ARL-TR-9762 ● SEP 2023

Semiconductor Energy Converter Calibration Laboratory for Betavoltaic Radioisotopes

by Jason Barbier, Nusrat Sarwahrdy, Dylan Flater, and Marc Litz

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9762 ● SEP 2023

Semiconductor Energy Converter Calibration Laboratory for Betavoltaic Radioisotopes

Jason Barbier, Nusrat Sarwahrdy, and Dylan Flater
University of Maryland

Marc Litz
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
September 2023	Technical Report	06/06/2022	08/12/2022

4. TITLE AND SUBTITLE
Semiconductor Energy Converter Calibration Laboratory for Betavoltaic Radioisotopes

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Jason Barbier, Nusrat Sarwahrdy, Dylan Flater, and Marc Litz

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-EC Adelphi, MD 20783	ARL-TR-9762

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID ID: Marc Litz, 0000-0003-0694-4152

14. ABSTRACT
Nuclear emissions from decaying radioisotopes (RIs) can be converted into power using specialized semiconductor cells. This technology has the advantage of power generation lifetime varying from decades to several centuries, which is useful in battle, logistics, and tamper-proof equipment. In this report, beta emitters Promethium-147 and Nickel-63 were used to irradiate both InGaP- and SiC-based cells. The expected output power of each device is calculated analytically by considering geometry and self-attenuation losses that reduce the number of particles that can reach the target active area. I-V curves measured the target device response to determine power, fill factor, and efficiency. Monte Carlo nuclear particle modeling demonstrated the expected flux of particles on the specified target. The RI source and target geometries have been measured and modeled, and both compare well. This capability can now be used in future semiconductor energy conversion efforts investigating ultra-wideband materials.

15. SUBJECT TERMS
Energy Sciences, betavoltaics, energy conversion, isotope spectra, semiconductor calibration, Monte Carlo nuclear particle transport code, MCNP

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	56

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Marc Litz	(301) 394-5556

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

Betavoltaic devices offer long-lasting, low-level power systems using mini-limited activity (<1 Curie [Ci]) of radioisotope (RI). This report discusses the experiments necessary to establish a calibration facility as well as interesting comparisons of measurements across combinations of ^{63}Ni and ^{147}Pm foils irradiating indium gallium phosphide (InGaP) and silicon carbide (SiC) energy converters (ECs).

1.1 US Army Goal

Using the long life and high energy density of RIs can enable persistent power sources for unattended sensors, logistics, and communications. Nuclear batteries allow a compact form of energy that does not need to be replaced within its half-life (2.62 years[1] for ^{147}Pm and 100 years[1] for ^{63}Ni). Implementing RIs has the advantage of surpassing chemical energy storage in energy density (Fig. 1).

Safety is an ongoing concern with this technology. Using several Curies of material (i.e., ~10 kCi of ^{238}Pu used in a satellite radio thermoelectric generator[2]) would be necessary to achieve the output power necessary to power large equipment. Without proper shielding and administrative controls this configuration could pose a hazard to Soldiers in the equipment's proximity. In contrast, low power used by betavoltaic ECs can meet the needs for small, lasting circuits without the need to carry or dispose of more than a couple millicuries of radioactive material.

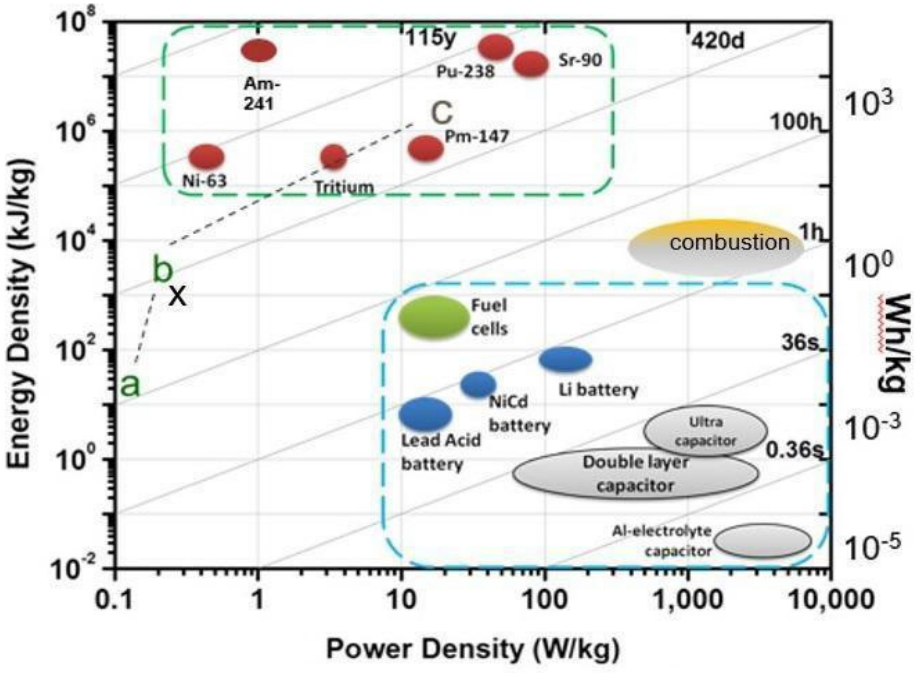

Fig. 1 Ragone plot[3] showing that RIs are at least 3 orders of magnitude greater energy density than chemical storage. The power density of chemical storage is typically greater than that of decaying isotopes.

1.2 Problem Statement

This report describes the new capability for the US Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory's (ARL's) Building 504 facility to calibrate semiconductor ECs under the full spectrum of beta decay from both ^{147}Pm and ^{63}Ni RIs. The beta energy spectrum is known, and a device efficiency for any EC material can be measured. The photovoltaic InGaP and betavoltaic SiC are included for this test with the plan to use gallium nitride (GaN), aluminum gallium nitride (AlGaN), and diamond semiconductors in future evaluations. This method can also be used for other beta isotopes in a similar configuration.

In Section 2 we describe semiconductor materials, their characteristics, and examples of device structures. In Section 3 we describe the experimental tools, setup, and measuring process including a description of the RI sources. In Section 4 we describe the modeling and simulation performed to calculate the details of RI coupling to the ECs and beta penetration range in materials. In Section 5 we describe the measurement results and compare these to modeling and expected values.

2. Background

RIs are the unstable form of ordinary matter. Each species will undergo one of five methods of decay to reach a more stable atomic configuration: alpha, beta, gamma ray, electron capture, and spontaneous fission. This experiment takes a closer look at capturing beta emissions. Power outputs are determined by factors in the RI and device. RI-specific contributions include the average energy carried by the particles, activity, and amount of substance. Device factors include intrinsic efficiency and coupling (geometric matching with and distance to the RI source).

2.1 Semiconductor Characteristics Useful in Betavoltaics

Understanding the crystal structure visually can aid in selecting desired properties. For semiconductors' ECs, three structures are of interest (although more options exist): zinc (Zn) blende, wurtzite, and diamond. Zn blende[4] (Fig. 2 [left]) refers to sphalerite in its common state in nature. When heated, the crystal becomes wurtzite (Fig. 2 [center]),[4] a polymorph with strong phosphorescent properties.

Diamond[5] (Fig. 2 [right]) has a tetrahedral structure and with impurities added through chemical vapor deposition can change from an insulator to semiconductor that can reach a notably larger bandgap (BG) than Zn crystals.

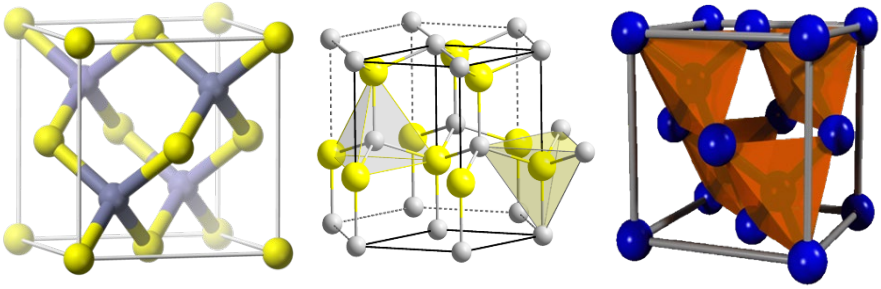

Fig. 2 Inorganic crystal structures including Zn blende (left), wurtzite (center), and diamond (right)

Semiconductors are designed in various diode geometries (i.e., Schottky, positive–intrinsic–negative [PIN], and bi-polar junction transistor [BJT]) that permit power or efficiency optimization depending on the desired application. Table 1 shows some qualities of semiconductors, such as BG, lattice constant, and displacement energy. These properties impact the radiation tolerance of the semiconductor material. Small lattice constant is desired (i.e., SiC and AlGaN), and large BG (i.e., ZnS, AlGaN, and diamond) and displacement energy make for radiation-tolerant semiconductor materials.

The larger the BG, the more energy can be generated per transition. The thermal conductivity is largest in diamond and permits use of higher currents (or RI activity). Typically, indirect BG materials have a larger diffusion length, which allows for free electrons created by electron hole pairs outside the depletion to drift into the depletion region and be collected for energy conversion.

Table 1 Characteristics of commonly used semiconductor ECs[6]: AlGaN density,[7] AlGaN thermal conductivity,[8] and InGaP and ZnS thermal conductivity[9]

	Si	SiC4H	GaN	AlGaN	Diamond	InGaP	ZnS	CdTe
Crystal Structure	diamond	wurtzite	wurtzite	wurtzite	wurtzite	Zn blende	Zn blende	Zn blende
BG structure	indirect	indirect	direct	direct	indirect	direct	direct	direct
Density [g/cm^3]	2.33	3.2	6.1	6.1	3.515	4.81	4.1	5.85
Lattice Constant [Å]	5.42	3.18	3.3	3.17	3.57	5.65	3.82	6.48
BandGap [eV]	1.14	3.23	3.39	5	5.5	1.79	3.91	1.45
Breakdown [MV/cm]	0.3	3-5	5	3.2	1-10		2	-
Displacement [eV]	12.9	38	39	15	35	4	15	250
Thermal conduct [W/cmK]	1.3	3.7	1.4	19	6-20	1.4	16.7	0.06

Understanding the distance particles travel (penetration range) in materials is also essential in selecting the best (most efficient) EC (penetration range values are shown in Table 2). For example, in SiC the range of ^{147}Pm is five to six times larger than that of ^{63}Ni because the beta particle energy maximum is 225 keV compared to 62 keV for ^{63}Ni. For each RI the penetration range is larger for SiC in comparison to InGaP, which is denser (4.81 g/cc) than SiC (3.2 g/cc), and particles penetrate less distance in higher density materials.

Table 2 Penetration range in micrometers (distance where 90% of isotope energy is deposited)[10]

Material	InGaP	SiC
^{63}Ni	4.1	4.4
^{147}Pm	19.3	24.8

2.2 Energy Conversion in Semiconductors

The average energy lost to electron-hole pair (EHP) creation is accounted for by the material's intrinsic BG, optical phonon losses, and residual kinetic energy. Experimentation shows the EHP model is subject to three quantitative tests[11]: 1) Fano-factor variations that reflect relative weight of phonon lost, 2) BG dependence suggests optical phonon losses remain constant, and 3) wavelength comparisons with the spectral distribution of electrons point to average impacts of a certain energy.

The BG of semiconductors determines the maximum voltage possible in the circuit. The gap is based on the difference of electron energy from the valence to conduction level. The BG energy differs for every compound. For InGaP this energy difference is 1.1 V, but the difference is larger for SiC at 2.0 V.

The semiconductor diffusion length in meters is defined as the distance charge carriers move between the creation and device terminals. It is calculated by Eq. 1, where D is diffusivity and τ is material lifetime. Generally, the larger the doping, the shorter the diffusion length becomes due to a higher recombination rate.[12]

$$L = \sqrt{D\tau}\,. \tag{1}$$

2.3 Betavoltaic Device Design

The intrinsic layer[13] properties (thickness and dopant levels) determine the depletion region. The configuration shown in Fig. 3 shows a layer of RI directly above the semiconductor for simplicity. Future designs will include surrounding the source with semiconductors to capture the emissions from both sides of the RI layer, maximizing electrical power output from the RI. Matching penetration range ensures the maximum number of beta particles reach the intrinsic layer, from which the free electrons created can be captured.

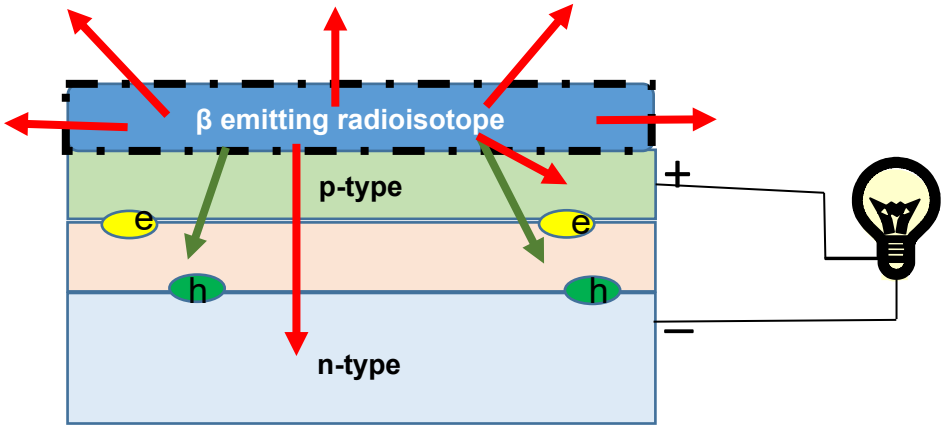

Fig. 3 Betas emitted from RI generate EHPs in an intrinsic layer that releases current to load

3. Experimental Setup

To establish a baseline and ensure repeatable results, the ECs are tested under each RI in spaced positions outward. The electrical output power is recorded as the distance from source changes. This position of maximum power output confirms the location where the target EC is centered within the RI foil region. This is the optimal location to place the EC for future tests.

3.1 Experimental Approach

The components of the experiment are shown in Fig. 4 and include the RI foil and the EC exposed below. These components are operated within a pillbox-sized radiation shield. The general setup involves sliding a semiconductor chip (EC) under the RI, housed in a shielded pillbox. The EC is placed and held in a cartridge/drawer (Fig. 5) that slides through the pillbox. The EC is connected electrically to the data acquisition (DAQ) station for recording current response as a function of applied voltage.

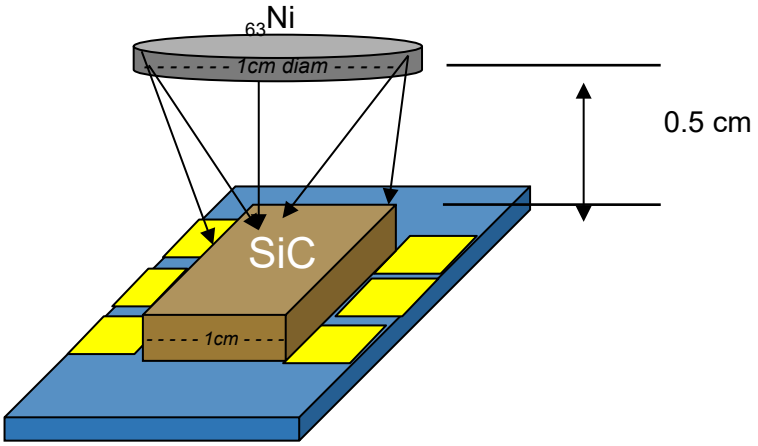

Fig. 4 Isotropic radiation of a ^{63}Ni RI foil subtended by area of the SiC EC

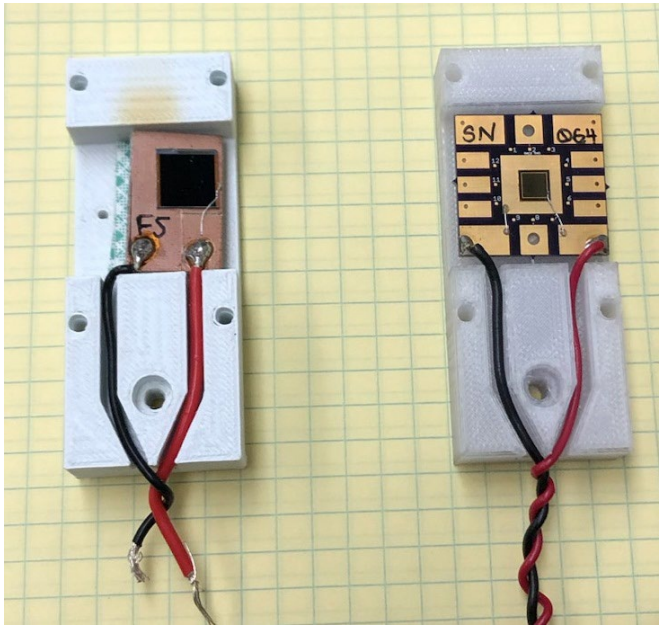

Fig. 5 InGaP (left) and SiC (right) ECs in cartridges

The ECs are epoxied onto a wafer with soldered wires. The package is attached to a 3-D printed plastic cartridge (white) that slides within the drawer under the pillbox. It was essential to construct a package that places the active surface of the EC as close as possible to foil to prevent attenuation of emitted particles due to air and reduced subtended area. The closest reliable distance obtained in this experiment to ensure ease of adjustment is 2 mm.

Power measurements take place at incremental distances from the RI. The cartridge containing the EC is placed within the slot and slides under the foil held in the pillbox. Distance changes are made by using a micrometer to establish the edge distance required. The micrometer is held with one edge at the end of the desk and the adjustable element touching the cartridge (Fig. 6).

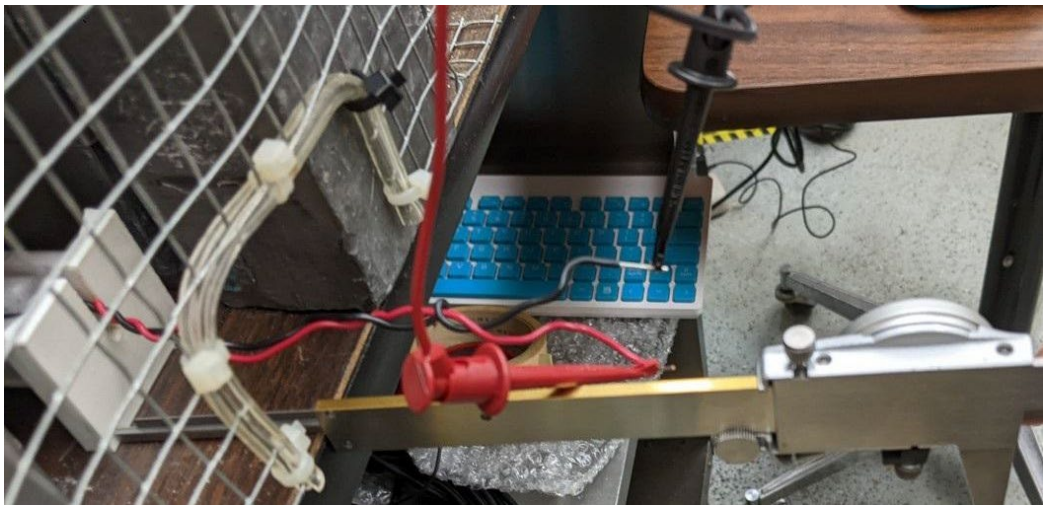

Fig. 6 The cartridge is pushed below the pillbox to a distance determined by the micrometer

3.2 Radioisotope Sources

The RIs are placed on foils consisting mostly of their stable nuclear forms.[14] The ^{147}Pm beta source at purchase had a nominal activity of 100 mCi, an active-to-total diameter of 15.6/22 mm encased in a 5-μm titanium window. The ^{63}Ni has a diameter of 10 mm (Fig. 7). This configuration ensures the RI is not touched, which prevents contamination and lowers risk to health through contact.

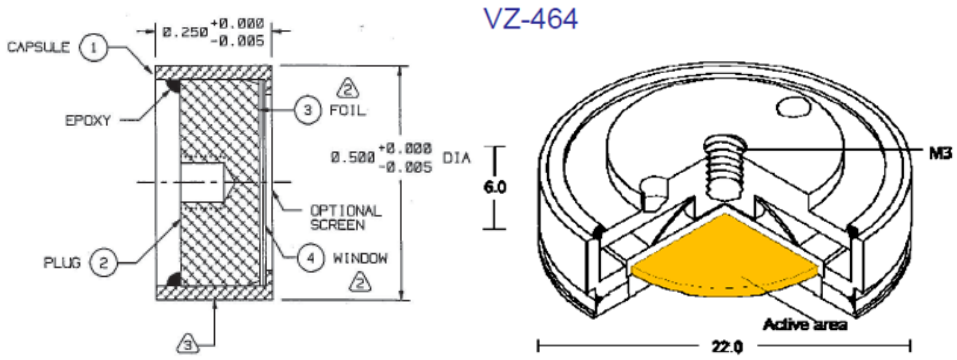

Fig. 7 RI housing schematics for (left) ^{63}Ni and (right) ^{147}Pm

To ensure safety and security, the RIs are housed in pillboxes. The pillboxes are placed in cages (Fig. 8) as per Nuclear Regulatory Commission license. Lead bricks surround the sides and top of the ^{147}PM pill box due to the RI's further penetrating beta particles.

Fig. 8 RIs (left) ^{63}Ni and (right) ^{147}Pm are housed in pillboxes that include a drawer to insert the EC cartridges

The activity of the RI sources is determined by the decay rate and amount of material. The decay rate is defined by Eq. 2, where λ is the material's decay constant, N_0 is the initial amount, and t is time elapsed. The Lambda values (decay/year) are 0.00697 for ^{63}Ni and 0.265 for ^{147}Pm.

$$N = N_0 e^{-\lambda t}. \quad (2)$$

The decay of the ^{63}Ni and ^{147}Pm foils used is represented in Fig. 9. Originally, 10 mCi of ^{63}Ni was obtained on March 1, 2018; and 100 mCi of ^{147}Pm was obtained on April 1, 2018. At the time of testing, the ^{63}Ni was present at 9.70 mCi and the ^{147}Pm at 30.5 mCi.

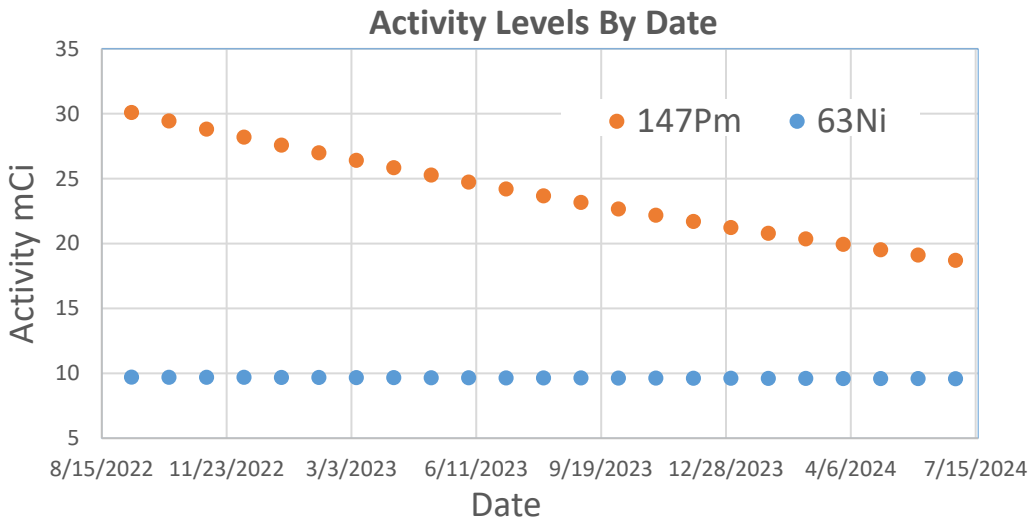

Fig. 9 Calculated effective activity for ^{147}Pm and ^{63}Ni as a function of time from original purchase

The RIs emit a spectrum of energies (Fig. 10) with an average of 16 keV[15] for ^{63}Ni and 62 keV[13] for ^{147}Pm. The high-energy betas penetrate deeper into the material. In the case of ^{147}Pm, which has a penetration range of 19.3 μm in InGaP, this far exceeds the charge collection thickness of 400 nm.

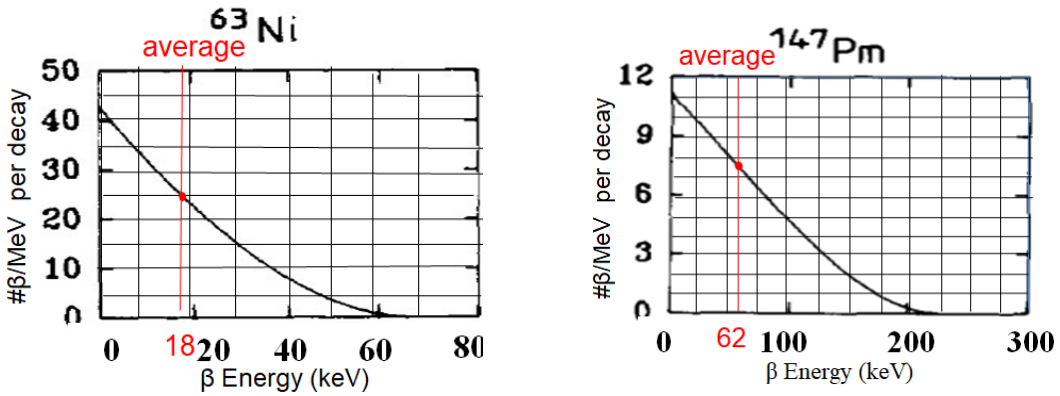

Fig. 10 Energy spectrum of electrons per million electronvolts (MeV) per decay vs. energy for that emitted particle for (left) ^{63}Ni and (right) ^{147}Pm

3.3 DAQ System

DAQ is performed using a current voltage (IV)-curve trace setup of equipment (Fig. 11). A Keithley 2450 source meter is used as a picoAmmeter,[16] and a Raspberry Pi computer is used to control the meter and the four-port switch. The controller initializes the ammeter, directs the signals from the device under test to the picoAmmeter, then collects the data from the ammeter and stores it on a USB drive.

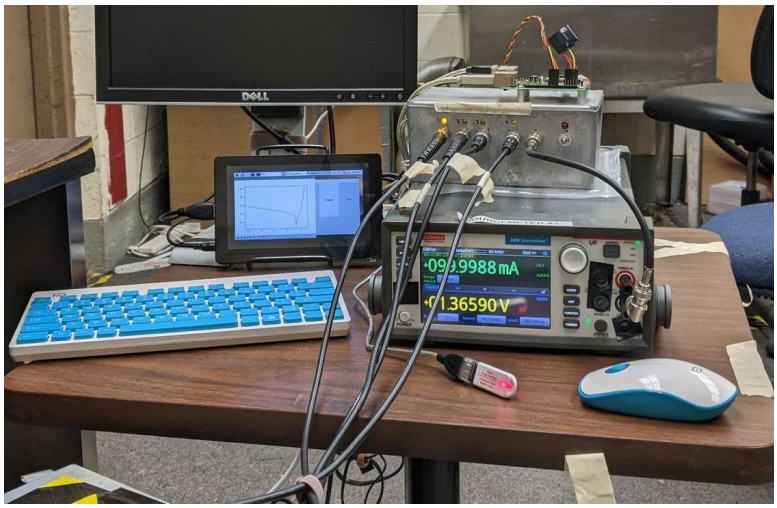

Fig. 11 The IV-curve test station includes the Keithley 2450 source meter (bottom right), a Raspberry Pi microcontroller (top right), four-device hub (center), and output console with keyboard and mouse

Custom Raspberry Pi DAQ code for running the IV-sweep and displaying a logarithmic plot is presented in Appendix B.

4. Modeling

Numerical calculations were performed using Monte Carlo Nuclear-Particle (MCNP) Transport Code simulations.[17] Energy deposited on the EC (SiC and InGaP) was calculated by using millions of particle tracks to mimic actual exposure. A geometry of figures in space is created (Fig. 12) with material properties included. Separate simulations are run with adjusted separation distances from the EC to source to gain a spread of values for the coupling efficiency.

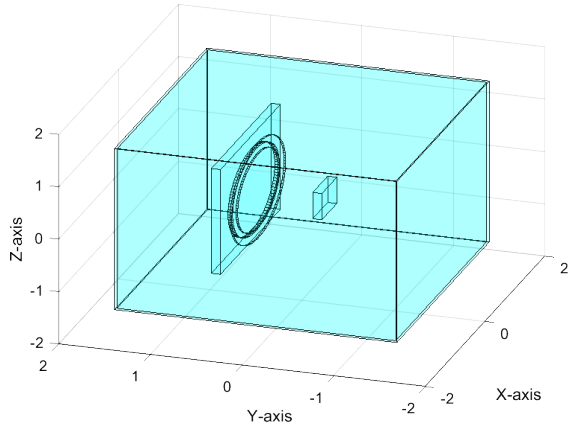

Fig. 12 Modeling geometry used to produce energy deposition (E_{dep}) data. The RI is depicted on the left as a ring attached to the backing plate. The EC is the small box on the right.

Simulation Approach

Calculations were performed along two axes. The vertical distance between RI and EC, and the lateral distance between RI and EC. Both were helpful in comparing expected values for experimental measurements. As we moved the drawer with the EC farther from the RI source, lateral attenuation was noticeable. The distance from RI in the pillbox to each EC in its cartridge was experimentally fixed at 2 mm for both ^{63}Ni and the ^{147}Pm. The simulations varied the distance, providing insight as to the value of positioning the EC as close as possible to the RI source.

To determine coupling efficiency, simulations calculated E_{dep} of electrons into the semiconductor with a penetration depth defined by the charge collection volume (300 nm for InGaP and 1500 nm for SiC). Lateral modeling mesh plots are shown in Fig. 13.

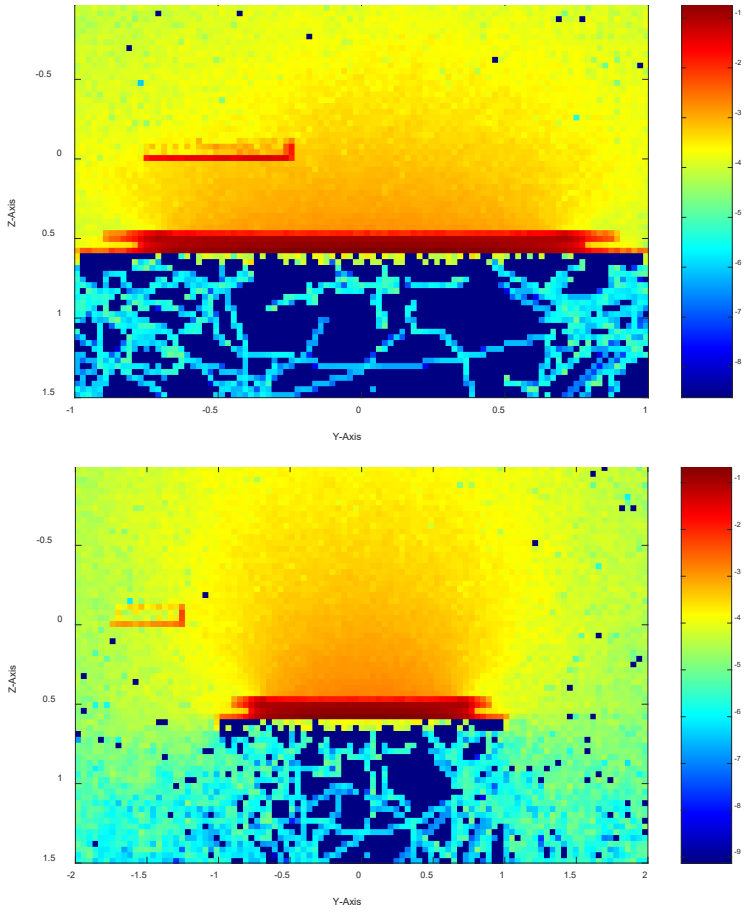

Fig. 13 Simulations of SiC (top) moving laterally across the plane to ^{147}Pm source (bottom) from (top) 0.25 mm to (bottom) –1.0 mm. The color bar shows higher concentrations of particle E_{dep} as redder and low to none as bluer. Z distance is plotted against Y distance.

The final model mesh also demonstrates E_{dep} for the setup (Fig. 14). The higher density of particle trails is emitted from the RI source. Many are deflected by the source, some are absorbed, but most pass through.

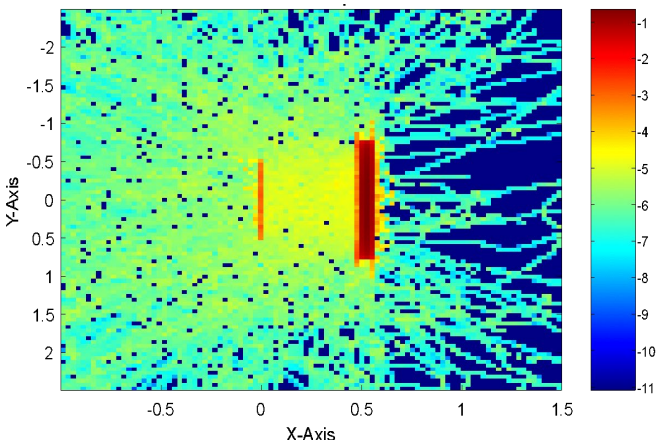

Fig. 14 E_{dep} (MeV/g) in InGaP detector (left) and ^{147}Pm source (right)

Energy deposited in air surrounding source and detector is 10,000 times reduced compared to energy deposited in InGaP and ^{147}Pm foil. A compilation of all the results of vertical separation of RI and EC for the four variations is shown in Fig. 15. Each interaction follows an exponential drop in efficiency, suggesting the significance of closing the distance gap. For example, for ^{63}Ni paired with the betavoltaic SiC (blue), the coupling efficiency of 2 mm separation is 1.75%, while for 7 mm it is 0.0343%.

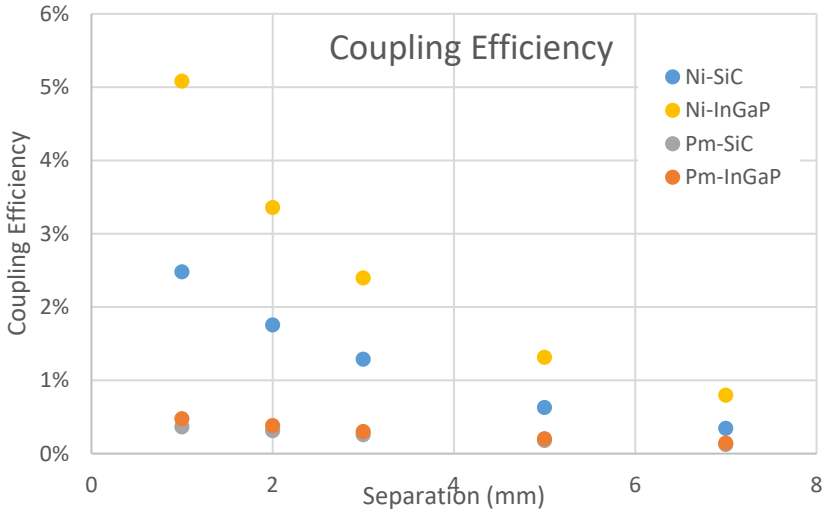

Fig. 15 Coupling efficiency for all four combinations of RIs and ECs used in experiments

5. Results

Tests are taken using two ECs (InGaP and SiC) exposed to two RIs (^{63}Ni and ^{147}Pm). In this section we report the data acquired following the tests.

The following descriptions include the IV-curves measured for each of the four RI–EC variations. These variations include ^{63}Ni on SiC, ^{63}Ni on InGaP, ^{147}Pm on SiC, and ^{147}Pm InGaP. The IV-curves are followed by a table of values of maximum power point (MPP) measured for each EC position.

5.1 Semiconductor Energy Converter Evaluation

The new ARL calibration lab at Building 504 enables testing of semiconductor ECs with unknown device efficiencies by exposing them to RIs of known energy outputs.

Pretest evaluations of the ECs are found in Appendix A. Data was first obtained under conditions of dark (covered, room light, and LED flashlight) to vary the photon stimulus to verify operation of EC.

Using the known RI activity, RI to EC coupling efficiency was numerically calculated through simulation in MCNP. Experimentally, the power output of the EC was measured using the IV-curve tracer. The detector efficiency is determined from Eq. 4.

$$P_{out} = P_{in} * \eta_{coupling} * \eta_{detector}. \quad (4)$$

Key parameters obtained from the test are short-circuit current (I_{sc}), a measurement of β-induced I_{sc}, and the open-circuit voltage, V_{oc}. The MPP (Eq. 5) is the product of the two quantities and fill factor (FF), which measures the squareness of the cell by finding the largest rectangle that fits in the IV-curve.[12]

$$MPP = I_{sc}V_{oc}FF\ . \quad (5)$$

5.1.1 ^{63}Ni on SiC and InGaP

Figure 16 shows the IV-curves from ^{63}Ni lateral power measurements. Currents are measured as increasing with distance from the table edge until the EC is directly below the source. Because the signal strength is too low, no usable InGaP data is plotted in this section.

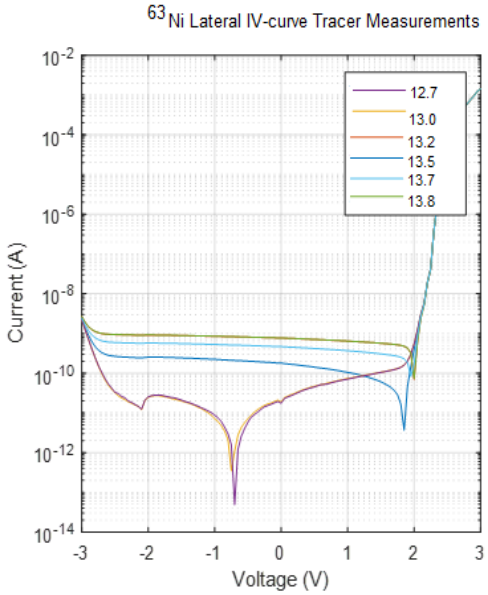

Fig. 16 Logarithmic plot of IV-curve traces across lateral motion for six measurements from ^{63}Ni irradiating SiC. The green line closest to the source (13.8 cm) correlates to the most power measured. The purple and orange curves received little power; the curve behaviors indicate a dark current.

From the IV-curve tracer result we calculated the MPP values as the EC was moved from the end of the table to the farthest distance inside the RI pillbox to record an MPP of 772 pW (Table 3).

Table 3 ^{63}Ni initial lateral measurements

Distance (cm)	SiC (pW)
12.7	18.9
13.0	17.6
13.2	181
13.5	468
13.7	768
13.8	772

Measurements from the test are plotted in Fig. 17. A gradual rise is seen for MPP as the cartridge containing the EC is placed farther from the table edge and closer to RI.

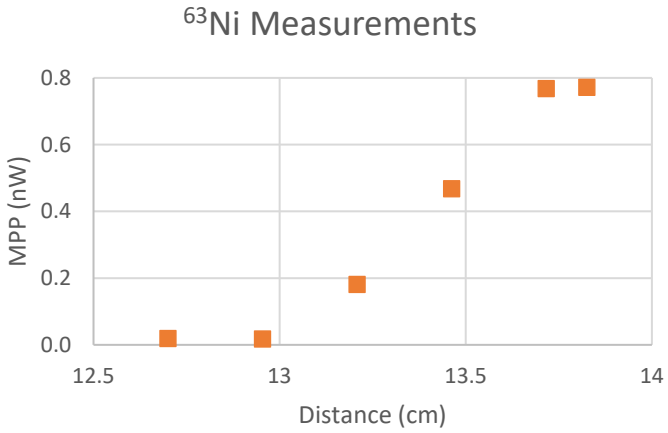

Fig. 17 Lateral power measurements of ^{63}Ni irradiating SiC; the isotope was too low to deliver measurable power to InGaP

5.1.2 ^{147}Pm on SiC and InGaP

Tests from ^{147}Pm irradiating SiC and InGaP are presented in Fig. 18. Both ECs recorded viable currents verifying the expected semiconductor and dark current IV-curve behavior.

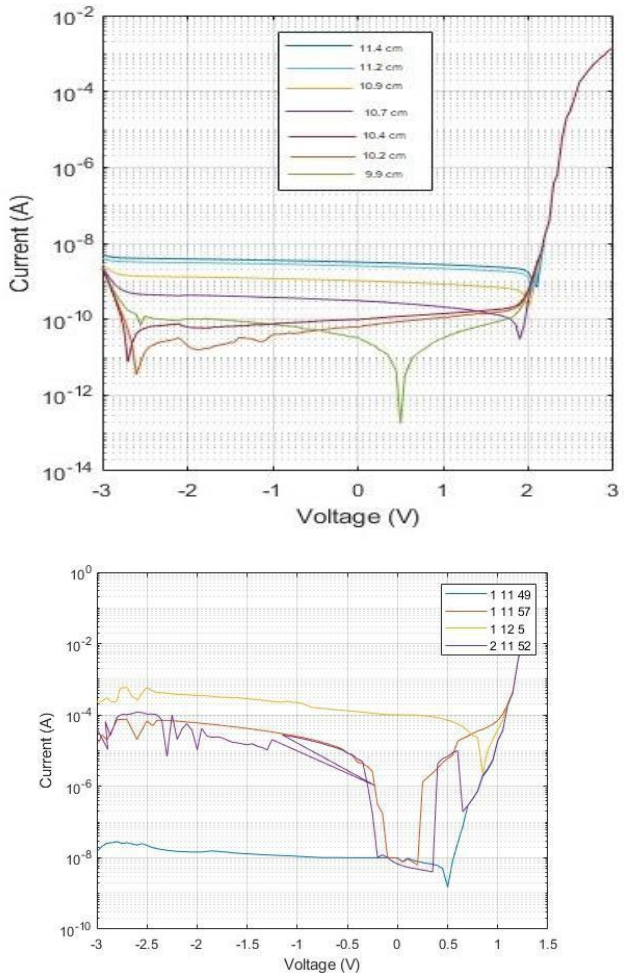

Fig. 18 Logarithmic plot of IV-curve traces across lateral motion for six specified measurements from ^{147}Pm to SiC (top) and InGaP (bottom)

Power was measured as the cartridges were moved farther into the pillbox as a function of energy in nW (Table 4). The InGaP chip failed after the third test (11.4 cm) and further tests at 10.7 and 11.2 cm were missed. The largest amount of power (2.43 nW for InGaP and 4.04 nW for SiC) was measured at the farthest possible distance from the table (11.4 cm) and indicates the closest the chip can be placed under the RI foils.

Table 4 ^{147}Pm lateral motion measurements of detector

distance [cm]	InGaP [nW]	SiC [nW]
10.4	0.566	0.00413
10.7	-	0.229
10.9	1.40	1.14
11.2	-	3.16
11.4	2.43	4.04

Measurements from the test are plotted in Fig. 19. Both sets follow a gradual increase in MPP as distance increases to a maximum allowable by the drawer. The farthest distance places the EC directly under the RI.

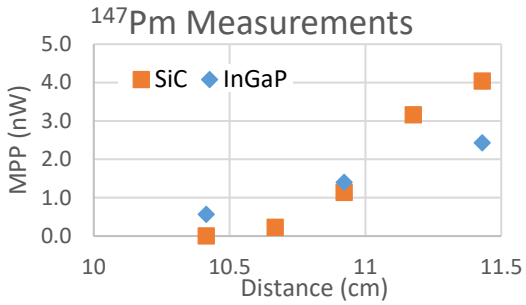

Fig. 19 Tests of ^{147}Pm irradiating SiC and InGaP ECs

5.2 Summarized Results

RI power reaching the detector is calculated from the activity of the RI source. The power is subtended by semiconductor area calculated in the MCNP modeling. The power out is measured in the picoAmmeter during the IV-curve sweep. The detector efficiency is then determined as the last missing variable (Table 5).

Table 5 Summary data table

Isotope	E_{avg} [keV]	Half-life [yr]	Activity [mCi]	EC	P_{in} [nW]	P_{out} [nW]	Edep [nJ/s]	η_{couple}	η_{EC}
^{63}Ni	16	92	10 1.07 µW	InGaP	35.8	0.130	4.84	3.35%	0.36%
				SiC	18.7	0.772	3.86	1.75%	4.13%
^{147}Pm	62	2.6	35 12.8 µW	InGaP	48.9	2.43	61.0	0.380%	4.97%
				SiC	40.1	4.04	43.1	0.312%	10.07%

The EC efficiencies are shown in Fig. 20. ^{63}Ni irradiating SiC has a notably large efficiency due to the shorter penetration range of the RI and surface area of the chip. ^{63}Ni-InGaP has the lowest efficiency because InGaP is photovoltaic, not betavoltaic, and is limited in how much energy can be extracted. These results can be used to predict future combinations to determine what EC efficiencies should be expected for novel devices.

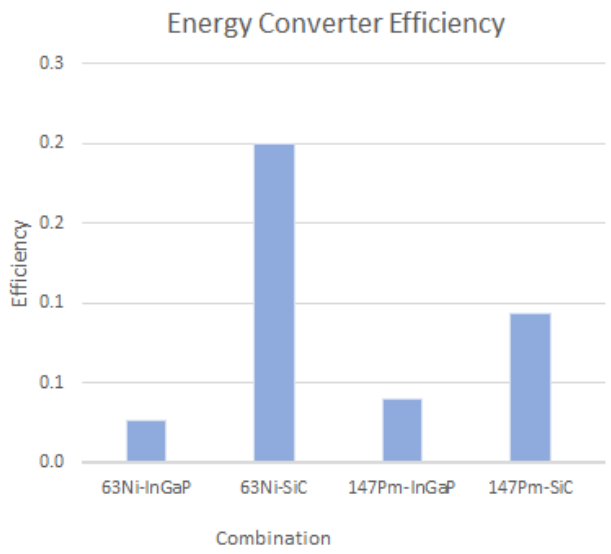

Fig. 20 Device efficiency comparison based on EC–RC combination

6. Conclusion

The tests completed in this report verify the usefulness of this calibration facility. Implementing a picoAmmeter enabled the detection of 0.416%–32.2% of power from the semiconductor.

Energy levels measured were in expected values within equipment tolerances. When irradiating SiC, the ^{63}Ni foil registered an MPP of 0.591 nW, V_{oc} of 1.89 V, and FF of 59.4%.

This facility is effective for betavoltaic designed devices but not as effective for devices stimulated through other energy sources (photon/alpha). We will next perform tests to investigate ultra-wide BG materials such as GaN, AlGaN, and diamond.

7. References

1. Xiao-Yi L, Lu J-B, Liu Y-M, Xue X, He R, Zheng, R-Z. Exploratory study of betavoltaic battery using ZnO as the energy converting material. Nucl Sci Tech. 2019.
2. Miotla D. Assessment of plutonium-238 production alternatives. Department of Energy (US); 2008.
3. Romer M, Miley GH, Luo N, Gimlin RJ. Ragone plot comparison of radioisotope cells and the direct sodium borohydride/hydrogen peroxide fuel cell with chemical batteries. IEEE Trans Energy Conv. 2008;23:171.
4. Klein C, Philpotts AR. Earth materials: introduction to mineralogy and petrology, 2nd ed. Cambridge University Press; 2017. ISBN 978-1-107-15540-4. OCLC 962853030.
5. Rajendran V. Materials science. Tata McGraw-Hill Pub; 2004. p. 2.16. ISBN 978-0-07-058369-6.
6. Wang J, Mulligan P, Brillson L, Cao LR. Review of using gallium nitride for ionizing radiation detection. Appl Phys Rev. 2015;2:031102.
7. Asgari A, Kalafi M. The control of two-dimensional-electron-gas density and mobility in AlGaN/GaN heterostructures with Schottky gate. Mater Sci Eng C. 2006;26:898–901.
8. Rui L. Integrated AlGaN/GaN HEMTs in MCM-D technology. Proceedings of the Electronic Components and Technology Conference; 2010. doi: 10.1109/ECTC.2020.5490783.
9. Varshni Y. Temperature dependence of the energy gap in semiconductors. Physics. 1967;34:149–154.
10. Litz M, Tompkins R, Russo J, Pullen C, Kierzewski I, Kelley, Doumbia M, Smith B. Isotope energy release and deposition characteristics in betavoltaic materials. Army Research Laboratory (US); 2020. Report No.: ARL-TR-8941.
11. Klein CA. Bandgap dependence and related features of radiation ionization energies in semiconductors. J Appl Phys. 1967.
12. Honsberg CB, Bowden SG. Photovoltaics education website. Arizona State University; 2019. https://www.pveducation.org/.
13. Khan M. Design and characterization of GaN P-I-N diodes for betavoltaic devices. Solid State Electron. 2017.

14. Eckert Ziegler. Isotope products catalogue 10mCi 63Ni and 100mCi 147Pm 10mCi VZ-464 (foil product schematics). Eckert Ziegler; 2022.

15. Cross WG. A short atlas of beta-ray spectra. Phys Med Biol. 1983.

16. Tektronix. Model 2450 source meter instrument reference manual. Tektronix; 2019.

17. Werner CJ, ed. MCNP user's manual, code version 6.2, LA-UR-17_299821. Los Alamos National Security, LLC; 2017 Oct. https://mcnp.lanl.gov /pdf_files/TechReport_2017_LANL_LA-UR-17-29981_WernerArmstrongEtAl.pdf.

Appendix A. Pretest Verification Measurements

A.1 Pretest Calibrations

To calibrate the energy converters (ECs), a test with a dark room and LED light was performed. Each curve's height change corresponds to more input power. Dark conditions have the lowest height and demonstrate resistive behavior beyond the 2-V open-circuit voltage (V_{oc}). In room light the EC current floor rises to about 10 nA as it receives some light. As the device is flooded with LED light, the current floor reaches a limit around 50 nA.

These pretests help establish a baseline using common optical photons before measuring under a beta source. A good device displays the proper response curves that have a dip at the V_{oc}, indicating a change from positive to negative currents (Fig. A-1). The values are adjusted to absolutes to display properly on a log plot.

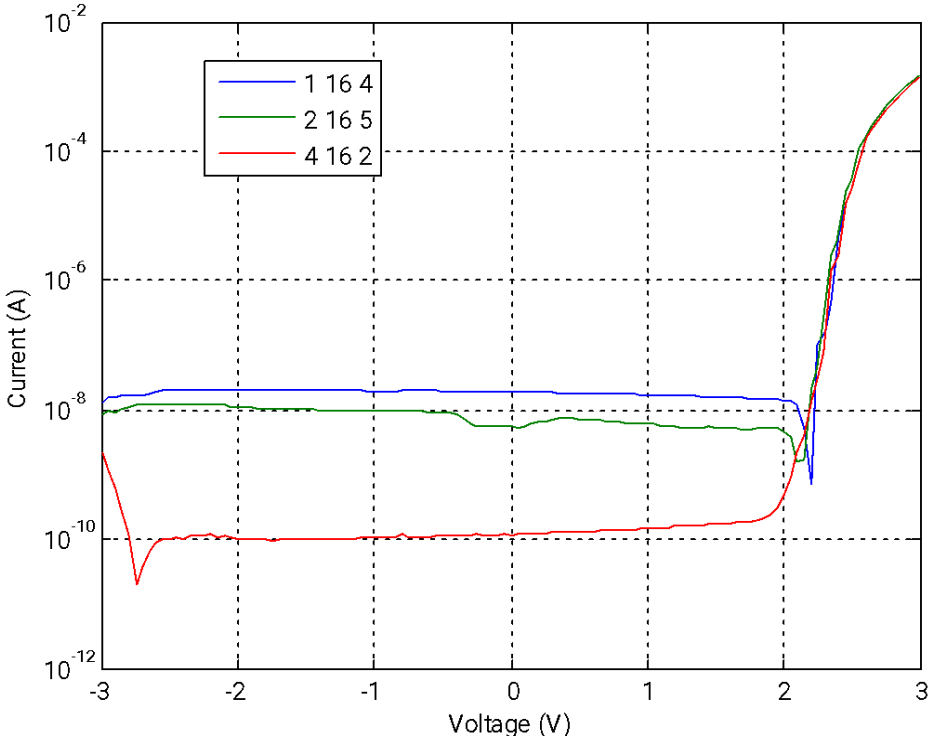

Fig. A-1 Silicon carbide (SiC) tritium lab EC under conditions dark (red), room light (green), and NEBO LED held at 2 inches (blue)

Appendix B. Current Voltage (IV)-Curve Tracer Code

This appendix appears in its original form, without editorial change.

```
#operate the source meter through PyVISA using SCPI

#import the PyVISA library
from tkinter import *
import tkinter
import tkinter.font as tkFont
import visa
import time
import csv
import keyboard
import sys
import termios
import os
from RPi import GPIO


#initialize main window
def init_window(win, width, height, ports):

    #determine the scaling coefficient
    if ports >= 2:

        scaleC = ports + 2

    else:

        scaleC = 2

    #configure the initial window
    win.title("Device Selection")
    win.minsize(str(scaleC*width), str(int(1.5*height)))


#center the window
def center_window(win, width, height, ports):

    #obtain the values for the screen width and height
    screenWidth  = win.winfo_screenwidth()
    screenHeight = win.winfo_screenheight()

    #calculate the position coefficient
    if ports >= 2:

        posC = ports + 2

    else:

        posC = 2

    #calculate the center of screen position
    positionRight = int(screenWidth/2 - posC*width/2)
    positionDown  = int(screenHeight/2 - 3*height/4)
```

```
    #position the window at the center of the screen
    win.geometry("+{}+{}".format(positionRight, positionDown))


#set the font and font size
def configure_formats(win):

    #create a text object
    text = tkinter.Text(win)

    #set the font style
    FS = tkFont.Font(family = "Times New Roman", size = 20)

    #configure the text
    text.configure(font = FS)

    #return the font settings
    return FS


#create a readout
def init_readout(win, height):

    #prepare the strings
    global rd1
    global rd2

    #create a string variable
    rd1 = StringVar()
    rd2 = StringVar()

    #create a label object
    label1 = tkinter.Label(win, textvariable = rd1, font = ("Times
New Roman", 20))
    label2 = tkinter.Label(win, textvariable = rd2, font = ("Times
New Roman", 20))

    #set the default labels
    rd1.set('Current Device: ')
    rd2.set('Next Device:    ')

    #locate the label
    label1.place(x = '50', y = str(int(1.1*height)))
    label2.place(x = '50', y = str(int(1.3*height)))


def update_readout(n1, n2):

    #set suspend as a global
    global suspend
```

```
    #construct the readout strings
    rst1 = 'Current Device:  ' + str(n1)
    rst2 = 'Next Device:      ' + str(n2)
    rst3 = 'Program Suspended'

    #update the readout
    rd1.set(rst1)
    #rd2.set(rst2)

    if suspend == 1:

        rd2.set(rst3)

    else:

        rd2.set(rst2)


def update_exit_readout(n):

    #construct the readout strings
    rst1 = 'Current Device:  ' + str(n)
    rst2 = 'Next Device:  Exit Program'

    #update the readout
    rd1.set(rst1)
    rd2.set(rst2)


#add the buttons to the GUI
def initialize_buttons(FS, width, height, ports):

    #set the suspend variable
    global suspend

    #set the suspend state
    suspend = 0

    #set the positional variable
    if ports >= 2:

        position1 = 0
        position2 = width
        position3 = 2*width
        position4 = 3*width
        position5 = ports*width
        position6 = (ports + 1)*width

    else:

        position5 = 0
        position6 = width
```

```
    #initialize the buttons
    if ports >= 2:

        initialize_button1(FS, width, height, position1)
        initialize_button2(FS, width, height, position2)
        initialize_button3(FS, width, height, position3)
        initialize_button4(FS, width, height, position4)

    initialize_button5(FS, width, height, position5)
    initialize_button6(FS, width, height, position6)


#position the buttons
def initialize_button1(FS, width, height, posV):

    #configure the button
    button = tkinter.Button(win, text = 'Device 1', command =
button1)

    #set the font
    button['font'] = FS

    #position the button
    button.place(bordermode = OUTSIDE, height = str(height), width =
str(width), x = str(posV), y = 0)


def initialize_button2(FS, width, height, posV):

    #configure the button
    button = tkinter.Button(win, text = 'Device 2', command =
button2)

    #set the font
    button['font'] = FS

    #position the button
    button.place(bordermode = OUTSIDE, height = str(height), width =
str(width), x = str(posV), y = 0)


def initialize_button3(FS, width, height, posV):

    #configure the button
    button = tkinter.Button(win, text = 'Device 3', command =
button3)

    #set the font
    button['font'] = FS

    #position the button
```

```
    button.place(bordermode = OUTSIDE, height = str(height), width =
str(width), x = str(posV), y = 0)


def initialize_button4(FS, width, height, posV):

    #configure the button
    button = tkinter.Button(win, text = 'Device 4', command =
button4)

    #set the font
    button['font'] = FS

    #position the button
    button.place(bordermode = OUTSIDE, height = str(height), width =
str(width), x = str(posV), y = 0)


def initialize_button5(FS, width, height, posV):

    #configure the button
    button = tkinter.Button(win, text = 'Suspend', command =
button5)

    #set the font
    button['font'] = FS

    #position the button
    button.place(bordermode = OUTSIDE, height = str(height), width =
str(width), x = str(posV), y = 0)


def initialize_button6(FS, width, height, posV):

    #configure the button
    button = tkinter.Button(win, text = 'Quit', command = button6)

    #set the font
    button['font'] = FS

    #position the button
    button.place(bordermode = OUTSIDE, height = str(height), width =
str(width), x = str(posV), y = 0)


#assign functionality to button 1
def button1():

    #set next device as global
    global next_device

    #indicate the button was depressed
```

```
    print('Button 1 Pressed')
    print('')

    #update the next device
    next_device = 1

    #revise the readout
    update_readout(device, 1)


#assign functionality to button 2
def button2():

    #set next device as global
    global next_device

    #indicate the button was depressed
    print('Button 2 Pressed')
    print('')

    #update the next device
    next_device = 2

    #revise the readout
    update_readout(device, 2)


#assign functionality to button 3
def button3():

    #set next device as global
    global next_device

    #indicate the button was depressed
    print('Button 3 Pressed')
    print('')

    #update the next device
    next_device = 3

    #revise the readout
    update_readout(device, 3)


#assign functionaltiy to button 4
def button4():

    #set next device as global
    global next_device

    #indicate the button was depressed
    print('Button 4 Pressed')
```

```
    print('')

    #update the next device
    next_device = 4

    #revise the readout
    update_readout(device, 4)


#assign functionaltiy to button 5
def button5():

    #set suspect as global
    global suspend
    global next_device

    #indicate the button was depressed
    print('Button 5 Pressed')
    print('')

    #update the suspend state
    suspend = suspend + 1

    #wrap the suspend state
    if suspend > 1:

        suspend = 0

    #revise the readout
    update_readout(device, next_device)


#assign functionaltiy to button 6
def button6():

    #set next_device as global
    global suspend
    global next_device

    #indicate the button was depressed
    print('Button 6 Pressed')
    print('')

    #update the next device
    next_device = 0

    #update the suspend state
    suspend = 0

    #print the new variable
    print(next_device)
    print('')
```

```
    #revise the readout
    update_exit_readout(device)


#enable automatic control
def automatic_control(instr, duration, ports):

    #set device to a global variable so as to be accessible by the
GUI
    global device
    global next_device
    global suspend

    #set the initial device to 1 and the next device to 2
    device = 1

    #set the initial readout
    if ports >= 2:

        next_device = 2
        update_readout(1, 2)

    else:

        next_device = 1
        update_readout(1, 1)


    #set the termination condition variable
    term = 0

    #convert to seconds
    duration = 60.0*duration

    #obtain current time
    t = time.time()

    #calculate timeout
    tout = t + duration

    #begin automatic control
    while t < tout:

        #set the correct analog channel
        if ports >= 1:

            device_select(device)

        #conduct a linear sweep
        linear_sweep(instr, device)
```

```
        #hold up the system if suspend is enabled
        while suspend == 1:

            #update the gui
            win.update()

            #wait
            time.sleep(1)

        #exit program upon appropriate selection
        if next_device == 0:

            return 0

        if ports >= 2:

            #select the next device
            device = next_device
            next_device = next_device + 1

            #cycle back to the first device
            if next_device > ports:
                next_device = 1

            #update the readout
            update_readout(device, next_device)

        #revise time
        t = time.time()

        #wait
        time.sleep(1)


#write the measurement data to file
def write_data(data, n):

    #parse the data
    V = data[n]
    I = data[n + 1]
    D = data[n + 2]
    T = data[n + 3]

    #append to a csv file
    with open(datapath, 'a', newline = '', encoding = 'utf-8') as
csvfile:

        writer = csv.DictWriter(csvfile, fieldnames = fieldnames)

        #write the data to file
        writer.writerow({'Voltage': V, 'Current': I, 'Date': D,
'Time': T})
```

```
def write_sweep_to_file(data, points, device):

    #convert data from string to list
    data = data.split(',')

    #remove the newline character from the end of the list
    data[-1] = data[-1].strip()

    #create the file to be written to
    create_file(device)

    #convert points to integer
    points = int(points)

    #evaluate all points points (note that the for loop goes from 0
to points - 1
    for index in range(points):

        #write the data to file
        write_data(data, 4*index)

        #update the gui
        win.update()


#set the filepaths
def create_file(device):

    #set the global variables
    global datapath
    global fieldnames

    #obtain the current time
    t = construct_time()
    date = construct_date()

    #set the data writepath
    datapath = usbpath + 'device' + str(device) + '_' + str(date) +
'_' + str(t) + '.csv'
    fieldnames = ['Voltage', 'Current', 'Date', 'Time']

    #open the relevant csv files
    with open(datapath, 'w', newline = '', encoding = 'utf-8') as
csvfile:

        writer = csv.DictWriter(csvfile, fieldnames = fieldnames)

        #set the data labels
        writer.writeheader()
```

```
#construct the date
def construct_date():

    #obtain the time
    complete_date = time.localtime()

    #construct the date
    year = complete_date[0]
    month = complete_date[1]
    day = complete_date[2]

    #convert to constant character length string
    year = convert_datetime(year)
    month = convert_datetime(month)
    day = convert_datetime(day)

    date = year + '-' + month + '-' + day

    return date


def construct_time():

    #obtain the time
    complete_date = time.localtime()
    time_seconds = time.time()

    #construct the time
    hour = complete_date[3]
    minute = complete_date[4]
    seconds = time_seconds % 60
    seconds = round(seconds)

    #convert to constant character length string
    hour = convert_datetime(hour)
    minute = convert_datetime(minute)
    seconds = convert_datetime(seconds)

    time_string = hour + '-' + minute + '-' + seconds

    return time_string


#set the instrument time
def set_instr_time(instr):

    #obtain the time
    complete_date = time.localtime()
    time_seconds = time.time()

    #construct the date
    year = complete_date[0]
```

```
    month = complete_date[1]
    day = complete_date[2]

    #construct the time
    hour = complete_date[3]
    minute = complete_date[4]
    seconds = time_seconds % 60
    seconds = round(seconds)

    #construct the system time configuration string
    tstr = 'SYST:TIME ' + str(year) + ', '  + str(month) + ', ' +
str(day) + ', ' + str(hour) + ', ' + str(minute) + ', ' +
str(seconds)

    #set the new time to the instrument
    instr.write(tstr)


#convert the dates and times to a constant character length string
def convert_datetime(value):

    #check to determine if the datetime value requires zero padding
    if value >= 10:

        rstring = str(value)
        return rstring

    else:

        rstring = '0' + str(value)
        return rstring


#configure the GPIO pins
def configure_GPIO():

    #set the GPIO mode
    GPIO.setmode(GPIO.BCM)

    #set the device pins
    GPIO.setup(22, GPIO.OUT)
    GPIO.setup(23, GPIO.OUT)
    GPIO.setup(24, GPIO.OUT)
    GPIO.setup(25, GPIO.OUT)


#choose a device to enable
def device_select(dselect):

    #check for invalid entries
    valid = 0
```

```
#use the configuation file to set pin assignments
if dselect >= 1 and dselect <= 4:

    dselect = GPIO_config[dselect - 1]

#set the first device on
if dselect == 1:
    print("Device 1")
    print("")

    GPIO.output(22, 1)
    GPIO.output(23, 0)
    GPIO.output(24, 0)
    GPIO.output(25, 0)

    valid = 1

#set the second device on
if dselect == 2:
    print("Device 2")
    print("")

    GPIO.output(22, 0)
    GPIO.output(23, 1)
    GPIO.output(24, 0)
    GPIO.output(25, 0)

    valid = 1

#set the third device on
if dselect == 3:
    print("Device 3")
    print("")

    GPIO.output(22, 0)
    GPIO.output(23, 0)
    GPIO.output(24, 1)
    GPIO.output(25, 0)

    valid = 1

#set the fourth device on
if dselect == 4:
    print("Device 4")
    print("")

    GPIO.output(22, 0)
    GPIO.output(23, 0)
    GPIO.output(24, 0)
    GPIO.output(25, 1)

    valid = 1
```

```
    if dselect == 5:
        print("Disable All Devices")
        print("")

        GPIO.output(22, 0)
        GPIO.output(23, 0)
        GPIO.output(24, 0)
        GPIO.output(25, 0)

        valid = 1


    if valid == 0:
        print("Invalid Entry")
        print("")

        GPIO.output(22, 0)
        GPIO.output(23, 0)
        GPIO.output(24, 0)
        GPIO.output(25, 0)

        return 1

    return 0


def program_shutdown(ports):

    #buffer the text outputs
    print('')
    print('')

    if ports >= 1:

        #turn off all devices
        device_select(5)

        #reset the GPIO devices
        GPIO.cleanup()

    #close the GUI
    win.destroy()

    #end the program
    return 0


#configure the VISA interface
def configure_interface():

    #test the PyVISA library
```

```
    rm = visa.ResourceManager('@py')
    rlist = rm.list_resources()

    print(rlist[1])
    print(rm.list_resources())
    print('')

    #obtain the address
    addr = rlist[1]
    #addr = 'USB0::1510::9296::04331608::0::INSTR'

    #set the instrument
    instr = rm.open_resource(addr)

    #set the protocol string
    pstring = '*LANG SCPI'

    #set the read command
    read = 'READ?'

    #set the buffer related strings
    cquery = 'TRAC:ACT?'
    cbuff = 'TRACE:CLEAR'


    #set the protocol
    instr.write(pstring)

    #clear the buffer
    #instr.write(cbuff)
    #print(instr.query(cquery))

    #configure the instrument
    #instr.write_termination('\n')

    #query the instrument
    print(instr.query('*IDN?'))
    print(instr.query('*LANG?'))

    #set the beep string
    beepstring = 'SYSTem:BEEPer 500, 0.5'

    #beep the instrument indicating the establishment of the SCPI
communication is successful
    instr.write(beepstring)
    time.sleep(1)
    instr.write(beepstring)
    time.sleep(1)
    instr.write(beepstring)

    #write the instrument handle as the output
    return instr
```

```
#perform initial instrument configuration
def configure_meter(instr, counts, ilimit):

    #set the reset string
    reset = '*RST'

    #set the source string
    stype  = 'SOUR:FUNC VOLT'
    sauto  = 'SOUR:VOLT:RANG:AUTO 1'
    slimit = 'SOUR:VOLT:ILIM ' + str(ilimit)

    #set the measurement strings
    mtype = 'SENS:FUNC CURR'
    mauto = 'SENS:CURR:RANG:AUTO 1'

    #set the averaging string
    savg = 'CURR:AVER:COUNT ' + str(counts)
    #savg = 'CURR:AVER:COUNT 1'
    eavg = 'CURR:AVER ON'
    qavg = 'CURR:AVER:COUNT?'

    #set the measurement counts
    mcount = 'COUN 1'

    #set the capacitance strings
    #caps = 'SOUR:CURR:HIGH:CAP ON'  did not seem to effect
instrument
    caps = 'SOUR:VOLT:HIGH:CAP ON'

    #set the display strings
    dstr = 'DISP:CURR:DIG 6'

    #set the autozero string
    azs = 'CURR:AZER OFF'

    #set the zero strings
    z1 = 'FUNC "CURR"'
    z2 = 'AZER:ONCE'

    #set the buffer related strings
    cbuff = 'TRACE:CLEAR'

    #set the instrument voltage protection setting
    #vprot = 'SOUR:VOLT:PROT PROT5'
    vprot = 'SOUR:VOLT:PROT PROT10'

    #set the autorange lower limt
    allim = 'CURR:RANG:AUTO:LLIM 1e-8'

    #reset the instrument
```

```
        instr.write(reset)

        #set the source type
        instr.write(stype)

        #set the source range
        instr.write(sauto)

        #set the source limit
        instr.write(slimit)

        #set the measurement range
        instr.write(mauto)

        #set high capacitance mode on
        instr.write(caps)

        #set to 6.5 digits
        instr.write(dstr)

        #set the counts
        instr.write(mcount)

        #set the overvoltage protection
        instr.write(vprot)

        #set the current low limt
        instr.write(allim)

        #set the averaging
        instr.write(savg)
        instr.write(eavg)
        #print('Number of Averages: ' + instr.query(qavg))

        #disable autozero
        #instr.write(azs)

        #zero the instrument
        #instr.write(z1)
        #instr.write(z2)

        #clear the buffer
        instr.write(cbuff)

        #set the measurement type
        #instr.write(mtype)

        #reset the instrument time
        set_instr_time(instr)


    #create a repository for the linear sweep settings
```

```
def sweep_settings(device):

    #set the index
    index = int(device - 1)

    #set the start, stop and increment parameters
    vs = start[index]
    vf = stop[index]
    dv = increment[index]

    #return the results
    return vs, vf, dv


#wait for the SourceMeter
def wait_for_meter(instr):

    #set the trace string
    tstate =  'TRIG:STAT?'

    #prepare wait routine
    opcheck = instr.query(tstate)
    opcheck = opcheck.split(';')
    opcheck = opcheck[0]

    #wait
    while opcheck == 'RUNNING':

        #update the gui
        win.update()

        #wait a few seconds
        time.sleep(1)

        #check if sweep has completed
        opcheck = instr.query(tstate)
        opcheck = opcheck.split(';')
        opcheck = opcheck[0]


#wait for the SourceMeter
def wait_for_write(delay):

    #wait
    for t in range(delay):

        #update the gui
        win.update()

        #wait a few seconds
        time.sleep(1)
```

```
#run a linear sweep using internal commands
def linear_sweep(instr, device):

    print('Begin Sweep')
    print('')

    #set the sweep parameters
    vs, vf, dv = sweep_settings(device)

    #calculate the number of points
    points = (vf - vs)/dv + 1

    #convert points to integer
    points = int(points)

    #set the sweep string
    swes = 'SOUR:SWE:VOLT:LIN ' + str(vs) + ', ' + str(vf) + ', ' +
str(points) + ', 1e-3, 1, AUTO, OFF'
    #swes = 'SOUR:SWE:VOLT:LIN ' + str(vs) + ', ' + str(vf) + ', ' +
str(points) + ', 1e-3, 1, AUTO, ON, OFF'

    print(swes)
    print('')

    #set the trigger strings
    inits = 'INIT'

    #set the trace string
    tstring = 'TRAC:DATA? 1, ' + str(points) + ', "defbuffer1",
SOUR, READ, DATE, TIME'

    #configure the linear sweep
    instr.write(swes)

    #wait between setup and initiation
    time.sleep(1)

    #run the linear swep
    instr.write(inits)

    #wait for the instrument
    wait_for_meter(instr)

    #set the beep string
    beepstring = 'SYSTem:BEEPer 500, 1'

    #beep to indicate the sweep is complete
    instr.write(beepstring)

    #retrieve the data
    data = instr.query(tstring)
```

```
    #write data to file
    write_sweep_to_file(data, points, device)

    #give the system pause to write data
    time.sleep(1)

    #set the beep string
    beepstring = 'SYSTem:BEEPer 500, 0.5'

    #beep the instrument indicating the establishment of the SCPI
communication is successful
    instr.write(beepstring)
    time.sleep(1)
    instr.write(beepstring)


#read the configuration file
def read_config_file():

    #set the data ararys to global
    global start
    global stop
    global increment
    global GPIO_config

    #prepare the data arrays
    start       = []
    stop        = []
    increment   = []
    GPIO_config = []

    #set the filename
    filename = '/home/pi/Desktop/sweep_configuration.csv'

    #open the csv file
    with open(filename, newline = '') as csvfile:

        #obtain data from file
        config_data = csv.reader(csvfile, dialect = 'excel')

        #prepare an array to read the config data
        data = []

        #transfer file data to an array
        for row in config_data:

            #append to data array
            data.append(row)

    #obtain the univeral parameters
    counts = int(data[3][1])
    ports  = int(data[4][1])
```

```
ilimit = float(data[5][1])

#convert current limit to A
ilimit = ilimit/1000

#convert device 1 data to float
st = float(data[8][1])
sp = float(data[9][1])
ic = float(data[10][1])

#write device 1 data to array
start.append(st)
stop.append(sp)
increment.append(ic)

#convert device 2 data to float
st = float(data[13][1])
sp = float(data[14][1])
ic = float(data[15][1])

#write device 2 data to array
start.append(st)
stop.append(sp)
increment.append(ic)

#convert device 3 data to float
st = float(data[18][1])
sp = float(data[19][1])
ic = float(data[20][1])

#write device 3 data to array
start.append(st)
stop.append(sp)
increment.append(ic)

#convert device 4 data to float
st = float(data[23][1])
sp = float(data[24][1])
ic = float(data[25][1])

#write device 3 data to array
start.append(st)
stop.append(sp)
increment.append(ic)

#configure the GPIO order
GPIO_config.append(int(data[28][1]))
GPIO_config.append(int(data[29][1]))
GPIO_config.append(int(data[30][1]))
GPIO_config.append(int(data[31][1]))

#write the outputs
```

```
    return counts, ilimit, ports


def set_USB_path():

    #prepare the filepath
    global usbpath

    #set the initial path
    pi_media_path = "/media/pi/"

    #list the directories
    mediadev_list = os.listdir(pi_media_path)

    #obtain the list size
    ls = len(mediadev_list)

    #check the list length
    if ls >= 3:

        #remove 'data' and 'log' so only the USB drive appears
        mediadev_list.remove('data')
        mediadev_list.remove('log')

    #generate the USB path
    usbpath = pi_media_path + mediadev_list[0]

    #append a data directory
    usbpath = usbpath + "/data/"


#indicate program start
print('Program Start')
print('')

#wait
time.sleep(1)

#set the duration of the test in minutes
duration = 200000

#set the width and height of the GUI window
width  = 160
height = 300

#read the configuration data
counts, ilimit, ports = read_config_file()

#obtain the window
global win
win = tkinter.Tk()
```

```
#set the font style
FS = tkFont.Font(family = "Times New Roman", size = 20)

#configure the GUI
init_window(win, width, height, ports)
center_window(win, width, height, ports)
initialize_buttons(win, width, height, ports)
init_readout(win, height)

#configure the GPIO interface
if ports >= 1:

    configure_GPIO()

#configure the USB datapath
set_USB_path()

#configure the VISA interface
instr = configure_interface()

#confiure the instrument
configure_meter(instr, counts, ilimit)

#begin automatic operation
automatic_control(instr, duration, ports)

#shut down the program
program_shutdown(ports)

#indicate that the program is finished
print("IV Measurement Complete")
print('')
```

List of Symbols, Abbreviations, and Acronyms

3-D	three-dimensional
AlGaN	aluminum gallium nitride
ARL	Army Research Laboratory
BG	bandgap
BJT	bi-polar junction transistor
Ci	Curie
DAQ	data acquisition
DEVCOM	US Army Combat Capabilities Development Command
EC	energy converter
E_{dep}	energy deposition
EHP	electron-hole pair
FF	fill factor
InGaP	indium gallium phosphide
IV	current voltage
I_{sc}	short-circuit current
kCi	thousand Curies
LED	light-emitting diode
mCi	millicurie, million Curies
MCNP	Monte Carlo Nuclear-Particle
MeV	million electronvolt
MPP	maximum power point
Ni	nickel
PIN	positive–intrinsic–negative
Pm	promethium
RI	radioisotope
SiC	silicon carbide
USB	universal serial bus
V_{oc}	open-circuit voltage
Zn	zinc
ZnS	zinc sulfide

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB
1 (PDF)	DEVCOM ARL FCDD RLA GC M LITZ