same box. The project repository contains an photograph of the components mounted in a simple, plastic box. (See the file *RadiationMonitor.jpg*.)

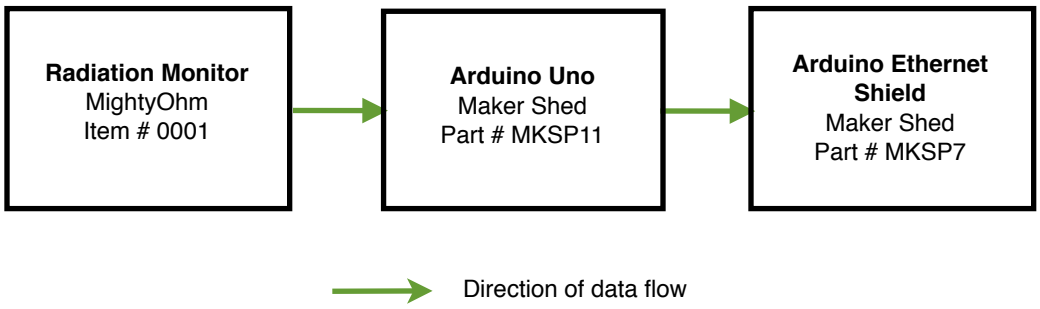

Figure 2. Block diagram of radiation monitor hardware showing individual electronic components.

Radiation Monitor Software

The software for the radiation monitor micro-controller consists of an Arduino “sketch”, along with an ancillary library that must be included at compile time. Referring to figure 3, the main module, or sketch, may be viewed as a “stack” comprising three layers. The top most layer includes the setup routine that runs once at boot up time, and the main loop that contains all the routines that run forever after initial setup. Initial setup runs routines that initialized the Ethernet interface and synchronize the system clock with a network time server. The main loop runs the 2nd layer routines, referred to in figure 3 as “Listen for Serial Port Command”, “Listen for Internet Client”, “Process Geiger Counter Data”, and “Synchronize System Clock”. In turn Listen for Serial Port Command calls “Process Serial Port Command”. Listen for Internet Client calls “Transmit HTML Document”, or “Transmit JSON String”. Synchronize System clock calls “Get Time from NTP Server”.

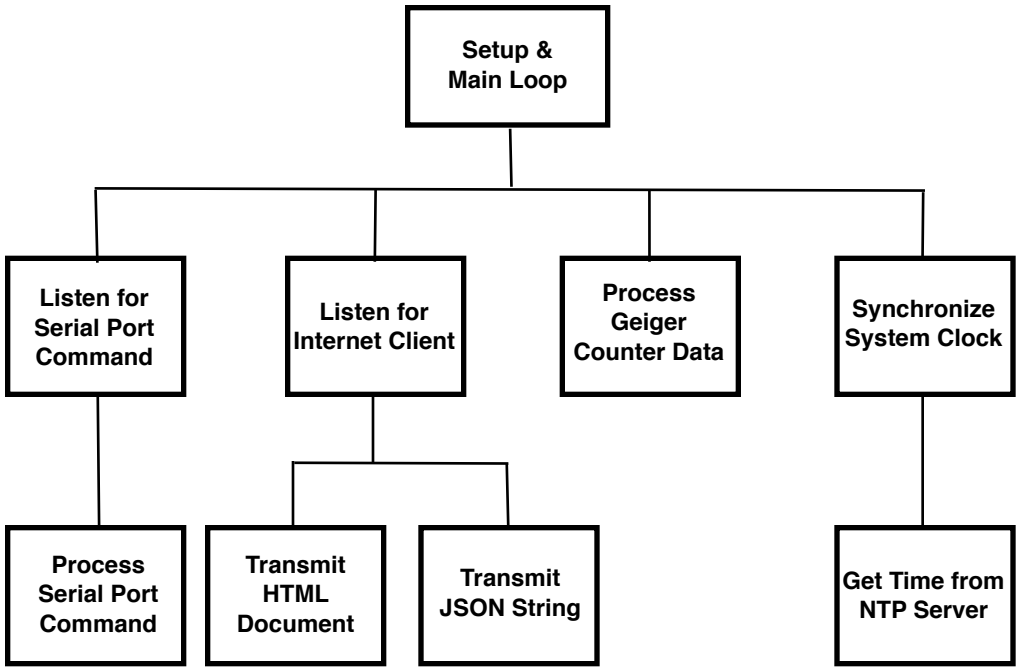

Figure 3. Micro-controller software block diagram showing high level components.

Listen for Serial Port Command intercepts keyboard characters received via the Uno's USB serial port. In a terminal session connected to the USB serial port, typing a "c" causes the radiation monitor to switch to command mode by calling *Process Serial Port Command*. In command mode, IP address mode, network time server IP address, and verbose mode can be changed via the command line.

Listen for Internet Client checks for HTTP client requests and triggers the appropriate response depending on the URL sent by the client. The response may be either an HTML document or a JSON formatted text string.

Process Geiger Counter Data processes the serial data stream from the MightOhm Geiger counter. The MightyOhm sends out data in a CRLF delimited stream of ASCII characters, for example, a typical line has the form

```
CPS, 0, CPM, 22, uSv/hr, 0.12, SLOW
```

Process Geiger Counter Data collects this data, byte by byte, in a character array. When a LF character arrives the contents of the array gets copied to a temporary buffer for use by Listen for Internet Client when Internet clients request the radiation monitor data.

Synchronize System Clock gets the current universal coordinated time (UTC) from *Get Time from NTP Server* and sets the system clock to the UTC. When an Internet client

requests data from the radiation monitor, the current UTC gets sent to the client along with the radiation data.

Remote Server Software

Referring to figure 4, the remote server components consist essentially of two pieces: an HTTP server component and an agent component. The HTTP server component includes the HTML document, with embedded Java Script, for displaying the radiation monitor data in a web browser. The agent, a Python script, manages data conversion and re-formatting, updating a database, and generating graphical charts.

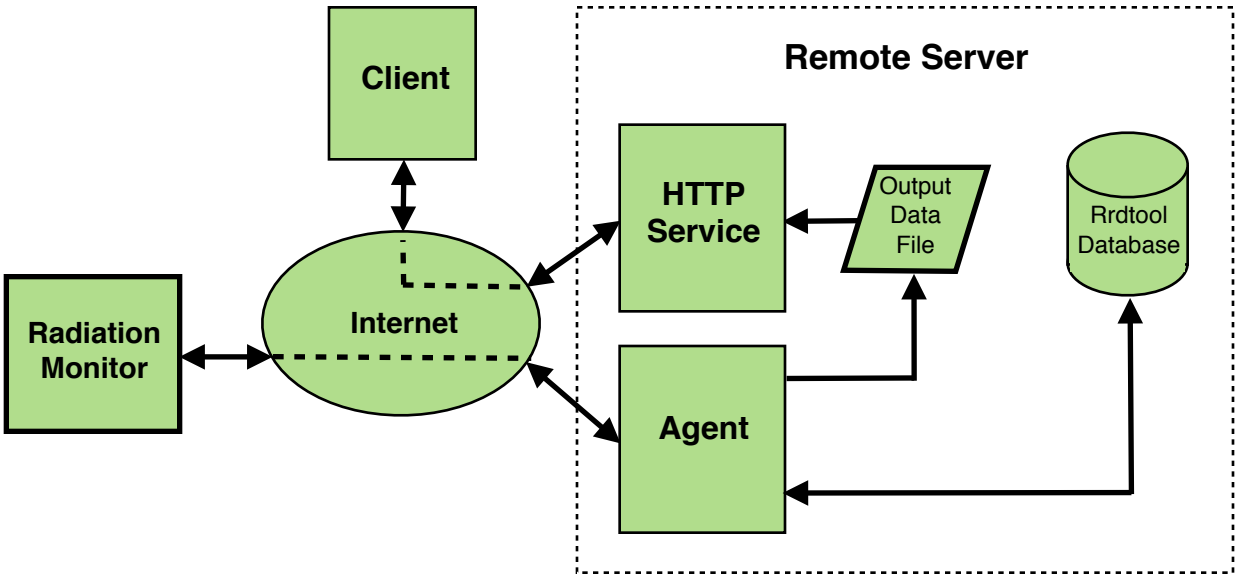

Figure 4. Block diagram of remote server software components showing data i/o, agent process, HTTP service, and database elements.

Events flow in the following manner. The agent periodically makes an HTTP request of the radiation monitor. The radiation monitor responds with a JSON formatted string containing the data. Below is a typical example of a data string received from the radiation monitor.

```
[{"radmon":"$,UTC=0:14:39 9/26/2015,CPS=0,CPM=19,uSv/hr=0.10,Mode=SLOW,#,"}]
```

The agent parses this string, writes the data to a rrdtool database, and, after formatting the weather data, the agent writes the data to the “Output Data File” for use by HTML documents. When a client requests a radiation monitor HTML document, Java Script embedded in the document reads the Output Data File and displays this data in an HTML document. Although not shown in figure 4, the agent, as mentioned earlier,

generates graphic charts for display in HTML documents. The graphic charts are stored as image files which Java Script in the HTML documents can display in the radiation monitor web page.

Building the Radiation Monitor

Parts List

Description	Quantity	Supplier	Part #
MightyOhm Geiger Counter	1	MightyOhm	0001
Arduino Uno micro-controller	1	Maker Shed	MKSP11
Arduino Ethernet Shield	1	Maker Shed	MKSP7

- Ordering information, hookup guides, and detail instructions for MightOhm are available at mightyohm.com/blog/products/geiger-counter/
- Ordering information for Arduino parts is available at www.makershed.com

Some additional hardware may be required, such as, a box to contain the Arduino parts and the MightyOhm Geiger counter. A small power supply (7 to 10 VDC) may be necessary if the Arduino's USB port will not be used for power.

Assembly Notes

1. Build the MightyOhm Geiger counter according to the manufacturer's instructions, substituting out the two resisters as directed in steps 2 and 3 below. The MightyOhm Geiger counter uses a 3 volt battery power supply and must be modified slightly in order to work with the Arduino Uno's 5 volt power. The following modifications will allow the Mighty Ohm Geiger counter board to operate using 5 volts supplied by the Uno instead of 3 volts supplied by battery. (See *MightyOhmSchematic.jpg* in the project docs folder.)
2. On the MightyOhm Geiger counter printed circuit board, change resistor R6 from a 330 Ohm to a 1K Ohm resistor.
3. On the MightOhm Geiger counter printed circuit board, change resistor R11 from a 100 Ohm to a 330 Ohm resistor.
4. Connect +5v from Arduino Ethernet shield to Geiger counter connector J6 pin 1.
5. Connect GND from Arduino Ethernet shield to Geiger counter connector J6 pin 3.
6. Connect Arduino pin D5 to Geiger counter connector J7 pin 5.
7. Mount the Arduino board and Geiger counter board in a suitable housing, such as a small plastic box. The housing should not be made of any material that shields radiation, such as lead.

This completes assembly of the radiation monitor.

Software Installation

1. Get the latest version of the Arduino IDE from https://www.arduino.cc/en/Main/Software. The IDE is available for Mac, Windows, and Linux operation systems.
2. Before compiling the radmon Arduino sketch, a time/date library must be installed. Download the time library from http://www.pjrc.com/teensy/td_libs_Time.html
3. When you run the Arduino IDE for the first time, the IDE creates a folder Arduino/sketch/libraries. Look for this folder in your home documents folder. Unzip the library and copy the Time folder to the libraries folder.
4. Using a USB cable connect the computer running the Arduino IDE to the Arduino Uno's USB port.
5. Follow the Arduino instructions for connecting the Uno to your computer and configuring the IDE for the Uno. Normally this means navigating to the Tools->Board menu item and selecting the Uno, and going to the Tools->Port menu item and selecting USB port to which the Uno is connected.
6. From the project Arduino folder, download the Arduino sketch named Radmon_v14.ino.
7. Open this sketch in the IDE.
8. Look for a line at the top of the sketch the looks like

   ```
   #define ETHERNET_MAC_ADDRESS 0x90, 0xA2, 0xDA, 0x0D, 0x84, 0xF6
   ```

   Change the hardcoded MAC address to the MAC address provided with your Ethernet shield.
9. Compile the sketch. If the Time library was installed properly, and everything else done correctly, the sketch should compile cleanly without errors.
10. Upload the compiled sketch to the Arduino Uno.

This completes the radiation monitor software installation. Before the radiation monitor can be connected to your network, the network configuration must be set up. Follow the instructions below to set up the radiation monitor's network interface.

Configuring the Radiation Monitor

1. The radiation monitor should already be connected to your computer from step 4 in the section "Software Installation". If not, then do step 4 above.
2. Close the Arduino IDE.
3. This step depends on what OS your computer is running. If you are running Microsoft Windows, you will probably have to use Putty, or something equivalent, to set up a terminal session with the radiation monitor. If you are running Linux or OS X, you can use a screen session. However you will need to tell screen which device the radiation monitor is connected to, and this will vary with the distribution. Normally look in the /dev folder for a device name with usb in it, or type the command '**ls /dev/tty.* I grep usbmodem**'. Then open a screen session by typing '**screen -S rad {dev path}**', where 'dev path' is the device path determined above.

4. If you have been successful connecting to the Uno, you will see a few lines of text showing the software version number and copyright notice.
5. Press the lower case ‘c’ character to enter command mode. A menu of command mode options will appear. Please note that the radiation monitor will not respond to HTTP requests while in command mode. You must exit command mode, as described below, for the radiation monitor to respond to HTTP requests.
6. Type ‘2’ and press enter. If you want to assign the radiation monitor a static IP address then enter it now. If you want to use a DHCP assigned address then simply press enter. See your network administrator for what IP address to use.
7. Type ‘3’ and press enter. If you want to enter a specific NTP server IP address, then enter it now. If you simply press enter, a default NTP server IP address will be used.
8. Type ‘1’ and press enter. Verify that the settings are as you want them. If they are not, the repeat steps 6 thru 8.
9. If the proposed settings are correct, type ‘6’ and press enter. At any time all changes can be discarded and the radiation monitor returned to normal mode by typing ‘5’ and pressing enter.

This completes configuring the radiation monitor.

Installing the Remote Server Software

Dependencies

- The remote server software should be installed on a recent Linux distribution such as Debian. (The author has successfully developed and installed the software on a Raspberry Pi running the Raspbian operating system.)
- Apache2 should be installed and configured to allow serving HTML documents from user’s public_html folder.
- Rrdtool should be installed - type **sudo apt-get install rrdtool**
- Python 2.7 usually comes pre-installed in virtually all Linux distributions. Type “python” at a command line prompt to verify Python has been installed.

Software Inventory

The software items that need to be installed on the remote server are

HTML folder:

- radmon.html
- index.html
- jQuery.min.js
- static/chalk.jpg
- static/eptile.gif

Agent folder:

- createRadmonRrd.py
- radmonAgent.py
- startwea
- stopwea

Installation

Note that the following installation procedure assumes that radmon HTML documents will reside in the user's public_html folder. Typically the full path name to this folder will be something like "/home/[user name]/public_html".

1. In public_html, use **mkdir** to create a folder "radmon" to contain the radmon HTML files.
2. From the html folder on this Github project site download all files and folders to the folder created in step 1.
3. The output data file (see figure 4) gets overwritten every 10 seconds and should not be stored on disk. Rather it is dynamic content that should be stored in the temporary file system resident in ram. Similarly the graphic image files get overwritten frequently and should also be stored in the temporary file system. Assure that the server /tmp folder gets mounted to the temporary file system. This can be done by adding the following line to the /etc/fstab file

   tmpfs /tmp tmpfs nodev,nosuid,size=50M 0 0

4. The HTTP service cannot freely access files and folders outside of the public_html folder. Therefore, in the public_html/radmon folder create a symbolic link to the temporary file system by running

   ln -s /tmp/radmon dynamic

   The HTTP service will look for dynamic content by following the "dynamic" link to the /tmp/weather folder.
5. If it does not already exist, use **mkdir** to create a folder named "bin" in the user home folder. For example, the full path name should look like "/home/[user]/bin'.
6. From the agent folder in this Github project site, download the files into the bin folder created in step 6. In most Linux installations the user's bash profile will automatically add the user's bin folder to the command search path. If such is the case, then the agent can be started up by simply typing **radmonAgent.py** followed by ENTER.
7. In the user home folder create a folder named "database". In the bin folder run the python script **createRadmonRrd.py**. Running this script creates an empty round robin database file where the agent will store radiation data as it arrives from the radiation monitor. This script should be run once and then kept in a secure place. Running it accidentally at some future date will result in *total loss of all previously stored data*.
8. In the user's home folder create a folder named "log". For connivence two scripts have been provided to make it easy to turn the agent on and off. The **startrad** script starts up the agent and causes all diagnositc output and error messages to be written to a log file in the log folder. The **stoprad** stops the agent from running.

This completes installation of the remote server software.

References and Resources

The following sources should be consulted before building the radiation monitor:

- Arduino Uno - www.arduino.cc/en/Main/ArduinoBoardUno
- Mighty Ohm Geiger Counter - mightyohm.com/blog/products/geiger-counter/

The following tutorials are useful for more in depth understanding of programming Arduino:

- C programming - www.gnu.org/software/gnu-c-manual/
- Arduino programming - www.arduino.cc/en/Tutorial/Foundations

The following tutorials are useful for more in depth understanding of the remote server software:

- Java Script - www.w3schools.com/js/default.asp
- jQuery - www.w3schools.com/jquery/default.asp
- PHP - www.w3schools.com/php/default.asp
- HTML - www.w3schools.com/html/default.asp
- Rrdtool - oss.oetiker.ch/rrdtool/
- Python - greenteapress.com/thinkpython/thinkpython.html