Australasian Physical & Engineering Sciences in Medicine (2019) 42:245–257
https://doi.org/10.1007/s13246-019-00723-y

TECHNICAL PAPER

Design, manufacture and deployment of a low-cost area radiation monitoring system using Raspberry Pi computers and open-source software

Steve Crossley[1] · **Ian Bucklow**[1] · **Jonathan Stafford**[1]

Received: 9 October 2018 / Accepted: 9 January 2019 / Published online: 14 February 2019

Abstract
Monitoring of background radiation levels in radiopharmaceutical laboratories is a key tool in minimising dose to workers. Retrofitting area monitoring systems in an existing laboratory can be disruptive and prohibitively expensive. We set out to develop a flexible low-cost area monitoring system utilising the power of inexpensive single board computers and open-source software. A complete system has been developed which includes local and remote real-time display with local warning, dose rate logging and automated plotting and backup of results from over 20 individual monitors connected via wifi.

Keywords Radiation safety · Area monitoring · Radiation measurement · Computing

Introduction

In areas where work is carried out using radioactive materials there is a possibility that inadequately shielded sources will raise the background radiation dose rate, resulting in unnecessary exposure of staff. A radiation area monitor can be used to detect and record the dose rate in a room or area. Systems are available commercially which link a number of monitors to provide area monitoring for a facility. These commercial systems are expensive. A quote for a system to be installed in our laboratories was of the order of $100k. Due to the expense, the number of locations monitored is usually limited, with many systems fixed at install. Work practices change over time, but the addition of extra monitoring locations requires additional expense and may not be possible if the system is no longer commercially supported.

Thanks to the decreasing cost of computing power and the shrinking physical size of computing systems, it is no longer always a requirement to produce custom circuits and chipsets running dedicated firmware to perform specialist computational tasks. It is possible to produce small devices running available operating systems which perform the necessary work in software. In an analogue dosimeter a relatively simple circuit converts the signal from a Geiger–Muller (GM) tube to a dose rate. If this work is done instead in software, the dose rate produced can be passed to a number of different processes running on the same hardware. Modifications to software are much simpler than hardware changes and can be made quickly, allowing for more rapid development.

A system has been developed which utilises a low cost, credit card sized, Wi-fi equipped computer (Raspberry Pi 3) with a small custom circuit board to produce a networked area radiation monitor. A number of these monitors can be connected over an existing network to a central server and any number of display stations to produce a complete area radiation monitoring system.

The monitors themselves, as well as measuring and logging the dose rate, act as simple web servers to enable display of the dose rate locally, if the monitor has a display screen, and remotely on any web browser connected to the same network. A central server collects the dose rate logs from all the monitors and automatically produces customisable summary plots which can be used to quickly review the results.

Any modern web browser connected to the same network can run a simple html page which will display the results from multiple (or all) monitors on the network, allowing for real time review of the area dose rates throughout the facility.

✉ Steve Crossley
steven.crossley@health.wa.gov.au

[1] Medical Technology & Physics, Sir Charles Gairdner Hospital, Nedlands, WA, Australia

The purpose of this technical note is to describe the development, manufacturing and deployment of an open-source area radiation monitoring system.

Methods and materials

Hardware

The Raspberry Pi 3B

A Raspberry Pi is an inexpensive, credit card sized single board computer originally developed by the Raspberry Pi Foundation for educational use. As of 2017, the latest version is the Raspberry Pi 3B (Fig. 1). It has a 1.2 GHz quad-core ARMv8 processor, 1 Gb of RAM, built in wifi and bluetooth and general purpose input/output (GPIO) pins to facilitate connections to sensors, display devices and other circuitry. A micro-SD card is used for data storage. Raspberry Pi's can run a number of operating systems [1]. The officially supported operating system is the Debian Linux based Raspbian [2]. As of October 2018, a Raspberry Pi 3B can be purchased in Australia for < $60 [3].

Radiation detection circuit

A custom circuit (Fig. 14 in "Appendix") was designed to provide power from a Raspberry Pi to a GM tube and to feed a clean output signal from the GM tube back to a GPIO pin on the Raspberry Pi. The circuit board was designed to have similar dimensions to and to sit directly above the Raspberry Pi. It can connect directly to the GPIO pins and be supported at the same fixation points (the yellow ringed holes in Fig. 1).

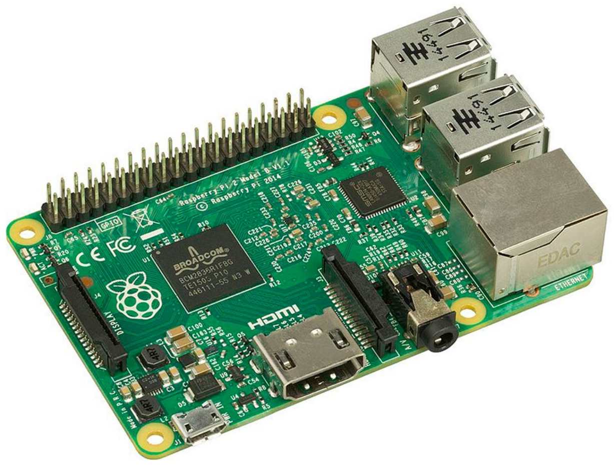


Fig. 1 Raspberry PI 3B showing the broadcom system on a chip (SOC) and the available GPIO ports (photo by evan amos)

A 5 V input is converted to the 500 V required by the energy compensated GM tube using a circuit based on a design by Maxim Integrated [4]. Radiation incident on the GM tube causes voltage drops on the 500 V rail. A DC blocking capacitor removes the 500 V leaving negative pulses on a 5 V signal. A pull-up resistor is used to keep the width of the pulses short to minimise the chances of incoming signals overlapping. The low voltage signal passes through a comparator which removes noise, leaving clean negative edge triggers which are fed to a GPIO input pin on the Raspberry Pi.

The board was designed using RS Component's free Design Spark software [5]. Printed circuit boards were ordered online and component mounting was completed in-house. There are services available to produce complete assembled circuit boards based on the design documentation produced during this project [6–8].

GM tubes

Two different GM tubes were tested, the 71,210 and the 71,412, both by LND Inc [9]. The 71,210 is physically larger and has been used in all but one of the monitors as it provides greater sensitivity and less fluctuation in the output at low dose rates. The less sensitive tube (LND 71,412) was used in RadPi007 and was intended for use inside a hot-cell where the dose rates are much higher. Either tube can be used with the same detection circuit. Compensation for the different sensitivity of the GM tubes is accomplished with a calibration factor in the software.

Display screen

The RadPi monitors can be connected to an optional display screen or can be used in 'headless' mode when a display is not needed. When 'headless' the monitors are controlled by remote ssh log in.

The raspberry Pi 3B has three possible ways of connecting a display. There is a standard HDMI output, a display serial interface (DSI) connector and the GPIO pins can be used to send video signals if the correct drivers are installed. The official Raspberry Pi display is a 7″ touchscreen which connects to the DSI.

After trialling a number of smaller screens the 7″ screen was found to be the most readable and easiest to set up. When using a Raspberry Pi as the Rad Pi Server a standard PC monitor was connected to the HDMI port.

Sound

A small amplification circuit and loudspeaker has been added to the RadPi to allow for audio warnings to be produced above certain dose rates.

Housing

A number of housings were trialled during the prototyping of the RadPi, including the cardboard box a Raspberry Pi 7″ screen was supplied in. The deployed design has settled on an aluminium case with plastic or slotted end caps. Holes are cut in the case to allow fitting of the screen and speaker (when applicable) and a micro-USB port. The USB port is for connecting the RadPi with the standard Raspberry Pi power supply. Either slotted or plastic end caps are used to improve the wifi signal strength compared to solid aluminium caps. The two designs going forward are a large slim case for the RadPi with a 7″ screen and a smaller case for the headless RadPi (Figs. 2, 3).

Software

Raspbian

Raspbian is a Debian Linux based operating system (OS) optimised for use on the Raspberry Pi [10]. The OS provides a desktop computing environment (PIXEL) with many pre-installed software development tools. A large array of open-source software is available which can be installed to make use of the Raspberry Pi hardware [11].

Converting a GM signal to a dose rate

A Python library (RPi.GPIO) [12] is available to convert signals from the GPIO pins on a Raspberry Pi into numerical values. A python script, geiger.py, shown in Fig. 13 in "Appendix", was written which utilises this Library to convert the signals from the GM tube into a dose rate.

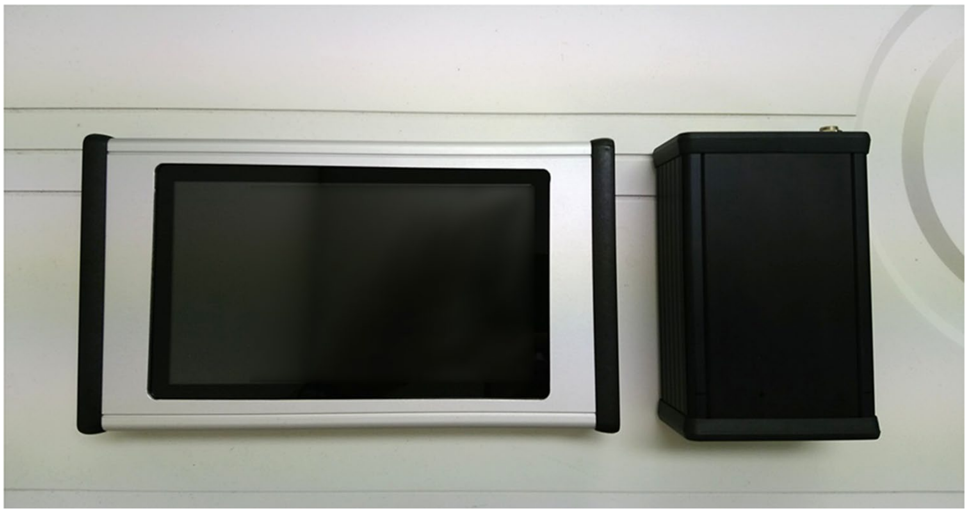


Fig. 2 The 7″ and headless RadPi models in cases

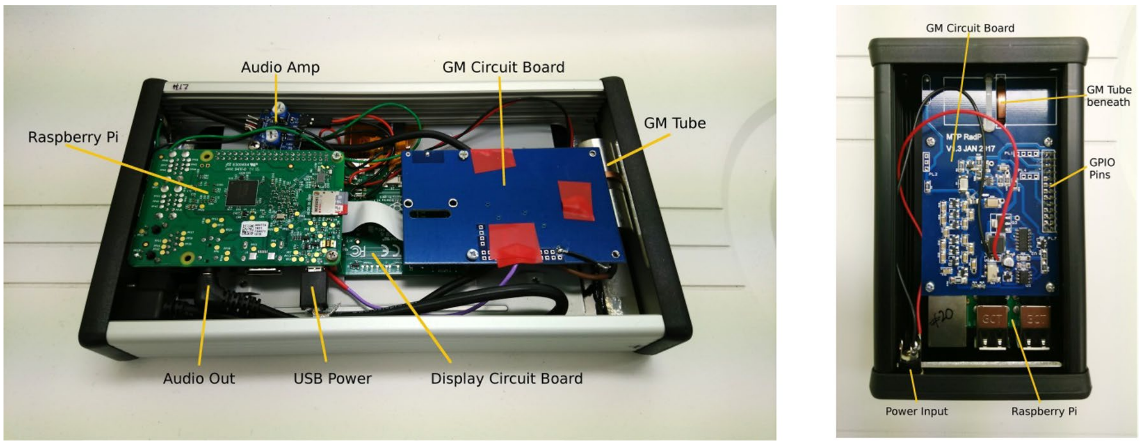


Fig. 3 Internals of 7″ and headless RadPi

At low dose rates the number of pulses detected each second is proportional to the dose rate. The second by second dose rate is stabilised using exponential smoothing. A calibration factor was established by comparing the dose rate measured by the system to that measured by a calibrated ion-chamber survey meter when exposed to a Cs-137 source at a range of source to detector distances (see Table 1; Fig. 6 in "Results" section). The calibration factor, along with other information specific to the individual monitor, is stored in a configuration file which is accessed by the geiger.py script.

Dose rate display and logging

The geiger.py script writes the calculated dose rate to two files. The first, the gauge file, is overwritten every second and contains the current dose rate. A line produced by the script containing the mean and peak dose rate for that minute is appended to a log file every minute.

The data in the gauge file is read by scripts using the justGage software [13] to produce a HTML page containing a graphical gauge displaying the dose rate and other information (Fig. 4) which is updated as the gauge file changes. A bottle [14] web-server running on the Raspberry Pi allows this HTML page to be viewed from a web browser. The web browser can be run on the monitor itself and displayed on a small attached screen, or it can be remote. Remote access allows for monitoring from anywhere on the local network. The hospital's wi-fi network is utilised to wirelessly link all the monitors and provide remote monitoring.

Custom html pages can be run on any PC on the network to display multiple area monitor gauges to provide an overview of the current dose rates throughout the facility.

The log file retains the history of the dose rates recorded by the monitor over time. It is synchronised across the network with a copy on a central server every 10 min. A small script moves this log file to a separate file each week and these weekly files are also synchronised to the server. These log files are stored in two locations as a backup against data corruption. The capacity of current SD cards means than many years of data can be stored on the monitors without concern for running out of space.

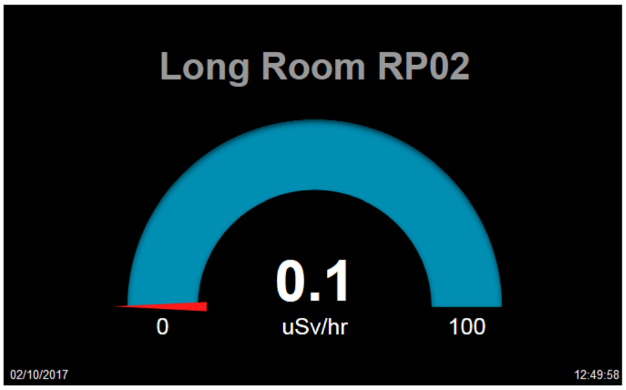


Fig. 4 Example web page created by justgage

Sound

The configuration file on each RadPi monitor contains a section called 'Tones' which specifies how the monitor produces audio. There is a count rate above which the monitor will produce a tone every second. As the count rate increases above this trigger level the frequency of the tone rises, sounding more urgent. The audio warning alerts users to the fact of an increased dose rate even if they are not looking at the monitor. The tones are produced by the play command of the sox software package which is called by the gieger.py script.

Remote server and plots

Another Raspberry Pi is used as a central server to store all the log files from the monitors. Any PC on the network running Linux could act as the server. Synchronisation of the logs between the server and monitors is handled by rsync [15] and happens every 10 min. Code updates made on the server are downloaded to each monitor whenever they are re-started. Scripts on the server run gnuplot [16] daily and weekly to produce plots of the dose rates over time. Each plot contains the maximum and mean dose rates against time for multiple monitors to allow for quick review. Example plots are shown in the "Results" section. As the plots are

Table 1 Example comparison measurements

Measured dose rates in μSv/h										
RadPi1	Rotem[a]	% Diff.	RadPi5	Rotem	% Diff.	RadPi9	Rotem	% Diff.	SmartIon[b]	% Diff.
630	670	−6	630	670	−6	630	660	−5	–	–
247	254	−3	260	254	2	185	186	−1	174	6
106	103	3	107	103	4	99	100	−1	101	−2
67	68	−1	68	68	0	66.5	67	1	67	−1
23.6	24.2	−2	24	24.2	−1	24.6	24.4	1	23.4	5

[a]A Rotem RAD-R200 GM tube survey meter (purchased in 2012)

[b]A Mini-Instruments 2100S SmartIon ion chamber survey meter (purchased 2003)

generated by gnuplot scripts from simple text logs they can be easily modified to include new monitors as they are added or to change the layout or included data as required.

Figure 5 shows the flow of data from the detected pulses to the various outputs of the RadPi system (logs, plots and real-time displays). Any number of remote browsers can request the web page from the bottle server of each monitor. The remote Firefox browser in Fig. 4 is displaying a HTML page containing nine iframes each being served by the bottle server on a different RadPi Monitor.

Placement of area monitors

Area monitors were placed in each room where radioisotopes are dispensed, measured or stored. Additional monitors were placed in publicly accessible areas close to high dose rate zones to confirm the adequacy of radiation shielding. Some monitors were placed close to dispensing areas within the local radiation shielding to improve radiation hygiene when using and storing radioactive sources. Placement is limited to areas where mains power and a useable wifi signal (or ethernet port) are available. A modified RadPi was made for the cyclotron bunker with the GM tube and detector circuit within the bunker and the Raspberry Pi and display mounted outside. A RadPi with a portable battery pack is possible but a standard Li battery pack used to charge a mobile phone struggles to deliver enough power (~2 W) and will run a RadPi with a display for a few hours at most.

Results

Calibration results

Each monitor was calibrated against a survey meter (either a Rotem RAD-R200 or SmartIon 2100S) which had been calibrated against a traceable source by an external agent. The RadPi and calibrated monitor were simultaneously irradiated by the same Cs-137 source at a range of distances to provide a range of dose rates. The Ion-chamber is significantly larger than the GM tubes in both the RadPi and the Rotem which made simultaneous irradiation at high dose rates unreliable so was only used for some tests. Cs-137 is the isotope most commonly used in the calibration of dosimeters. Energy compensated GM tubes were chosen to minimise the variation in sensitivity with photon energy (Fig. 6).

Maximum dose rate

It takes a finite time for the voltage in the detection circuit to return to 5 V after detection of a gamma photon. As the dose rate increases the probability of pulses overlapping increases. If the pulses overlap the circuit cannot detect the voltage drop and so the count is lost. This causes a decrease in detector efficiency as dose rates increase. During development it was noticed that the efficiency drops noticeably at dose rates that are within the range of expected values for a radiopharmaceutical production facility. The pull-up resistor was changed to return the voltage on the input to 5 V more quickly after each pulse. The shorter detection events

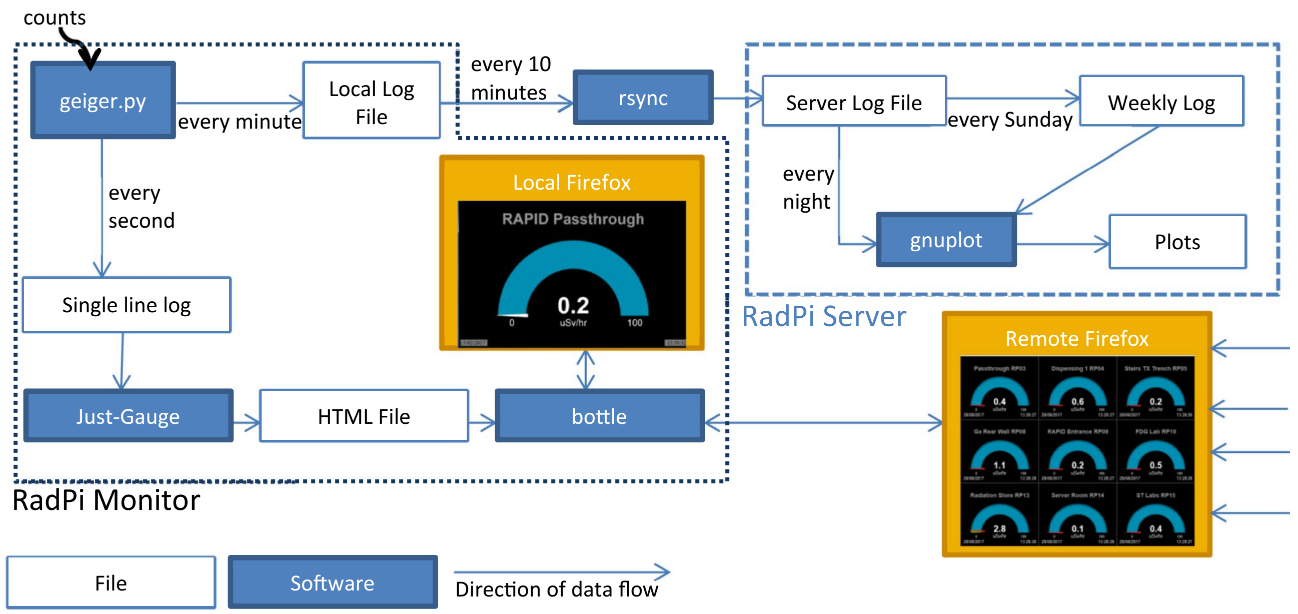
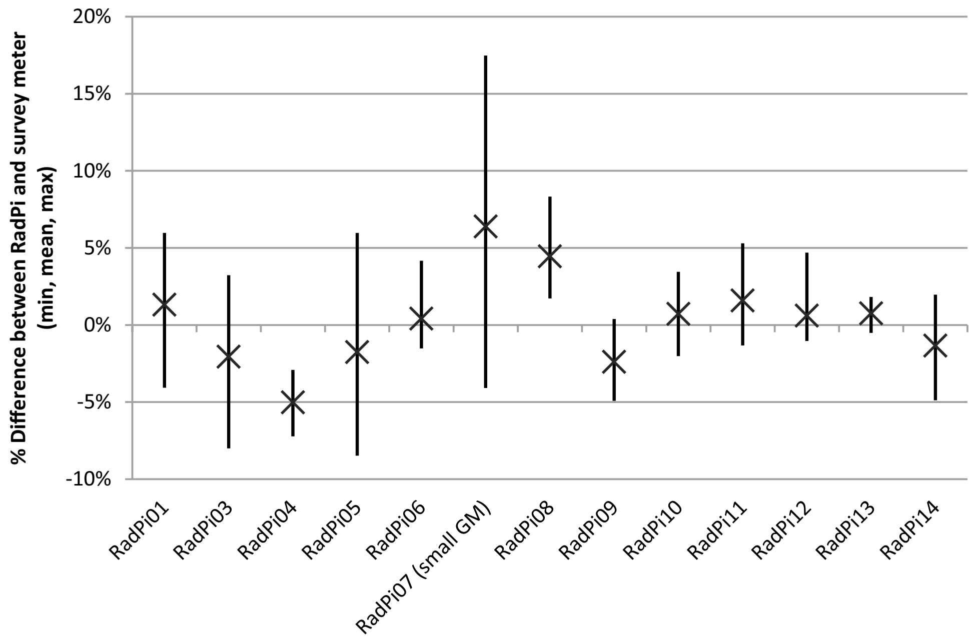


Fig. 5 System overview

Fig. 6 Differences between RadPis and survey meters



have a reduced chance of overlap in the incoming signal, thereby reducing the dead-time. After this adjustment the maximum dose rate which produced a linear response was around 1500 μSv/h for the monitors with the LND 71210 tube and about 5000 μSv/h for RadPi007 with the smaller LND 71412 GM tube.

Deviation from linearity is minimal after these thresholds, and the signal is not saturated until much higher dose rates. Dead time correction could be applied to increase the maximum measureable dose rate, though this is currently not required as the maximum measurable dose rates are already significantly higher than should be experienced by an area monitor.

Dose rate live displays

Figure 4 shows a screenshot from an operating RadPi monitor. Figure 7 is a screenshot from a HTML page made up of iframes displaying the web pages of 15 monitors simultaneously, including one inside a hot-cell (RadPi7 HRtube).

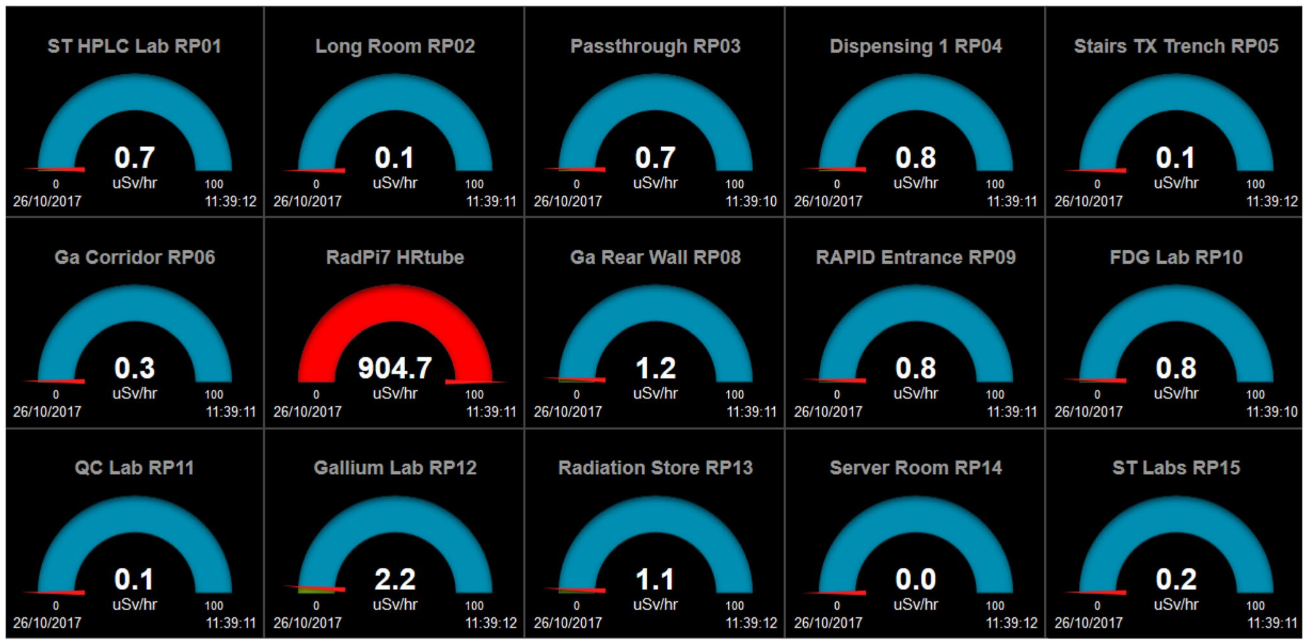


Fig. 7 Multiple monitor web page screenshot

Figure 7 is a screenshot of a web page which is displayed on a monitor inside the laboratory entrance corridor. It displays the gauges from all the area monitors in the labs and their locations. The map is simply a static image included in the web page (Fig. 8).

Dose rate logs

An excerpt from one of the log files is shown here. A weekly log file is typically < 600 kB and contains around 10,000 records (Fig. 9).

Dose rate plots

Example plots produced by gnuplot from log files (Figs. 10, 11).

```
RadPi004 : 2017-07-01 10:22:00 mean= 0.3 max= 0.5 uSv/hr
RadPi004 : 2017-07-01 10:23:00 mean= 0.3 max= 0.6 uSv/hr
RadPi004 : 2017-07-01 10:24:00 mean= 0.2 max= 0.5 uSv/hr
RadPi004 : 2017-07-01 10:25:00 mean= 0.5 max= 0.8 uSv/hr
RadPi004 : 2017-07-01 10:26:00 mean= 0.3 max= 0.6 uSv/hr
RadPi004 : 2017-07-01 10:27:00 mean= 0.3 max= 0.5 uSv/hr
RadPi004 : 2017-07-01 10:28:00 mean= 0.2 max= 0.5 uSv/hr
RadPi004 : 2017-07-01 10:29:00 mean= 0.2 max= 0.6 uSv/hr
RadPi004 : 2017-07-01 10:30:00 mean= 0.2 max= 0.5 uSv/hr
RadPi004 : 2017-07-01 10:31:00 mean= 0.2 max= 0.5 uSv/hr
RadPi004 : 2017-07-01 10:32:00 mean= 0.3 max= 0.5 uSv/hr
RadPi004 : 2017-07-01 10:33:00 mean= 0.2 max= 0.5 uSv/hr
RadPi004 : 2017-07-01 10:34:00 mean= 0.3 max= 0.5 uSv/hr
RadPi004 : 2017-07-01 10:35:00 mean= 0.3 max= 0.6 uSv/hr
RadPi004 : 2017-07-01 10:36:00 mean= 0.3 max= 0.6 uSv/hr
```

Fig. 9 Section of a log file showing records for each minute

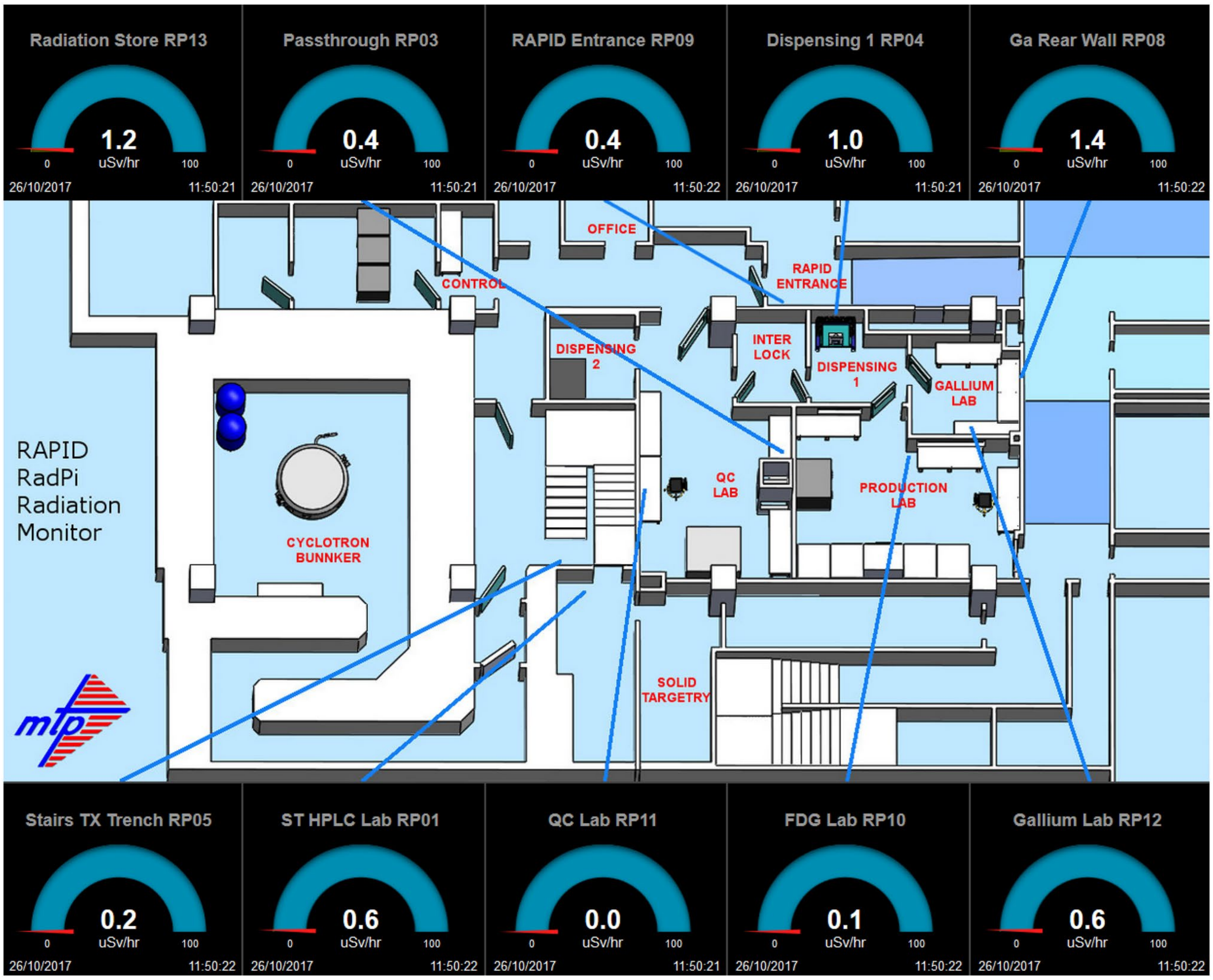


Fig. 8 Multiple monitor page with locations, displayed on a monitor near the entrance of the laboratory

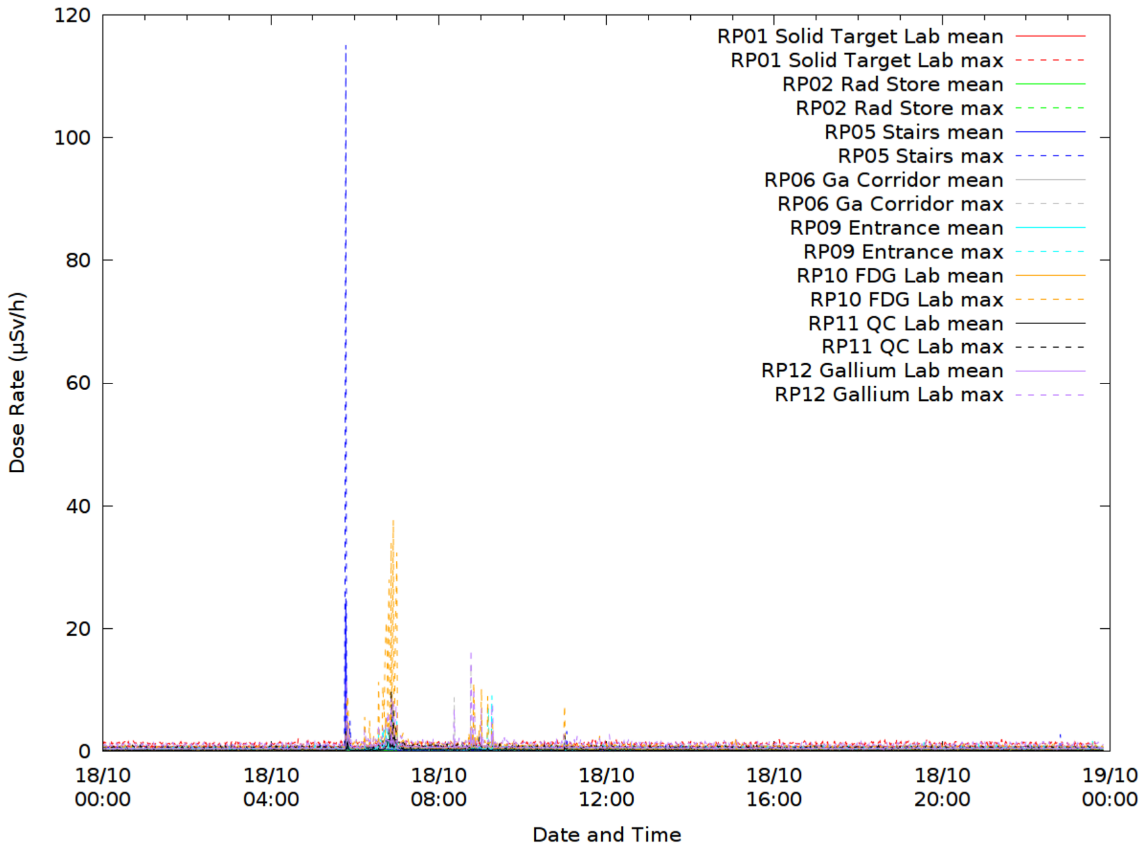


Fig. 10 Daily plot for area monitors showing transient spikes in dose rate

Discussion

Over time, work practices in a radiopharmaceutical lab change. These changes can be to the number, type and activity of radioactive materials being produced and used, and to the locations within the lab which are utilised. It is important for staff to always be aware of the dose rates in and around their work locations and for medical physicists to be able to monitor and review dose rates in areas where large activities of radioisotopes are produced and used. An area monitoring system which can be expanded with more units and where the units can be easily moved to new locations can assist in maintaining an adequate radiation safety environment, while coping with changes in work practices. Reporting requirements may also change over time and the ability to modify outputs and add additional functionality is much easier and less costly for a system when you have the full specifications and access to the software.

Development and deployment of this system has highlighted specific issues related to radiation shielding around benchtop radiopharmaceutical areas which could have been missed and could have resulted in unrecorded exposures to staff and breaches of regulatory requirements. Changes to work practices have been put in place as a direct response to information provided by the RadPi system. Work was moved from one location to another within the laboratory and additional shielding was installed to prevent dose rates exceeding acceptable levels in the publicly accessible areas surrounding the laboratory.

Proprietary area monitoring systems are expensive (in the order of $100k for a multi-room system with logging) which tends to limit the number of locations monitored. Proprietary systems can also become obsolete. Once a particular model of detector is no longer available the system cannot be expanded and repairs may not be possible. Due to the unknown communications protocols used, old proprietary systems can be difficult to interface with new software and hardware systems, necessitating a complete replacement of all detectors and interfaces.

A disadvantage of a build-your-own solution can be a lack of commercial support options. Large open source software projects have a large on-line community to offer help, but developers of small projects often move on to other things and have no obligation to assisting in updating or fixing their software. In the case of the RadPi the custom software is relatively simple and can be understood by a novice and the other software components (Raspbian, rsync etc.) are large projects which have large communities behind them. If individual software components become outdated there will be replacement projects which could be used to replace them. Thorough documentation of open-source systems can prevent obsolescence by allowing others to continue developing

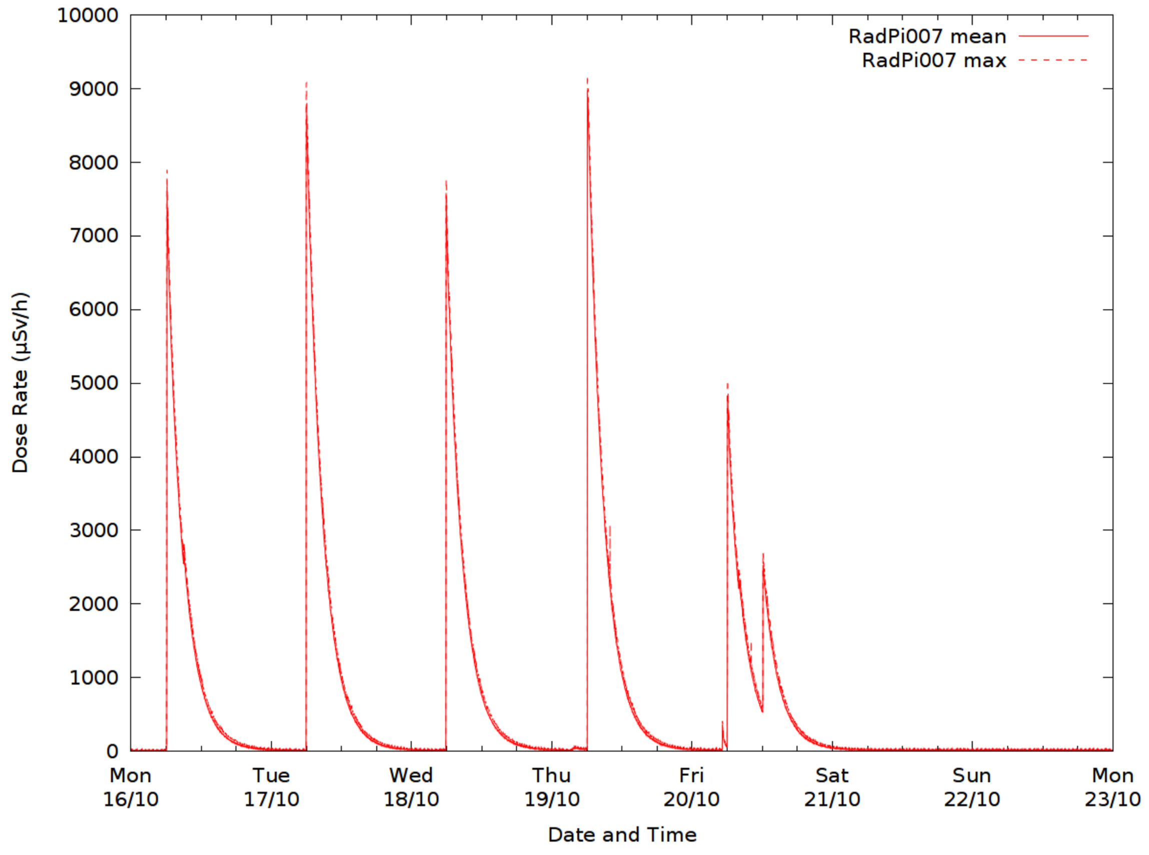


Fig. 11 Weekly plot of dose rates inside hot-cell showing decay of stored isotope

the system without input from the original designers. The authors have written a guide document to explain the construction, programming and operation of the RadPi system in addition to making the plans and software available.

With an area monitoring system based on open technologies the software used to generate results is available and can be customised to produce results in the format required by the user. As all communications use standard, well-documented interfaces and protocols, an open system can easily be expanded by adding new units. New units need not follow the same hardware design if they are customisable enough to produce compatible outputs or existing outputs can be modified to incorporate those of the new model. This greatly reduces the risk of obsolescence.

The design produced in this project is suitable for use in any area where there is potential for high dose rates of gamma radiation. It is not recommended for accurate measurement of background radiation levels as we have no method of calibrating down at that level. Calibration within 20% of a reference standard is usually deemed acceptable for radiation detection equipment. All the monitors produced to date have attained that across a range of dose rates.

Due to its open nature, this design is intended to be used as the basis for further development. Additional or alternative sensors could be added to provide more information such as temperature, humidity or location. Two projects are already underway; one using a modified design to remotely monitor and log pressure in laboratory gas cylinders, the other a hand contamination monitor using thin window pancake GM tubes. It is our hope that by making the design publicly available it will be further developed by interested parties.

Funding No external funding was applied for or used in the fulfilment of this project.

Compliance with ethical standards

Conflict of interest All the authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Appendix

See Figs. 12, 13, 14.

Fig. 12 geiger.py code

```
#!/usr/bin/python

import RPi.GPIO as GPIO
import time
import datetime
import paramiko
import socket
import os
import configparser

deviceName = socket.gethostname()
filename = deviceName + ".log"
settings = configparser.ConfigParser()
configFile = '/home/pi/geiger_log/' + deviceName + '.conf'
settings.read(configFile)
geigerInput = int(settings.get('Calibration', 'geigerInput'))
averageOver = float(settings.get('Calibration', 'averageOver'))
convFactor = float(settings.get('Calibration', 'convFactor')) # Converts
CPS to uSv/h
if "Tones" in settings.sections():
    minCount = int(settings.get('Tones', 'minCount'))
    minFreq = int(settings.get('Tones', 'minFreq'))
    toneScale = float(settings.get('Tones', 'toneScale'))
else:
    toneScale = 0
tick = 0.00
avgCount = 0.00
doseRate = 0.00
rateArray = [0] * 60 # array to collect all results in a minute
sec = 0

# designate log file.
with open('/home/pi/geiger_log/' + filename, 'a') as logfile_local:
    # set numbering for GPIO pins
    GPIO.setmode(GPIO.BOARD)
    #Setup pin 12 to read on voltage drop
    GPIO.setup(geigerInput, GPIO.IN, pull_up_down=GPIO.PUD_UP)

    def addCount(channel):
        global tick
        tick += 1
        return

    GPIO.add_event_detect(geigerInput, GPIO.FALLING, callback=addCount)

    while True:
        try:
            time.sleep(1 - time.time() % 1)
            # Using Exponential Moving Average
            if avgCount == 0:
                avgCount = tick
            else:
                avgCount = ((avgCount * (averageOver - 1)) / averageOver) +
(tick / averageOver)
            # Reset Count immediately to reduce error
            tick = 0
            doseRate = round((avgCount * convFactor), 1)
            ts = time.time() # current time from epoch
            st = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d
%H:%M:%S')
            sec = int(st[-2:]) # get seconds from timestamp
            rateArray[sec] = doseRate
            if toneScale > 0:
                tone = doseRate * toneScale + minFreq
                if tone > minCount:
                    os.system("play -n -q -b 16 synth 0.2 sine " +
str(tone) + " fade 0.05 0 0.05 gain -5")
            if sec == 0 :
                text = deviceName + ' : ' + st + ' mean= ' +
str(round((sum(rateArray)/len(rateArray)), 1)) + ' max= ' +
str(max(rateArray)) + ' uSv/hr'
                logfile_local.write(text + "\n")
                logfile_local.flush()
                rateArray = [0] * 60
            # Write gauge file
            with open('/home/pi/geiger_gauge/' + deviceName + '_gauge.log',
'w') as gaugefile:
                #gaugefile.write('\n')

gaugefile.write(datetime.datetime.fromtimestamp(ts).strftime('%d/%m/%Y
%H:%M:%S') + ' ' + str(doseRate) + '\n')
                gaugefile.flush()
        except KeyboardInterrupt:
            logfile_local.close()
            gaugefile.close()
            GPIO.cleanup()     # clean up GPIO on CTRL+Z exit
    logfile_local.close()
    gaugefile.close()
    GPIO.cleanup()
```

Fig. 13 Example gnuplot script

```
set terminal pngcairo size 1200,900 enhanced font 'Verdana,16'
set output './DailyPlots/Daily_Area_' . `date +%Y%m%d` . '.png'
set pointsize 2
set termoption dash
set key inside right top
set xlabel "Date and Time"
set ylabel "Dose Rate ({/Symbol m}Sv/h)"
set yrange [0:100 < *]
set xdata time
set timefmt "%Y-%m-%d %H:%M:%S"
daystart = "`date -d 'today 00:00:00' +%Y-%m-%d`" . " 00:00:00"
dayend = "`date -d 'tomorrow 00:00:00' +%Y-%m-%d`" . " 00:00:00"
set format x "%d/%m\n%H:%M"
set xrange [daystart:dayend]
set linestyle 1 lt 1
set linestyle 2 lt 2
plot '< tail -n8640 ./geiger_log/RadPi001.log' using 3:6 with lines ls 1 lc
"red" title "RP01 Solid Target Lab mean", \
     '< tail -n8640 ./geiger_log/RadPi001.log' using 3:8 with lines ls 2 lc
"red" title "RP01 Solid Target Lab max", \
     '< tail -n8640 ./geiger_log/RadPi002.log' using 3:6 with lines ls 1 lc
"green" title "RP02 Rad Store mean", \
     '< tail -n8640 ./geiger_log/RadPi002.log' using 3:8 with lines ls 2 lc
"green" title "RP02 Rad Store max", \
     '< tail -n8640 ./geiger_log/RadPi005.log' using 3:6 with lines ls 1 lc
"blue" title "RP05 Stairs mean", \
     '< tail -n8640 ./geiger_log/RadPi005.log' using 3:8 with lines ls 2 lc
"blue" title "RP05 Stairs max", \
     '< tail -n8640 ./geiger_log/RadPi006.log' using 3:6 with lines ls 1 lc
"grey" title "RP06 Ga Corridor mean", \
     '< tail -n8640 ./geiger_log/RadPi006.log' using 3:8 with lines ls 2 lc
"grey" title "RP06 Ga Corridor max", \
     '< tail -n8640 ./geiger_log/RadPi009.log' using 3:6 with lines ls 1 lc
"cyan" title "RP09 Entrance mean", \
     '< tail -n8640 ./geiger_log/RadPi009.log' using 3:8 with lines ls 2 lc
"cyan" title "RP09 Entrance max", \
     '< tail -n8640 ./geiger_log/RadPi010.log' using 3:6 with lines ls 1 lc
"orange" title "RP10 FDG Lab mean", \
     '< tail -n8640 ./geiger_log/RadPi010.log' using 3:8 with lines ls 2 lc
"orange" title "RP10 FDG Lab max", \
     '< tail -n8640 ./geiger_log/RadPi011.log' using 3:6 with lines ls 1 lc
"black" title "RP11 QC Lab mean", \
     '< tail -n8640 ./geiger_log/RadPi011.log' using 3:8 with lines ls 2 lc
"black" title "RP11 QC Lab max", \
     '< tail -n8640 ./geiger_log/RadPi012.log' using 3:6 with lines ls 1 lc
"purple" title "RP12 Gallium Lab mean", \
     '< tail -n8640 ./geiger_log/RadPi012.log' using 3:8 with lines ls 2 lc
"purple" title "RP12 Gallium Lab max"
```

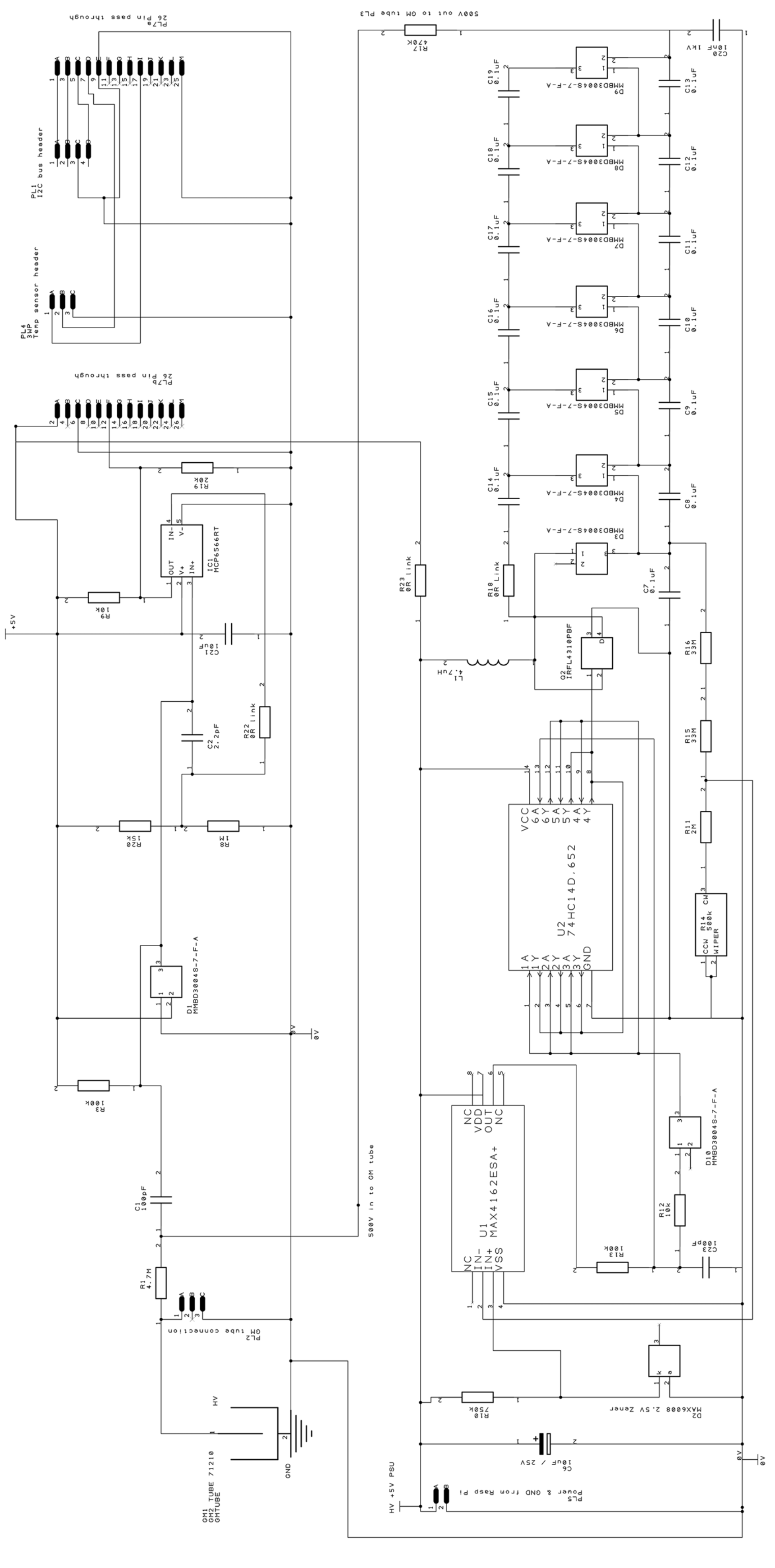


Fig. 14 Circuit diagram

References

1. Foundation RP, "Downloads," (2017) [Online]. https://www.raspberrypi.org/downloads/. Accessed 2 Oct 2017
2. Foundation RP, Page "RaspbianD," (2017) [Online]. https://www.raspberrypi.org/downloads/raspbian/. Accessed 2 Oct 2017
3. Components RS, Components "RS," (2018) [Online]. http://au.rs-online.com/web/p/processor-microcontroller-development-kits/8968660/. Accessed 1 Oct 2018
4. Integrated M, "APPLICATION NOTE 3757: Bias Supply Powers Low-Power Geiger-Mueller Tube," 22 03 2006. [Online]. https://www.maximintegrated.com/en/app-notes/index.mvp/id/3757. Accessed 2 Oct 2018
5. Components RS, "DesignSpark Home," [Online]. https://www.rs-online.com/designspark/home. Accessed 1 Oct 2018
6. Circuits S, "Screaming Circuits," (2017) [Online]. https://www.screamingcircuits.com/. Accessed 26 Oct 2017
7. Circuits A, "Advanced Circuits 4PCB," (2017) [Online]. http://www.4pcb.com/. Accessed 26 Oct 2017
8. Digital AC, Digital "AC," (2017) [Online]. http://www.acddigital.com.au/. Accessed 26 Oct 2017
9. Inc LND, Inc "LND," [Online]. https://www.lndinc.com/. Accessed 01 Oct 2018
10. Thompson M, Green P, "Raspbian Homepage," (2017) [Online]. https://www.raspbian.org/FrontPage. Accessed 02 Oct 2017
11. Thompson M, Green P, "Raspbian Repository," (2017) [Online]. https://www.raspbian.org/RaspbianRepository. Accessed 02 Oct 2017
12. Croston B, "RPi.GPIO," [Online]. https://pypi.python.org/pypi/RPi.GPIO. Accessed 02 Oct 2017
13. Djuricic B, "justGage," (2017) [Online]. http://justgage.com/. Accessed 02 Oct 2017
14. Hellkamp M, "Bottle: Python Web Framework," (2017) [Online]. https://bottlepy.org/docs/dev/. Accessed 25 Oct 2017
15. Davison W, "rsync homepage," [Online]. https://rsync.samba.org/. Accessed 09 Oct 2018
16. Williams T, Kelley C, "Gnuplot Homepage," [Online]. http://www.gnuplot.info/. Accessed 09 Oct 2018