REMOTE LABORATORY FOR RADIATION DETECTION AND PHYSICAL SECURITY EDUCATION

A Thesis

by

ANTHONY GALINDO

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Craig Marianno
Committee Members,	Sunil Chirayath
	Dylan Shell
Head of Department,	Michael Nastasi

May 2021

Major Subject: Nuclear Engineering

ABSTRACT

A remote laboratory for radiation detection and physical security was developed and implemented to provide students and professionals with education and training in nuclear security. Beginning with a proof-of-concept remote laboratory for radiation detection previously developed at Texas A&M University, additional radiation detection experiments were developed in conjunction with experiments for training in the use of physical protection systems. The radiation detection experiments include alpha spectroscopy, Compton scattering, Geiger-Müller (GM) dead time determination, gamma attenuation, high-purity germanium gamma spectroscopy, uranium enrichment quantification, the inverse square law, and scintillation detectors. A series of physical security experiments involves the use of sensors for surveillance and security applications, including using light sensors for material reflectivity, ultrasonic sensors for velocity determination, and infrared sensors for remote object measurements. The culmination of the remote laboratory curriculum is a greater understanding of how nuclear facilities secure their nuclear material. The remote laboratory has been, and will continue to be, used in an undergraduate radiation detection course. Future work for the remote laboratory can include additional radiation detection experiments, particularly in neutron detection, as well as improving the functionality by incorporating educational videos into the LabVIEW Virtual Instrument (VI) for each experiment.

ACKNOWLEDGEMENTS

I could not have completed this research without the support of my husband and my parents. Blaine, I am thankful for your endless love and patience that continue to allow me to pursue my professional interests. Mom and dad, I could not have asked for a better support system and role models throughout my life.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Dr. Craig Marianno and Dr. Sunil Chirayath of the Department of Nuclear Engineering and Dr. Dylan Shell of the Department of Computer Science and Engineering.

The proof of concept of this remote laboratory, discussed in Section 2.2, was published by Grant Emery of the Department of Nuclear Engineering in 2018. The SolidWorks designs in Section 4.4 and experiment tests and feedback in Section 5.2 - Section 5.4 were provided by Dalton Wise of the Department of Nuclear Engineering and Joshua Long of the Department of Civil Engineering. The procedures for the Alpha Attenuation experiment in Appendix B were provided by Jackson Wagner of the Department of Nuclear Engineering.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by the Graduate Diversity Excellence Fellowship from the Office of Graduate and Professional Studies and a Graduate Assistant Researcher position from the Department of Nuclear Engineering.

NOMENCLATURE

3D	3-Dimensional
ABS	Acrylonitrile Butadiene Styrene
ARM	Advanced RISC Machine
CAD	Computer-Aided Design
EV3	Evolution 3
FDM	Fused Deposition Modeling
GM	Geiger-Müller
GUI	Graphical User Interface
GX	Geometrical Expressions
HPGe	High-Purity Germanium
IDE	Integrated Development Environment
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
LME3	LEGO MINDSTORMS EV3
MCA	Multi-Channel Analyzer
NaI	Sodium Iodide
PIPS	Passivated Implanted Planar Silicon
PLA	Polylactic Acid
RAM	Random-Access Memory

RISC	Reduced Instruction Set Computing
ROI	Region of Interest
SDHC	Secure Digital High Capacity
STL	Stereolithography
VI	Virtual Instrument
VPN	Virtual Private Network

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

1.1. Motivation

Remote education capabilities provide an institution with the opportunity to teach students and train professionals outside of its geographical area. A challenge, however, is incorporating functional demonstrations and experiments that provide students and professionals with an interactive experience for greater understanding. Additionally, some institutions do not have access to the equipment and materials necessary to complete these experiments. This access to materials is further restricted within nuclear engineering and health physics programs; radiation sources are essential for completing radiation detection experiments, but there are legal, safety, and security realities that restrict the ease of access.

Under the current impact of the COVID-19 pandemic, remote education capabilities have proven to be an essential component of an institution's services. Transitioning courses and laboratories from the traditional, in-person format to an online interface is a challenge; the lack of direct communication and actions that are ordinarily present in a traditional educational setting are now replaced with chat windows, videos, and conference calls. Shifting a laboratory to an online interface is exceptionally more difficult since most of the learning in this setting is traditionally done through hands-on work and functional demonstrations. To effectively combat this issue, robotics will need to be incorporated into a virtual laboratory to give users the opportunity to move components and take measurements independently. This will mimic the hands-on work found in the traditional laboratory setting to further the student's understanding.

1.2. Research Objectives

There is a lack of remote laboratory education for students and professionals in the nuclear engineering and health physics disciplines. This challenge will be met by expanding upon a proof-of-concept remote laboratory for radiation detection education previously developed Grant Emery of Texas A&M University (Emery, 2018). The proof-of-concept remote laboratory provides a remote user access to industry-standard software, radiation detection instrumentation, and the control of radiation sources in three radiation detection experiments. The objective of the current research is to provide a remote laboratory for nuclear/radiological security education that emphasizes the use of sensors in physical protection systems. The experiments will involve the use of light sensors, ultrasonic sensors, and infrared sensors for material analysis and remote object measurements, and it will provide remote students with a more encompassing understanding of nuclear facilities outside of the laboratory. As an added benefit, the proof-of-concept remote laboratory will receive improvements to existing experiments as well as additional experiments to expand the scope of the radiation detection laboratory curriculum.

There is not currently a fully functioning remote laboratory for radiation detection that can be utilized by both students and professionals, and there certainly is not one that also includes sensor experiments for physical security education. After completion of this research, radiation detection courses in the Department of Nuclear Engineering at Texas A&M University can be taught fully online. Other institutions will also be able to make use of the remote laboratory for training their students and/or professionals in the case that they do not have access to certain equipment and materials. Eventually, this remote laboratory could even be used to develop experiments across various other disciplines, better equipping Texas A&M University to conduct remote laboratory education.

2. PREVIOUS WORK

2.1. Previous Remote Laboratory Work

There has been a recognized need for a reliable remote laboratory in colleges and universities. In 2010, a graduate student at the Georgia Institute of Technology designed and implemented a remote laboratory for teaching mechanical engineering concepts (Hyder, 2010). This Master's research introduced three remotely controlled experiments covering 3-Axis Gantry control systems, heat transfer by convection, and flywheel control systems. Using a remote desktop connection, a student was able to access a laboratory computer and, depending on the experiment, controlled hardware in real time using software such as LabVIEW or Armfield. A webcam and microphone mounted above the experiment provided the student with video and audio to further immerse the remote student in the experiment. A survey of students who completed the experiments showed that the remote experience was the same or better as the experience found in a traditional laboratory setting. Additionally, the majority of these same students felt that the video and audio helped to provide the same experience as that of a traditional laboratory setting.

The Center for Innovative Research in Cyberlearning (CIRCL) has made strides in remote laboratory education across the engineering disciplines by producing "iLabs" (Center for Innovative Research in Cyberlearning, 2020). More relevant to this research is their "Radioactivity iLab" that explores the inverse square law using a Geiger counter and a ^{90}Sr source (KAMU, 2020). Before the experiment, the student completes a virtual journal that provides an interactive lecture on the theory of the experiment. The virtual interface of the remote laboratory allows the student to adjust the distance between the Geiger counter and the ^{90}Sr source. In this experiment, ^{90}Sr source is on a vertically mounted linear slide with the Geiger counter placed directly underneath with its detector face positioned upwards. The interface also walked the student through the laboratory using built-in checkpoints, negating the need for written procedures. Lastly, the student observed the experiment in real-time via webcam; the connectivity allowed the student to view the apparatus in a laboratory in Australia. A survey of students who have completed the "Radioactivity iLab" showed that "students prefer the remote lab to a simulation," emphasizing the point that hands-on experience in a laboratory setting, whether virtual or traditional, increases the student's understanding (KAMU, 2020).

In 2011, a graduate student at Clemson University published his thesis research on the development of an online radiation detection laboratory course (Kopp, 2011). The purpose of the research was to create a remotely accessible radiation detection laboratory course for Clemson University's distance education program. Of the initial goal of 12 experiments, the remote laboratory featured 4 experiments – nuclear electronics, gamma-ray spectroscopy with scintillation detectors, gamma-ray attenuation and external dosimetry, alpha spectroscopy and absorption in air. The laboratory mimicked the hands-on movement of objects with a linear slide and a rotary table. The limitation of the Clemson remote laboratory is the lack of access to industry-standard software for the user. The experiments rely on custom-made LabVIEW programs that retrieve data from detector electronics, display the data within the LabVIEW interface, and export the data to a spreadsheet. Although this is advantageous for post-experiment data analysis, the user is not exposed to spectroscopy software that is standard in the industry and national laboratories. Additionally, the remote laboratory experiments were never tested by a group of students, so there is no feedback mentioned in the student's thesis on how well the system performs. Lastly, according to Clemson University's online degree programs, there is not an online degree program involving the use of this student's remote laboratory for radiation detection (Clemson University, 2020).

A remote laboratory for sensor education was developed through a collaboration of researchers from various Indian institutions (Ramya, Purushothama, & Prakash, 2020). The laboratory aimed to provide sensor education as applied to industrial automation applications. Using embedded systems and internet of things (IoT) as the platform, two sensor experiments were developed for remote education – proximity induction and the sorting of metal and non-metal elements for industrial applications. These experiments made use of a rotary motor to spin materials and a conveyor belt to imitate sliding objects down an assembly line. The experimental apparatus was viewed through an internet protocol (IP) camera that sent audio and video to the remote user. The user was able to access the remote laboratory and complete the experiment with an Android mobile application developed specifically for this remote laboratory.

2.2. Proof-of-Concept Remote Laboratory

A remotely accessible laboratory for radiation detection education was developed by a graduate student at Texas A&M University in 2018 (Emery, 2018). This proof-of-concept remote laboratory will serve as the framework of the research discussed in this thesis. A linear slide and a rotary table were used to manipulate elements of the laboratory, whether they be sources or detectors. The laboratory was accessed through a remote desktop connection so that the user could use industry-standard software, view the laboratory’s camera feed to observe the experiment, and operate LabVIEW programs specific to operating the linear slide and rotary table in each experiment. There were three experiments developed – gamma source identification, uranium enrichment quantification, and Geiger-Müller (GM) dead time determination and statistical analysis.

3. METHODOLOGY

Physical security experiments for sensor education will be developed and added onto the radiation detection component of the laboratory to provide nuclear security elements to the remote laboratory. A series of three related physical security experiments will introduce the student to key concepts that are essential to understanding how sensors operate in a facility's physical protection system. The end goal of the series of experiments is for the student to understand how light reflectivity and object movement are used to accomplish security objectives.

The first experiment explores light reflectivity which is an important concept to understand when working with light-based sensors. For example, infrared sensors measure heat and motion with the basis of its operation rooted in measuring the reflection of emitted infrared light. To introduce students to this concept, a student-friendly light sensor will need to be selected. An experiment utilizing this sensor will demonstrate how heat is absorbed in materials of different colors and properties. The amount of heat that is absorbed is indicative of the amount of reflected light that is returned to the light sensor. From this experiment, the student will recognize the importance of material considerations when light-emitting sensors are utilized.

The second physical security experiment focuses on movement. The remote measurement of moving objects allows a facility to determine if secure areas have been breached. Movement can be detected using microwave sensors, ultrasonic sensors, or even light-emitting sensors such as infrared. The value of monitoring movement around a secure area is the ability for a facility to have continuous monitoring for identifying a security breach. As sound waves are emitted from a sensor, the waves are reflected from nearby objects and returned to the sensor. The sensor can use the elapsed time between each reflected wave to determine how far away a nearby object is located. If an object approaches too closely to a secure area or breaches a boundary, a facility's physical protection system can alert authorities to respond. An experiment will need to be designed that employs a student-friendly ultrasonic sensor to introduce students to detecting motion. From this experiment, the student will recognize the importance of detecting motion and being able to ascertain the velocity of a moving object with a sensor.

The final experiment of the series will serve as the culmination of the concepts learned in the light sensor and ultrasonic sensor experiments. An important capability for a security system is to remotely monitor an entry point into a secure area. If the velocity of a moving object was already determined by using an ultrasonic sensor, a physical protection system can use a light-emitting sensor placed perpendicular to the moving object to measure its length. This will allow the student to understand the value in a facility's capability of determining what is moving in and out of a secure area. The light-emitting sensor should be an infrared sensor since it is able to detect reflected light and determine the distance between it and an object. The student will recognize the importance of light reflectivity by measuring objects with different material properties. The light emitted by the infrared sensor will be absorbed differently by each material type, thereby affecting the measurement of the object. This portion of the experiment will serve to prove the importance of material considerations when using light-based sensors.

The light sensor, ultrasonic sensor, and infrared sensor experiments will each need their own procedures and apparatus designed for remote education. Videos will be produced for each sensor experiment to provide background on the sensor, such as how it works, how it is used for physical security, and how to operate it using basic programming. To adhere to disability services, the videos will have companion transcripts to accommodate students with hearing disabilities. The experiments will need custom-made platforms and objects to complete the experiment objectives. For example, the ultrasonic sensor experiment will require the use of a wall to reflect ultrasonic signals. Similarly, the infrared sensor experiment will require the use of an irregularly shaped object. Both of these components will also need to fit seamlessly on either a rotary table or a linear slide, requiring custom-made platforms or more intricate object designs.

In addition to developing physical security laboratories, part of this research will improve the functionality of an existing proof-of-concept remote radiation detection laboratory. In the gamma source identification experiment shown in Figure 1, there is not adequate shielding set in place to keep unwanted gamma rays from interfering with source measurements. This will include adding shielding around sources and collimating detectors. The shape of the rotary table will need to be changed from an octagon to a circle to avoid the possible interference of corners during rotation.

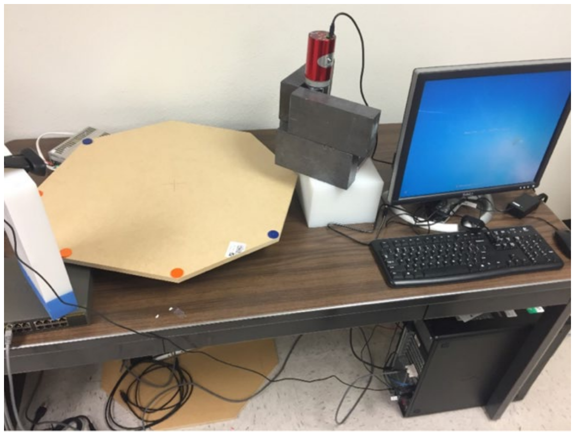


Figure 1. Gamma source identification experiment in the proof-of-concept remote laboratory for radiation detection education (Emery, 2018).

In the uranium enrichment quantification experiment shown in Figure 2, there are very few improvements that need to be made. There are too many sources for a lead brick to be placed in between each source without exceeding the weight restriction on the motor, as specified in Emery's thesis. (Emery 2018). The existing apparatus will be maintained.

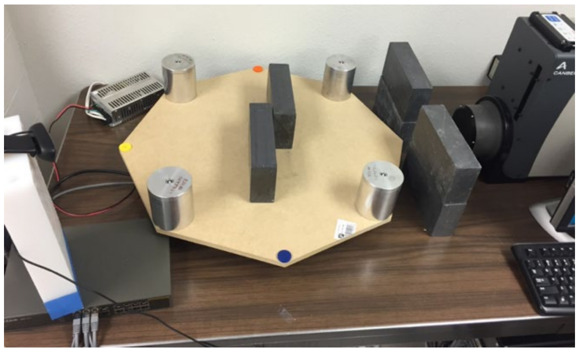


Figure 2. Uranium enrichment quantification experiment in the proof-of-concept remote laboratory for radiation detection education (Emery, 2018).

In the GM dead time determination and statistical analysis experiment shown in Figure 3 and Figure 4, the apparatus is not professional as the GM probe is duct taped to the linear slide and a compact-disk case is used to hold up another component. The apparatus also fails in the way that the measurements are taken; the two half-sources are measured at different angles with respect to the GM detector face, and the lead brick interferes with the source and detector because of its length and inconvenient placement. Custom-made platforms will be designed and three-dimensionally (3D) printed for the GM detector probe and the half-sources, and an entirely new apparatus will be designed for the experiment.

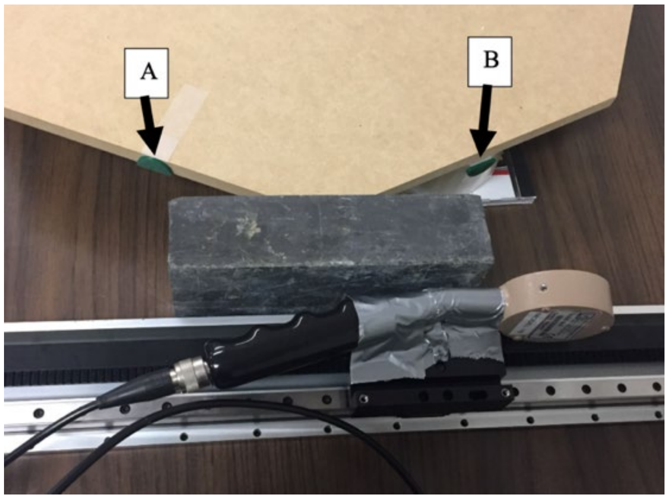


Figure 3. GM dead time determination and statistical analysis experiment in the proof-of-concept remote laboratory for radiation detection education. This is the apparatus when half-source A and half-source B are measured separately (Emery, 2018).

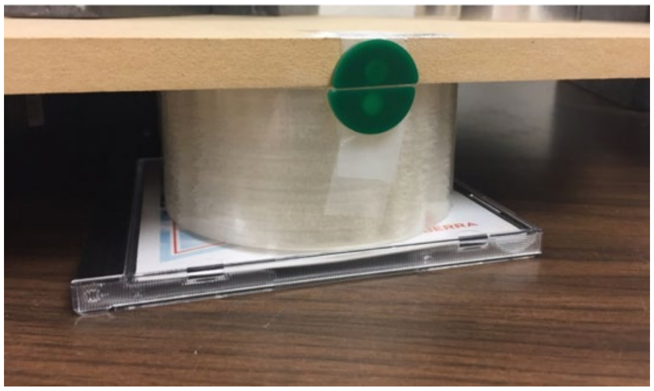


Figure 4. GM dead time determination and statistical analysis experiment in the proof-of-concept remote laboratory for radiation detection education. This is the apparatus when half-source A and half-source B are measured as a whole source (Emery, 2018).

Additional radiation detection experiments that are used for in-person classes will be converted to remote formats to supplement the radiation detection curriculum of the remote laboratory. Existing in-person procedures will be converted into remote procedures. The existing in-person procedures will also be used to design a suitable apparatus for the remote laboratory. The apparatus will be different for each experiment, but they will all include custom-made platforms to correctly hold sources and equipment.

For both the physical security and radiation detection experiments, applicable experiments will be given foamboard layout templates. These foamboards will be placed on top of the rotary table and outline the shapes of all of the equipment and other components. This will give the teaching assistant a visual set of instructions on how to set up each experiment. The foamboards will also help maintain the geometry of the physical apparatus between set ups, resulting in the normalization of sources of error between students no matter when they are completing the experiment.

4. SYSTEM COMPONENTS

4.1. Host Computer

The host computer is responsible for performing all of the remote laboratory capabilities. Using the Microsoft Remote Desktop application, students will access this computer to open and operate industry standard software, control laboratory components, and view the live-time camera feed. The remote laboratory employs the Lenovo ThinkCentre M720q Tiny Computer running Windows 10 with the ThinkCentre Tiny-In-One 24" Display (Lenovo, 2020). The full specifications of the computer are shown in Table 1.

Table 1. Technical specifications of the host computer.

Processor	Intel Core i7 8700T, 2.4GHz 6 Core
Operating System	Windows 10 Enterprise
Memory	32 GB
Chipset	Intel I350
Internet Ports	T4 Quad Port Gigabit NIC (5 Ethernet Ports Total)

4.2. LEGO MINDSTORMS EV3 Education Core Set

The LEGO MINDSTORMS EV3 (LME3) Education Core Set is a system designed to introduce students to building and programming for engineering and robotics applications (LEGO Education, 2020). This core set was chosen for the physical security experiments because it was designed to be easy to use and for educating students in science and technology. The core set includes the Evolution 3 (EV3) Intelligent Brick, (3) servo motors, (5) sensors (gyro, ultrasonic, color, and (2) touch), an EV3 rechargeable DC battery with charging cable, connecting cables, and a variety of LEGO assembly parts. All components of this core set are stored in the provided storage bin. The sensors included in the set are not actually employed in security facilities; however, they perform well enough to teach concepts to students. Of the components included in the core set, the servo motors, gyro sensor, and touch sensors are not included in any of the physical security experiments designed for the remote laboratory. The LME3 Education Core Set is shown in Figure 5.

Figure 5. LEGO MINDSTORMS EV3 Education Core Set.

4.2.1. EV3 Intelligent Brick

The host computer can communicate with the EV3 Intelligent Brick via USB connection or Bluetooth (LEGO Education, 2020). To avoid Bluetooth connectivity issues that may arise in a student's remote laboratory session, the host computer communicates with the Brick via USB connection. The Brick features a USB port, (4) input ports, (4) output ports, a high-resolution 178x128 pixel black and white display, a built-in speaker, and an interface with (6) buttons. Additional specifications are shown in Table 2, and the Brick is shown in Figure 6-Figure 8.

Table 2. Specifications of the EV3 Intelligent Brick (LEGO Education, 2020).

Processor	ARM 9
Operating System	Linux
Flash Memory	16 MB
RAM	64 MB
Expanded Memory	32 GB (using Mini SDHC card reader)

The student operates the Brick by uploading a file to its library. The file can continue to be updated and tested throughout the coding process, allowing for a seamless experience even for a remote student. The Brick is able to be operated using Python, LabVIEW, and 3rd Party programs; however, current LEGO support is focused on the utilization of Python programs. Data is collected through its input ports at speeds of up to 1000 samples per second, and the data can be observed either on the EV3's built-in display or on the host computer. For remote laboratory students, the data will be observed in the Visual Studio Code IDE.

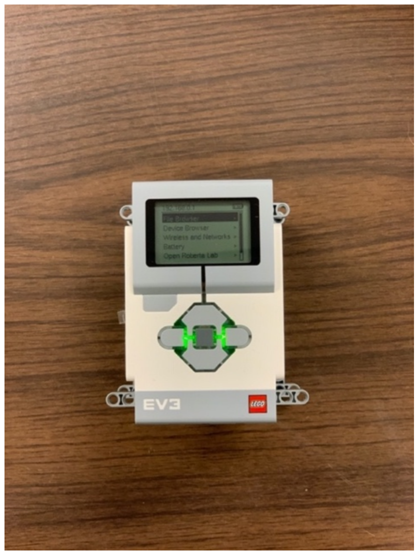


Figure 6. EV3 Intelligent Brick interface.

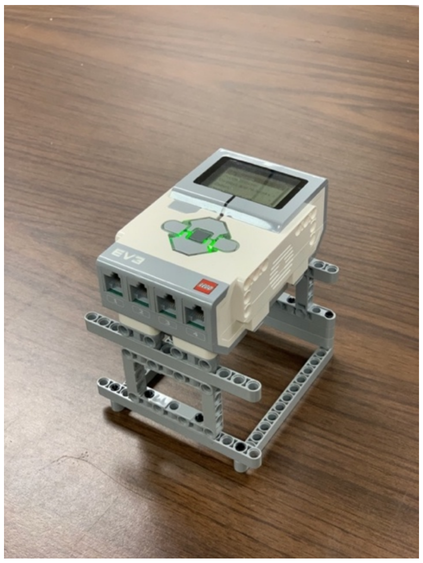


Figure 7. EV3 Intelligent Brick sensor ports.

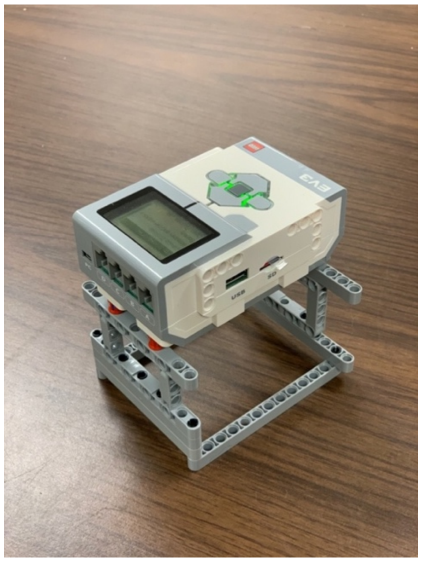


Figure 8. EV3 Intelligent Brick, showing the motor ports, PC port, USB port, and SD card port.

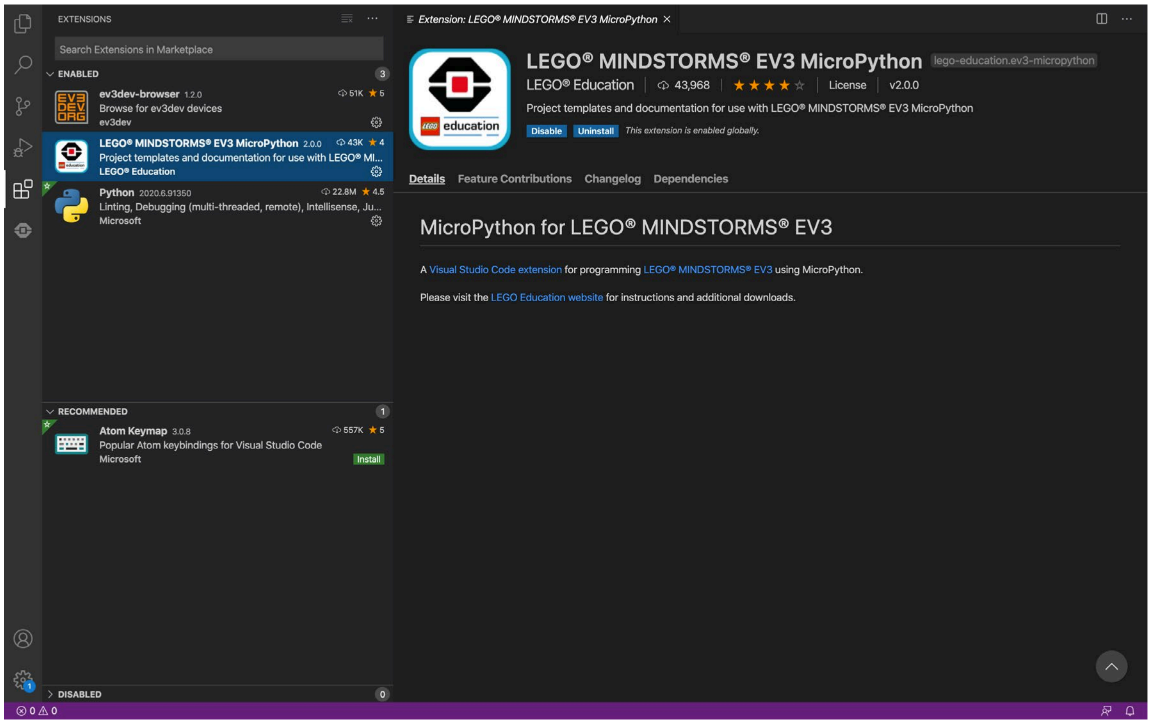
4.2.2. EV3 MicroPython and Visual Studio Code

LEGO Education developed the EV3 MicroPython programming language, an optimized version of Python specific to programming MINDSTORMS EV3 motors, sensors, and other components (LEGO Education, 2019). Visual Studio Code is the integrated development environment (IDE) that executes EV3 MicroPython programs and contains the necessary libraries for operation (Microsoft, 2020). Outside of EV3 applications, MicroPython is an open-source version of Python that is ordinarily used to run on microcontrollers (George, 2014). By creating EV3 MicroPython, LEGO Education provided students the opportunity to learn fundamental Python programming that is applied to machine learning and artificial intelligence concepts.

To use EV3 MicroPython on the EV3 Intelligent Brick, the EV3 MicroPython image provided by LEGO Education must be flashed onto a microSD card and inserted into the EV3 Intelligent Brick. A flashing tool such as Etcher is required to flash the image onto a microSD card (Etcher, 2019). As long as the microSD card is inserted into the Brick, the system will operate using EV3 MicroPython. Switching back to using the standard LME3 firmware is as simple as removing the microSD card and rebooting the Brick (LEGO Education, 2019).

The host computer has Visual Studio Code installed on it, and it will be required that each student install it on their own machine. Visual Studio Code is needed to develop and execute the Python programs necessary for completing the physical security experiments in the remote laboratory. LEGO Education developed EV3 MicroPython to be compatible solely with Visual Studio Code, so it is within this IDE that all of the accessibility, libraries, and other features exist. Students will be required to have it installed on their own machines to complete pre-laboratory assignments using the same software as the host computer. A part of the pre-laboratory assignment involves watching an instructional video on the installation and use of Visual Studio Code and EV3 MicroPython, as will be discussed in Section 4.6.

After installing Visual Studio Code on the host computer, LME3 MicroPython has to be installed as an extension. This extension provides the features necessary to create LME3 projects, access EV3 MicroPython libraries, and connect to the EV3 Intelligent Brick (LEGO Education, 2019). A screenshot of the LME3 MicroPython extension installation is shown in Figure 9.



Browse for ev3dev devices

Project templates and documentation for use with LEGO® MI...

Linting, Debugging (multi-threaded, remote), Intellisense, Ju...

Popular Atom keybindings for Visual Studio Code

Project templates and documentation for use with LEGO® MINDSTORMS® EV3 MicroPython

This extension is enabled globally.

MicroPython for LEGO® MINDSTORMS® EV3

A Visual Studio Code extension for programming LEGO® MINDSTORMS® EV3 using MicroPython.

Please visit the LEGO Education website for instructions and additional downloads.

Figure 9. Installation of the LEGO MINDSTORMS EV3 MicroPython extension in Visual Studio Code.

4.2.3. EV3 Color Sensor

The EV3 Color Sensor can provide a reflected light intensity value by measuring reflected red light and ambient light (LEGO Education, 2020). For the purposes of this work this device will be referred to as a light sensor rather than a color sensor. The EV3 Color Sensor detects the intensity of the reflected red light and ambient light in units of Lux, with 0 Lux being characteristic of complete light absorption and 100 Lux being characteristic of complete light reflection. Figure 10 shows the EV3 Color Sensor. The manufacturer did not report an uncertainty associated with the performance of the EV3 Color Sensor.

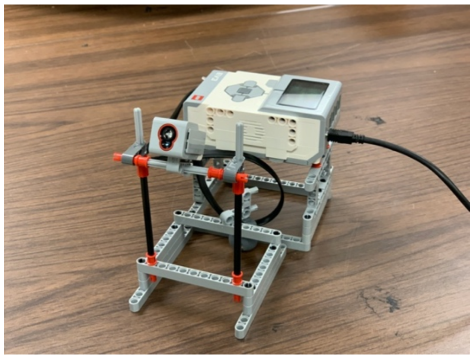


Figure 10. EV3 Color Sensor connected to the EV3 Intelligent Brick.

4.2.4. EV3 Ultrasonic Sensor

The EV3 Ultrasonic Sensor emits sound signals and measures the time between emission and reflection to estimate the distance between itself and an object (LEGO Education, 2020). The sensor can reportedly measure distances of up to 250 centimeters with an uncertainty of ± 1 cm. This sensor outputs the measured distance in units of millimeters. Figure 11 shows the EV3 Ultrasonic Sensor.

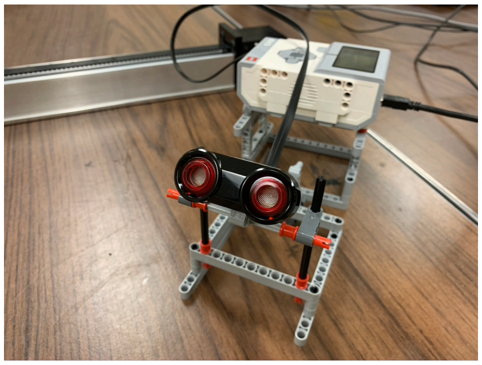


Figure 11. EV3 Ultrasonic Sensor connected to the EV3 Intelligent Brick.

4.2.5. EV3 Infrared Sensor

The EV3 Infrared Sensor measures the reflection of an emitted infrared signal to measure distance (LEGO Education, 2020). This sensor outputs the measured distance in units of millimeters. The manufacturer reports that the sensor measures proximity measurements up to 100 cm depending on the color and material type of the surface reflecting the infrared signal. The manufacturer did not report an uncertainty associated with the performance of this device. It is necessary to note that the EV3 Infrared Sensor does not come with the LME3 Education Core Set, so it must be purchased separately for the remote laboratory. Figure 12 shows the EV3 Infrared Sensor.

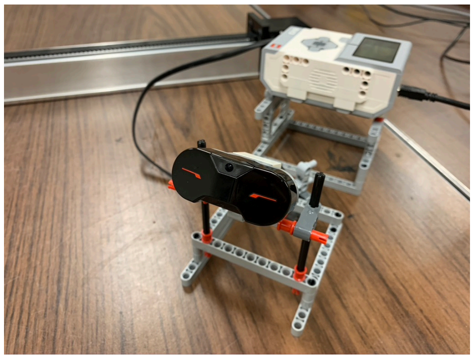


Figure 12. EV3 Infrared Sensor connected to the EV3 Intelligent Brick.

4.2.6. LEGO Bricks

The LME3 Education Core Set contains a sorting tray filled with LEGO bricks and numerous other LEGO parts. It is advantageous to have these LEGO parts for the construction of platforms that hold the EV3 Intelligent Brick and sensors, as shown in Figure 6-Figure 8 and Figure 10-Figure 12. The LEGO parts included in the LME3 Education Core Set are shown in Figure 13, and the corresponding labelled tray of parts is shown in Figure 14.

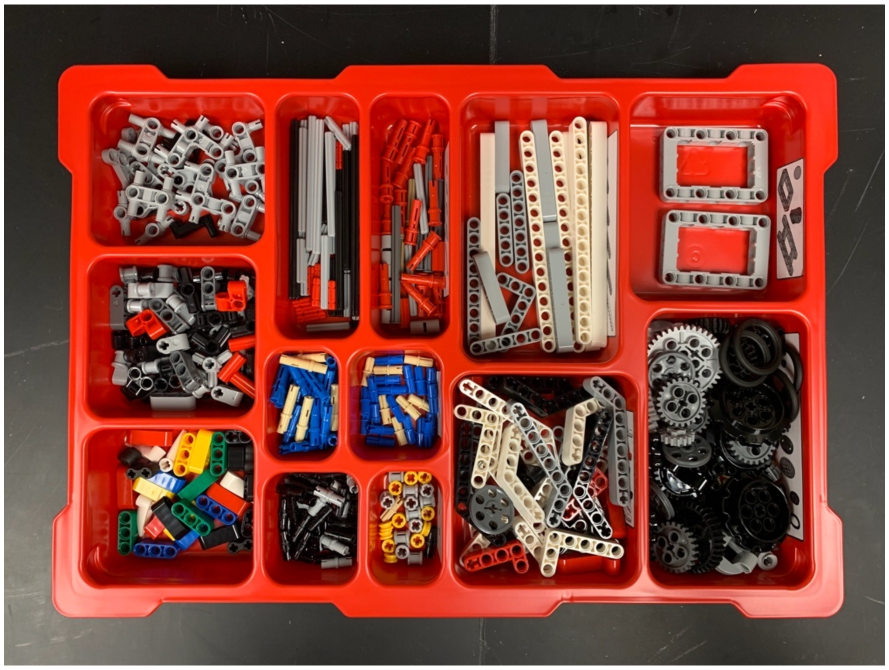


Figure 13. LEGO MINDSTORMS EV3 Education Core Set tray of LEGO bricks and parts.

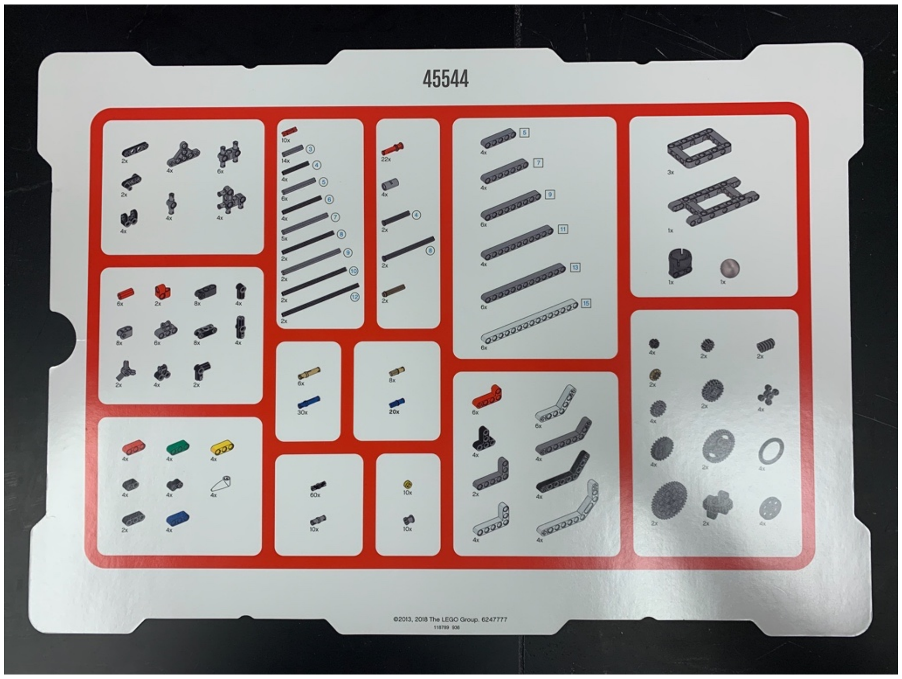


Figure 14. Corresponding labelled tray of LEGO bricks and parts.

4.3. 3D Printing

A constraint in any remote laboratory is preserving the geometry and accessibility found in a traditional laboratory setting. Remote versions of radiation detection experiments and physical security experiments were made possible by 3D printing custom made platforms and objects that satisfy an experiment's geometrical constraints and overall learning objectives.

4.3.1. SOLIDWORKS

The 3D renderings of platforms were designed using the Texas A&M University License of the Education Edition of SOLIDWORKS 2019. SOLIDWORKS is a computer-aided design (CAD) program developed by Dassault Systèmes that is used for solid modeling in engineering applications (Dassault Systèmes, 2019). The Education Edition is an integrated version of the SOLIDWORKS Suite to simplify the user experience for students.

In SOLIDWORKS, a Part is “a 3D representation of a single design component” whereas an Assembly is “a 3D arrangement of parts and/or other assemblies.” Additionally, there is the option to create “a 2D engineering drawing, typically of a part or assembly”, known as a Drawing. In all design scenarios, a Part or an Assembly was created depending on the overall technical specifications of a needed laboratory component (Dassault Systèmes, 2019).

After producing a 3D rendering of a laboratory component, the design was saved with the stereolithography (STL) file extension, denoting it as a “neutral file format designed such that any CAD system can feed data to the rapid prototyping process” (Grimm, 2004). An STL file is the composition of the geometry’s coordinates in a triangular mesh, and it approximates the geometry so that a slicing software can convert it into a geometrical expressions (GX) file.

4.3.2. Flashforge Guider II 3D Printer

After a Part or Assembly was designed using SOLIDWORKS, the design was produced with the Flashforge Guider II 3D Printer. The Flashforge Guider II is an extruder-type, industrial-grade 3D printer capable of printing with PLA, Acrylonitrile Butadiene Styrene (ABS), and other specialty filament types (FlashForge Corporation, 2017). It has a build volume of 280x250x300 mm with a positioning precision of 11 microns for the X and Y coordinates and 2.5 microns for Z coordinates. It operates using a single, 0.4 mm diameter extruder with a layer resolution of ± 0.2 mm. The printer also features connectivity via USB drive, USB cable, WiFi, and Ethernet. The Flashforge Guider II 3D Printer is shown in Figure 15.

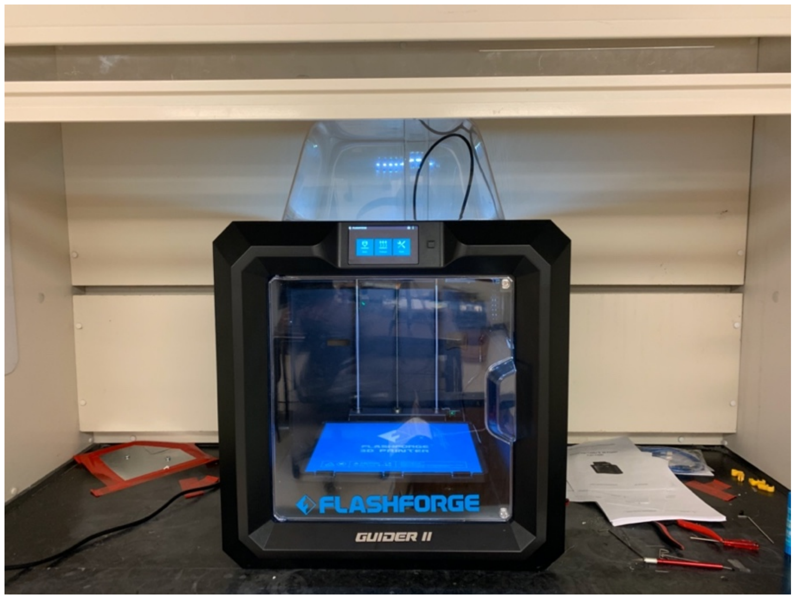


Figure 15. Flashforge Guider II 3D Printer.

Fused Deposition Modeling (FDM) is the 3D-printing technique used by the Flashforge Guider II. In this technique, an extruder at its operating temperature, which varies depending on the filament type, moves in the XY plane as the printing platform moves vertically along the Z axis (Chennakesava Sai & Shivraj Narayan, 2014). The filament is fed out of the moving extruder in a melted state and onto the moving platform to form layers. These layers cool and harden to build the solid 3D design. An important component of the technique is the use of support material to make the building process possible. In order to place filament in a region without any underlying layers of solid filament, support structures are positioned beforehand so that the filament can be placed accordingly. The Flashforge Guider II does not have a separate spool of support material and therefore does not have the additional extruder; it uses the build material at a much lower density. Supports are essential for preserving structural integrity, and they are eventually removed by breaking or dissolving them after printing (Schmidt & Umetani, 2014).

Polyactic Acid, more commonly referred to as PLA, was the filament material employed to construct the laboratory components. It was selected because of its low cost and respectable strength, lifetime, and dimensional accuracy (Simplify3D, 2020). The material was consistently extruded at a temperature of 220 °C and a variable printing speed of between 60-90 mm per second depending on the design. Acrylonitrile Butadiene Styrene, more commonly referred to as ABS, is another popular filament material used for 3D printing; however, it was not used to print components for the remote laboratory because of its tendency to shrink, resulting in dimensional inaccuracy (Simplify3D, 2020).

The Flashforge Guider II uses its in-house slicing software called FlashPrint to convert STL files into GX files (FlashForge Corporation, 2020). GX files convert the geometry in STL files into algebraic expressions that a 3D printer can use to position its extruder accordingly during the printing process. More specifically, this slicing software converts the geometry in the STL file into numerous thin layers that produce instructions for the printer to follow. These instructions for the printer include the extruder speed needed for each layer, the temperature to use, and how much filament to extrude (Simplify3D, 2020). An important requirement for the user in this step of the process is to ensure the correct operational temperatures for the extruder and the printing platform. This is also the step where branching supports are enabled. Once FlashPrint converts the STL file into the GX file, the GX file can be uploaded to the Flashforge Guider II for printing using any of the aforementioned connectivity pathways.

4.3.3. 3D Renderings of Experiment Components

A number of experiment components were designed in SolidWorks and produced with the Flashforge Guider II. These components were needed to support the learning objectives of remote experiments and establish necessary geometrical requirements. The following renderings were designed using custom measurements for seamless integration into each experiment apparatus. The files of all 3D renderings will be provided so that the components can be printed in the future as needed.

4.3.3.1. Physical Security Experiments

The ultrasonic sensor experiment requires the use of a wall that will bounce ultrasonic signals back towards the sensor for distance measurements. This wall will need to be placed on the linear slide so that it can be moved towards and away from the ultrasonic sensor. Figure 16 - Figure 18 show different views of the ultrasonic sensor wall. For a seamless fit on the linear slide, measurements of the linear slide base were taken with calipers. The legs of the wall base that moves on the linear slide have the same measurement of 0.70 cm, the length of the base is 9.90 cm, and the height of the base is 2.30 cm. The wall itself has a width of 1 cm, a length of 10 cm, and a height of 6 cm. It features fillet corners for a sleek design; these corners do not offer any sort of functional advantage.

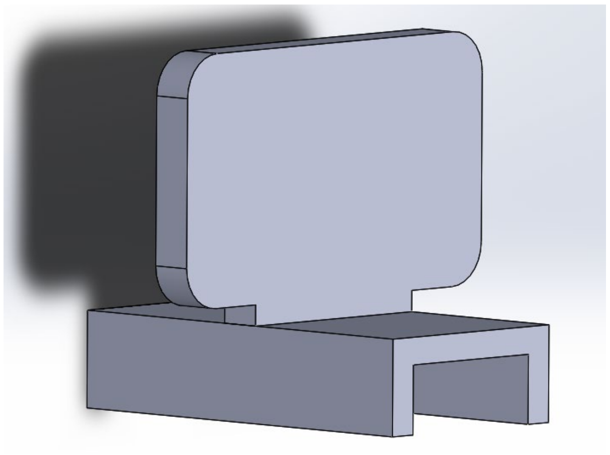


Figure 16. Angled view of the ultrasonic sensor wall.

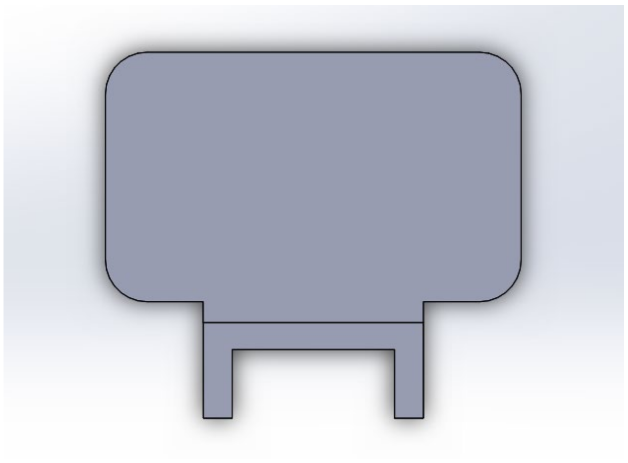


Figure 17. Front view of the ultrasonic sensor wall.

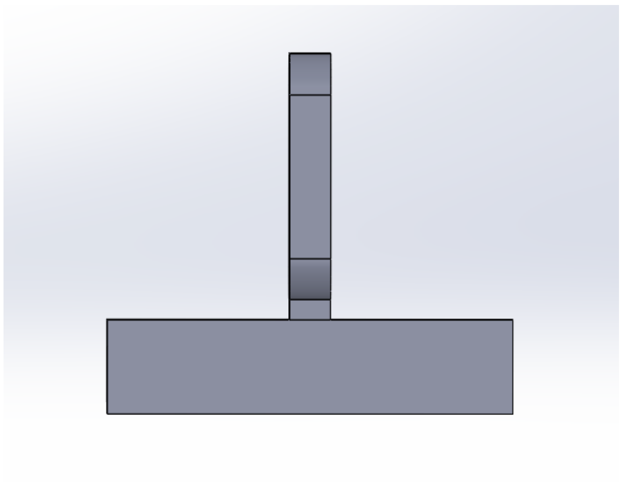


Figure 18. Side view of the ultrasonic sensor wall.

The infrared sensor experiment requires the use of an irregular-shaped object that mimics the presence of multiple objects travelling past the sensor. A wall that resembles a comb with missing teeth was used as the design for this component, and the length of each tooth in this comb as well as the space between each tooth were arbitrarily chosen. The varied length of each tooth results in a design that mimics spaced apart objects of different sizes. The infrared sensor wall is shown in Figure 19 - Figure 21. This wall also needed to be placed on the linear slide, so it uses the same base measurements as the ultrasonic sensor wall. Refer to Figure 22 and Table 3 for the length of each region of interest (ROI) of the infrared sensor wall. Each constituent is 5 cm high and 1 cm thick with an overall wall-length of 30 cm.

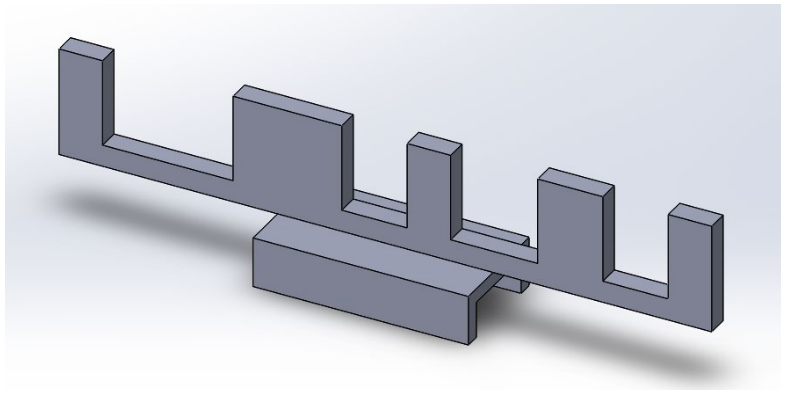


Figure 19. Angled view of the infrared sensor wall.

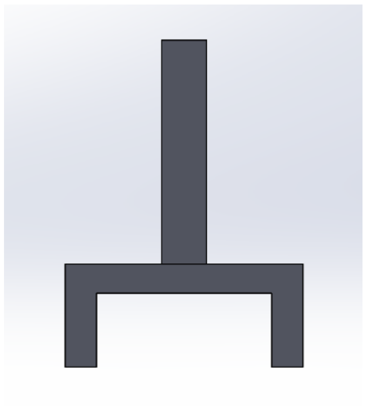


Figure 20. Front view of the infrared sensor wall.

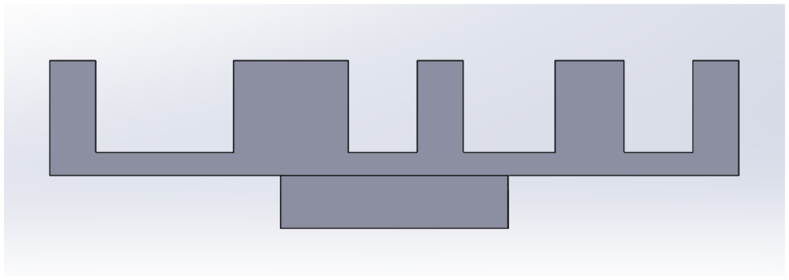


Figure 21. Side view of the infrared sensor wall.

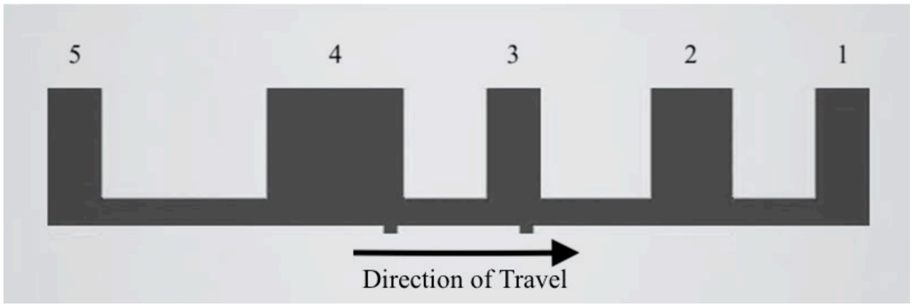


Figure 22. Infrared sensor wall labeled with ROIs.

Table 3. ROI measurements of the infrared sensor wall with uncertainties specified by the caliper's manufacturer (General, 2020).

ROI	Nominal Measurement (mm)	Actual Measurement (mm)
1	20.0	20.19 ± 0.04
2	30.0	30.19 ± 0.04
3	20.0	20.25 ± 0.04
4	50.0	50.10 ± 0.04
5	20.0	20.04 ± 0.04

4.3.3.2. Radiation Detection Experiments

The GM Dead Time Determination and Counting Statistics experiment needed a component to hold a GM probe above sources as they are measured. The design of the holder is shown in Figure 23 - Figure 24 with Figure 23 being the back piece and Figure 24 being the front piece. The back piece has an angled extrusion cut to fit the handle of the GM probe detector, and it also features two openings for insertion. The front piece has a custom-made tray for the GM detector face to lay in and an opening to allow for measurements of a source placed below it. The front piece also has two prongs that insert into the two openings on the back piece. The separation of the overall design was necessary for reliable 3D printing. The final assembly of the printed components with a GM probe is shown in Figure 25.

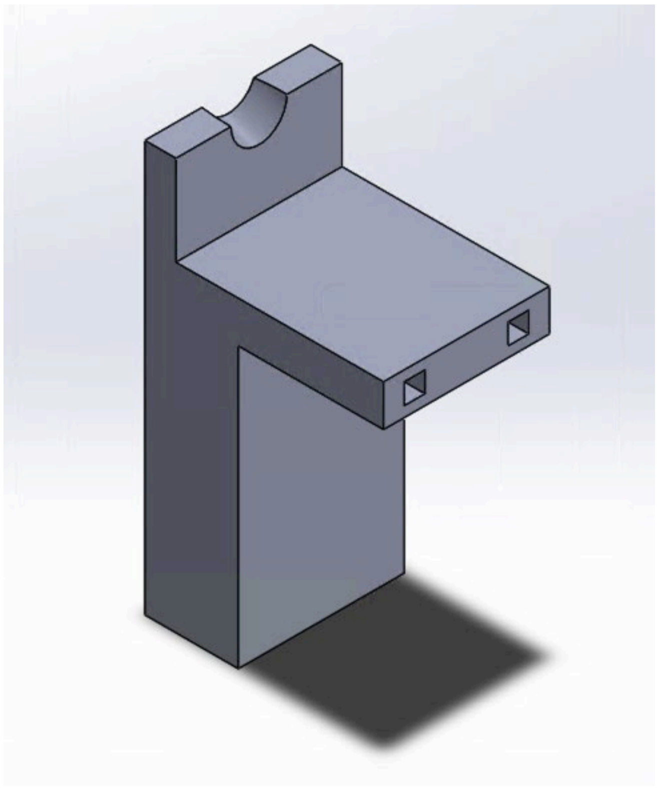


Figure 23. 3D rendering of the back piece of the GM detector probe holder.

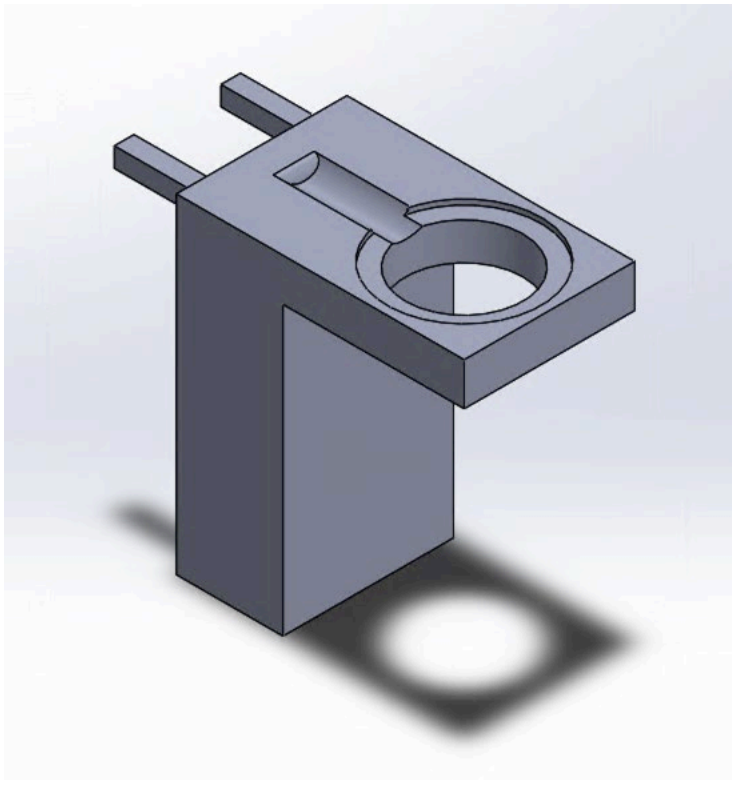


Figure 24. 3D rendering of the front piece of the GM detector probe holder.

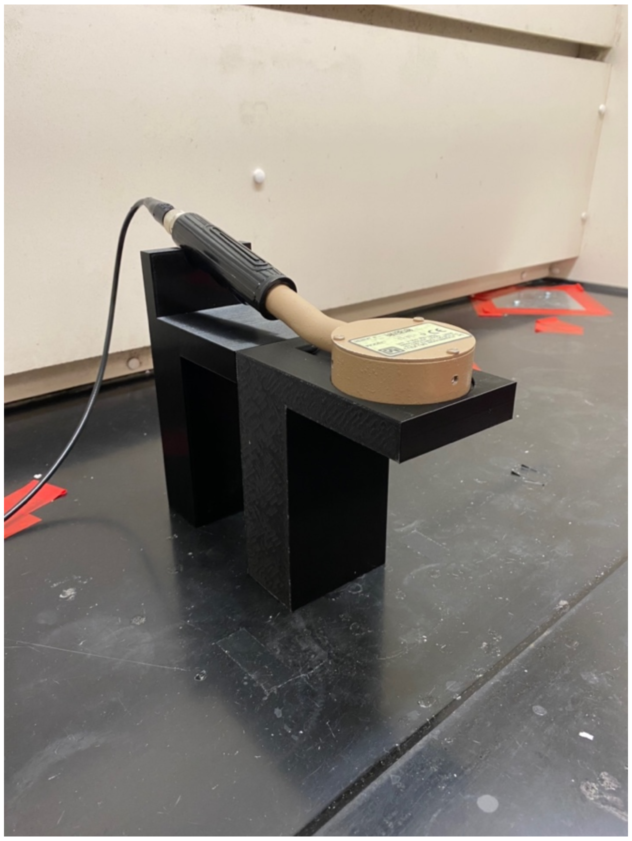


Figure 25. The completed assembly of the GM detector probe holder with the front piece inserted into the back piece.

The GM Dead Time Determination and Counting Statistics experiment also required platforms for two half sources under the face of the GM detector probe. For this apparatus, a source platform needed to be designed for both the linear slide, shown in Figure 26, and for the rotary table, shown in Figure 27. The base of the source platform for the linear slide uses the same measurements as the base of both the ultrasonic sensor wall and the infrared sensor wall to seamlessly integrate with the linear slide. Both of these source platforms feature trays for the placement of a check source that are extended 10 cm away from the base. Once in place, the sources are 2.5 cm directly below the face of the GM probe.

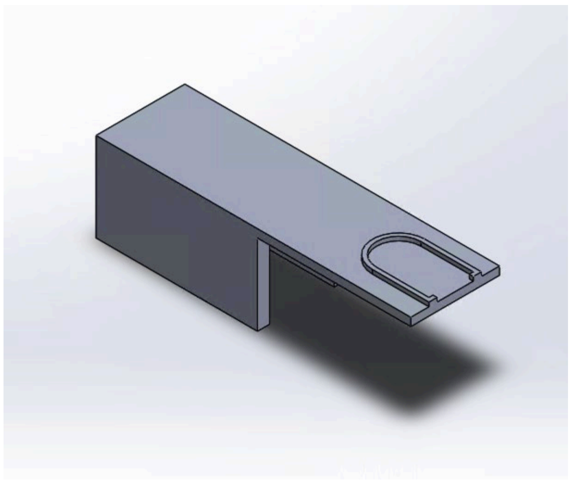


Figure 26. 3D rendering of the linear slide source platform for the GM Dead Time experiment.

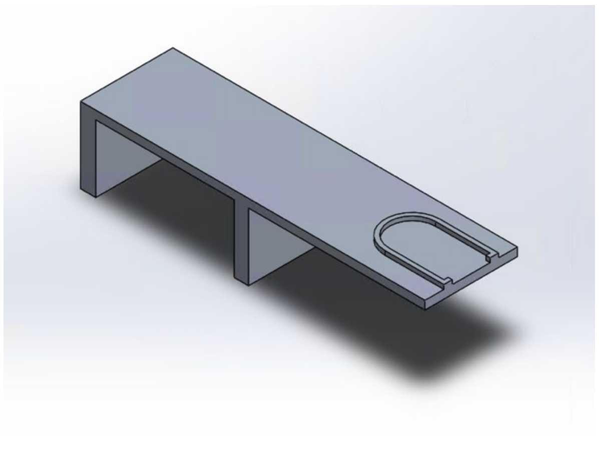


Figure 27. 3D rendering of the rotary table source platform for the GM Dead Time experiment.

An alternative rotary table source platform was constructed and used for other experiments that involved check sources. A table-like structure was designed to hold up to three, stacked check sources simultaneously (Figure 28). The three short prongs along the rim of the circular tray hold the sources in place, and they were designed this way to reduce the amount of material needed for printing. This can also be said about the legs and the circular cutout. This source platform was designed to hold check sources at the vertical height that aligns with the center of the sodium iodide (NaI), lanthanum bromide (LaBr), and HPGe detector faces used in the other experiments.

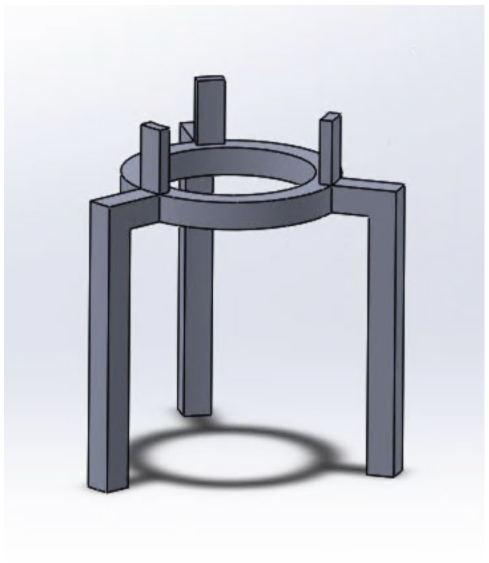


Figure 28. 3D rendering of a general source platform for experiments that use the rotary table.

The Attenuation Coefficients experiment needed a holder for various thicknesses of attenuator slabs. In this experiment polyethylene, lead, copper, and aluminum of various thicknesses are used to attenuate photons. To hold the different-sized elements, three holder sizes were designed: small, medium, and large. Each design depended on the needed thickness of the attenuators. To provide a seamless fit for each holder, a screwed press was designed to firmly hold the slabs. This forces the attenuators to stand upright and mitigate the possibility of slanting. An example of an attenuator holder with the opening for a screw is shown in Figure 29, and the screw design is shown in Figure 30.

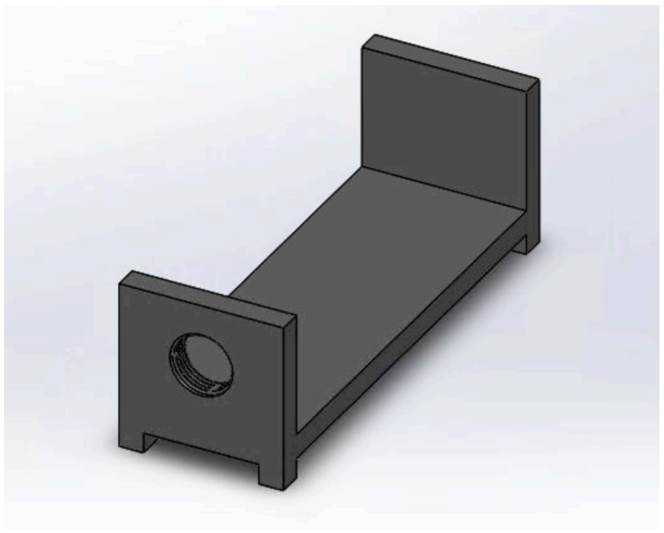


Figure 29. 3D rendering of a large sized attenuator holder without the screw.

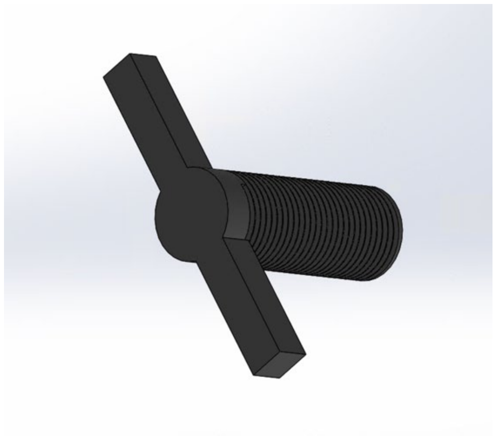


Figure 30. 3D rendering of the screw that will be inserted into an attenuator holder.

4.4. Foamboard Layout Templates

Foamboard layout templates were designed for select experiments to maintain the configuration of the physical apparatus between set ups. If not used, the apparatus may be set up with slightly different geometries between students, resulting in deviations in experimental results. The layout templates consisted of underlying, traced-out foamboards that the system components were placed on. For experiments that used the rotary table, circular layout templates were cut out of the middle portion of a white, 28"x40" Elmer's Tri-Fold Foam Display Board (one layout board per foam display board); the circular layout templates have diameters equal to the rotary table, 24 inches (Walmart, 2020). The measurements and overall shapes of laboratory components were traced on the foamboards to provide future teaching assistants with the correct measurements between components.

4.5. Instructional Videos for Physical Security Experiments

An important component of the physical security experiments was the introduction to Python programming. Therefore, instructional videos were developed to introduce students to MicroPython, the setup of Visual Studio Code, and the setup of the programs required to complete each of the experiments. A total of (4) videos were created for students to watch before their assigned laboratory time: *Introduction to LEGO MINDSTORMS Education EV3*, *Introduction to the Light Sensor Experiment*, *Introduction to the Ultrasonic Sensor Experiment*, and *Introduction to the Infrared Sensor Experiment*. Each video had a companion transcript made to accommodate students with hearing disabilities.

The *Introduction to LEGO MINDSTORMS Education EV3* video provides details of the technology and software that are used to complete the physical security experiments. It begins with a short discussion of the LEGO Core Set, familiarization with the EV3 MicroPython programming language, and a walk through of the setup of Visual Studio Code to connect to the EV3 Intelligent Brick.

The videos specific to each of the (3) physical security experiments provide details of the physics behind each sensor, a brief introduction to the experiment apparatus and procedures, and a line-by-line walkthrough of the code development needed to complete each experiment. The physical security experiments are supposed to emphasize an education in sensor fundamentals rather than programming, so the instructional videos walk the students through each line of the code so that they will enter the remote laboratory prepared. The code is commented in each of the line-by-line walkthroughs to instill healthy programming habits and attempt to improve understanding of the functions and concepts being used. The presentation slides and accompanying video transcripts for each video are provided in Appendix A.

4.6. Components from Emery's Remote Laboratory

Many of the components employed by Emery's remote radiation detection laboratory are used for this research (Emery, 2018). These include:

- The Mirion Technologies GENIE 2000 software for gamma spectroscopy (Mirion Technologies, 2016);
- The National Instruments LabVIEW software to control motors (National Instruments, 2016);
- The Ethernet stepper motors to move laboratory components (LinTech Motion Control Inc., 2020);
- The existing rotary table and linear slide;
- The Logitech C270 webcam to provide a video feed to the remote student (Logitech, 2020).

4.7. Determination of Linear Slide Velocity

The velocity of the linear slide at different input values was needed so that the remote student can use them in the physical security experiments. Unfortunately, the linear slide did not come with a reported velocity since it can be configured to have any suitable value. To control the linear slide, there is a numeric LabVIEW input labeled as "Velocity" without any units given. The challenge associated with this was not knowing how the LabVIEW velocity value corresponded to the desired measured velocity of the linear slide. To determine this relationship, the EV3 Ultrasonic Sensor was used to determine the velocity for LabVIEW inputs ranging from 1,000 to 10,000 with increments of 1,000. The 3D-printed ultrasonic sensor wall was placed on the linear slide to reflect the ultrasonic signals back to the sensor for distance measurements. For each LabVIEW velocity input, a plot of position vs. time was produced in Excel with the manufacturer's reported distance uncertainty of ± 10 mm (LEGO Education, 2020). The plot for the LabVIEW velocity input of 5,000 is shown in Figure 31 as an example, and the error bars show uncertainty to 2σ (± 20 mm). In this plot, the slope of the line is the measured velocity in units of mm s^{-1}. The LINEST function in Excel was used to find the uncertainty in the slope of the line to give the uncertainty in the measured velocity.

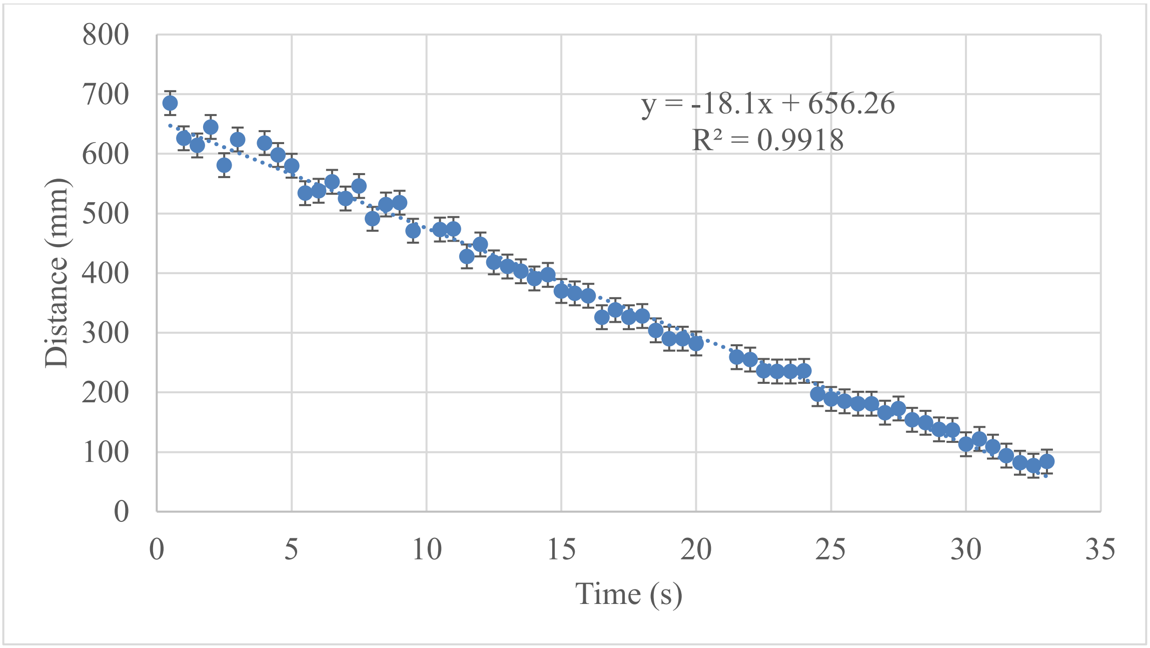


Figure 31. Measured velocity of the linear slide corresponding to the LabVIEW velocity input of 5,000.

A plot of the measured velocity vs. LabVIEW velocity input was produced with uncertainty to 2σ. A table of this plot's data is shown in Table 4, and the plot is shown in Figure 32. The R^2 value of 0.9997 supports a linear relationship of measured velocity vs. LabVIEW velocity. With this relationship known, the LabVIEW velocity can be linearly interpolated from the data for a desired measured velocity. For example, a desired measured velocity of 15 mm s^{-1} corresponds to a LabVIEW input of 5,219 using linear interpolation with the data in Table 4. Furthermore, this data was needed to provide an expected velocity to the remote student in the ultrasonic sensor experiment. This allows the student to compare their measured velocity to the expected velocity of the object.

Table 4. Measured velocity of the linear slide as it corresponds to each LabVIEW velocity input. The negative value denotes that the object was moving towards the sensor.

LabVIEW Input Velocity	Measured Velocity (mm s^{-1})	2σ Measured Velocity (mm s^{-1})
1000	-3.61	0.10
2000	-7.16	0.13
3000	-10.57	0.25
4000	-14.22	0.30
5000	-18.10	0.43
6000	-21.61	0.63
7000	-25.51	0.76
8000	-29.15	1.02
9000	-32.97	1.18
10000	-36.91	1.36

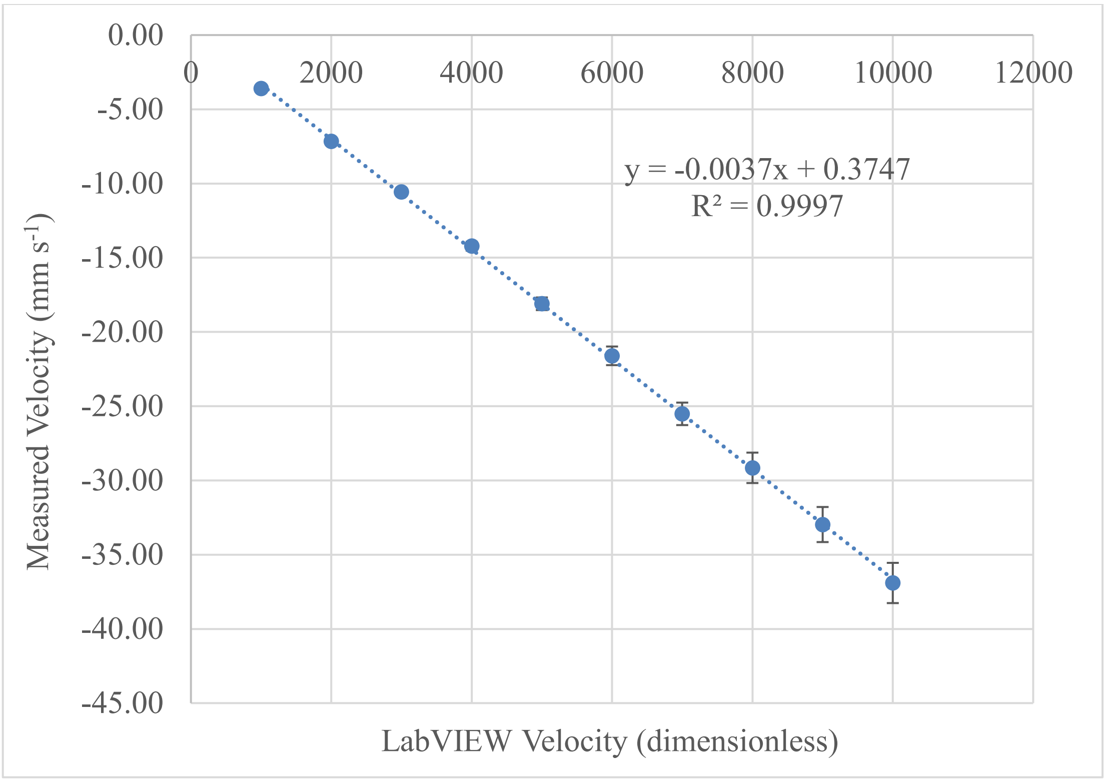


Figure 32. Relationship of measured velocity vs. LabVIEW velocity.

5. PHYSICAL SECURITY EXPERIMENTS

Physical security experiments were developed to promote a nuclear security component for the remote laboratory. They explore material reflectivity, object surveillance, and remote object measurement using the LME3 Education Core Set and accompanying light, ultrasonic, and infrared sensors. Students will complete these experiments to gain an understanding of how sensors can be used in a facility's physical protection system.

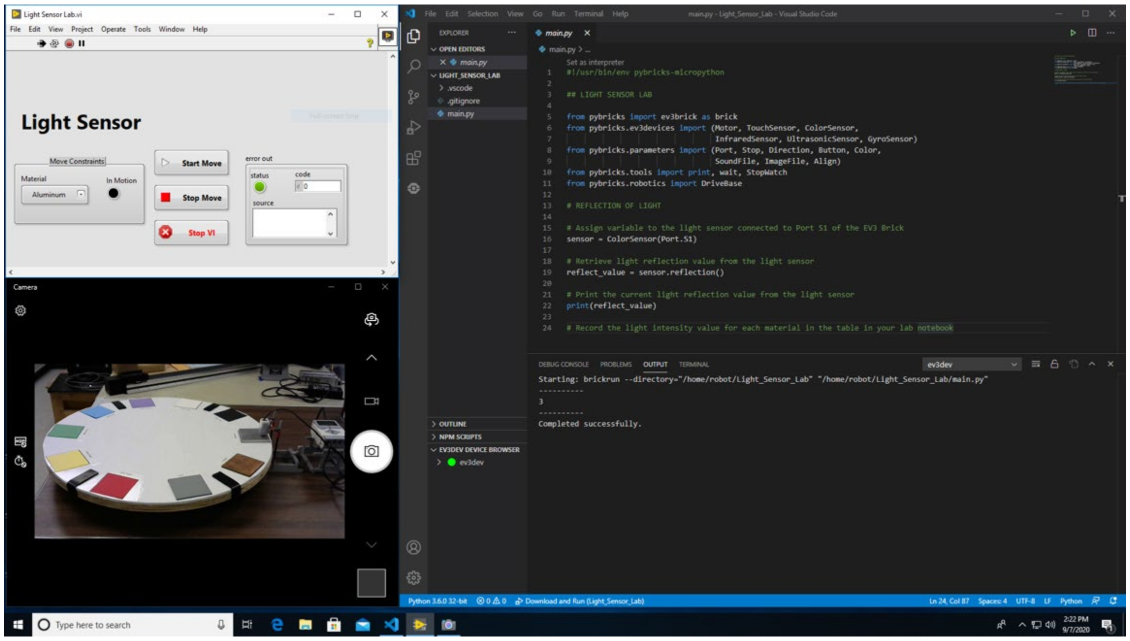
5.1. Preparing for the Experiments

Before beginning any of the physical security experiments, the remote student must set up EV3 MicroPython in Visual Studio Code on their home computer, create a project, and transfer it to the host computer. After launching Visual Studio Code, the LME3 MicroPython extension by LEGO Education must be installed from the Extensions tab. Once it is installed, a project must be created by navigating to the LME3 MicroPython extension on the left-hand side of the Visual Studio Code application. There, the student will appropriately name their project and save it in an easily accessible location. The "main.py" file will appear under the project folder in the explorer tab, and this is where the student will write their Python program using the instructional video for a given experiment. Once completed, the student will enter the remote laboratory with a prepared MicroPython program used to control and receive data from the LEGO sensors. The MicroPython program should be transferred by email or Google Drive from the student's home computer to the host computer before the scheduled laboratory time.

5.2. Connecting to the Experiments

Setting up a remote desktop connection with the host computer comes first and foremost. For cyber-security issues it is suggested a virtual private network (VPN) be used to connect to an institution's network. Next, the student will use a remote desktop client on their computer. For Windows computers, it is suggested to use the Windows Remote Desktop application that is included in the computer's pre-installed programs (Microsoft, 2020). For Mac users, the Microsoft Remote Desktop application is suggested and can be downloaded from the App Store (Apple, 2020). When establishing the remote desktop connection, the student will need the IP address of the host computer. The student will need remote login permission from the institution's information technology (IT) department.

Using the remote procedures in each experiment, the student is first asked to open the camera application to view the apparatus, download and open the appropriate LabVIEW program specific to controlling the rotary table or linear slide, and position these two windows so that both can be seen simultaneously. Figure 33 shows an expected view a student has on their computer screen. Next, the student is asked to launch Visual Studio Code and open the folder that contains the program they prepared with the instructional video. This folder should have been transferred from the student's home computer to the host computer by way of email or Google Drive. The student then connects to the EV3 Intelligent Brick using the "EV3DEV DEVICE BROWSER" drop-down menu. After selecting “Click here to connect to a device,” a searchable window will appear at the top of the screen. The student will select “ev3dev” to connect to the Brick, and a green circular light should appear in the “EV3DEV DEVICE BROWSER” drop-down menu. This drop-down menu is where the student can view all of the contents of the Brick. Lastly, the MicroPython code is run by pressing F5 or by selecting “Run ➔ Start Debugging” in the toolbar (LEGO Education, 2019).



```
#!/usr/bin/env pybricks-micropython

## LIGHT SENSOR LAB

from pybricks import ev3brick as brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import (Port, Stop, Direction, Button, Color,
                                 SoundFile, ImageFile, Align)
from pybricks.tools import print, wait, StopWatch
from pybricks.robotics import DriveBase

# REFLECTION OF LIGHT

# Assign variable to the light sensor connected to Port S1 of the EV3 Brick
sensor = ColorSensor(Port.S1)

# Retrieve light reflection value from the light sensor
reflect_value = sensor.reflection()

# Print the current light reflection value from the light sensor
print(reflect_value)

# Record the light intensity value for each material in the table in your lab notebook
```

```
Starting: brickrun --directory="/home/robot/Light_Sensor_Lab" "/home/robot/Light_Sensor_Lab/main.py"
----------
3
----------
Completed successfully.
```

Figure 33. A student’s view of the light sensor experiment on the host computer.

5.3. Light Sensors for Material Reflectivity

5.3.1. Introduction

Different colors reflect light in varying ways, with lighter colors reflecting greater amounts of light and darker colors reflecting lesser amounts of light (Scientific American, 2020). This also occurs with different material types; emissive and polished materials will reflect more light than dull and unpolished materials. Light from the sun contains wavelengths corresponding to all visible light colors to produce an overall “white” light. Darker objects do not have any of these visible light colors, resulting in the absorption of the light instead of reflection. Light that is not reflected from a surface is absorbed as heat in the material.

The light sensor experiment was developed to introduce students to scenarios that make use of the color-dependent reflectivity of light. From a security standpoint, light sensors can be used to process images in surveillance applications based on varying intensities of reflected light. Different materials and colors absorb different amounts of light, and this light is retained in the material as heat. By understanding which materials and colors will be the most difficult to reimage or analyze, proper decisions can be made to supplement any possible inadequacies in a surveillance system with other sensors.

The physical apparatus of the light sensor experiment, shown in Figure 34, includes the EV3 Intelligent Brick, EV3 Color Sensor, rotary table, foamboard layout diagram for the rotary table, a set of assorted-color 3”x3” squares of construction paper, and 3”x3” squares of aluminum, copper, and lead (Neenah, 2020). The layout diagram has 10 squares with black outlines to show a teaching assistant where to place each material. The center of each square is placed equidistantly to ensure a consistent angle separation of 36 degrees. The color sensor is fixed to hover over a single square portion of the rotary table. The EV3 Color Sensor is referred to as a light sensor for this experiment because it detects the intensity of the reflected ambient light and outputs a value of this intensity in units of Lux (LEGO Education, 2019).

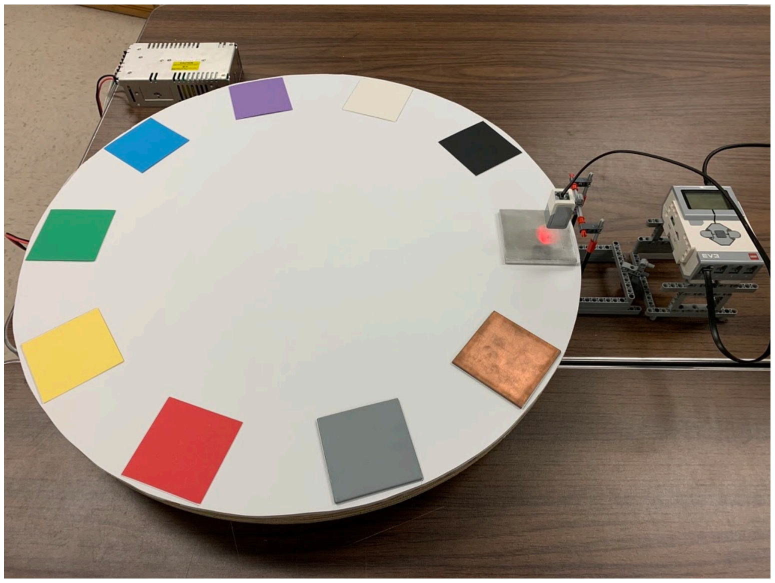


Figure 34. The light sensor experiment apparatus.

The interface to control the rotary table is a LabVIEW VI with a drop-down menu containing each material. When a material is selected in the drop-down menu, the corresponding absolute position is used by the motor to position it under the sensor. This interface is shown in Figure 35.

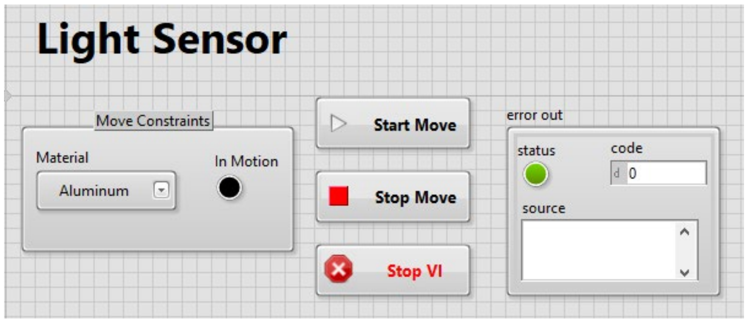
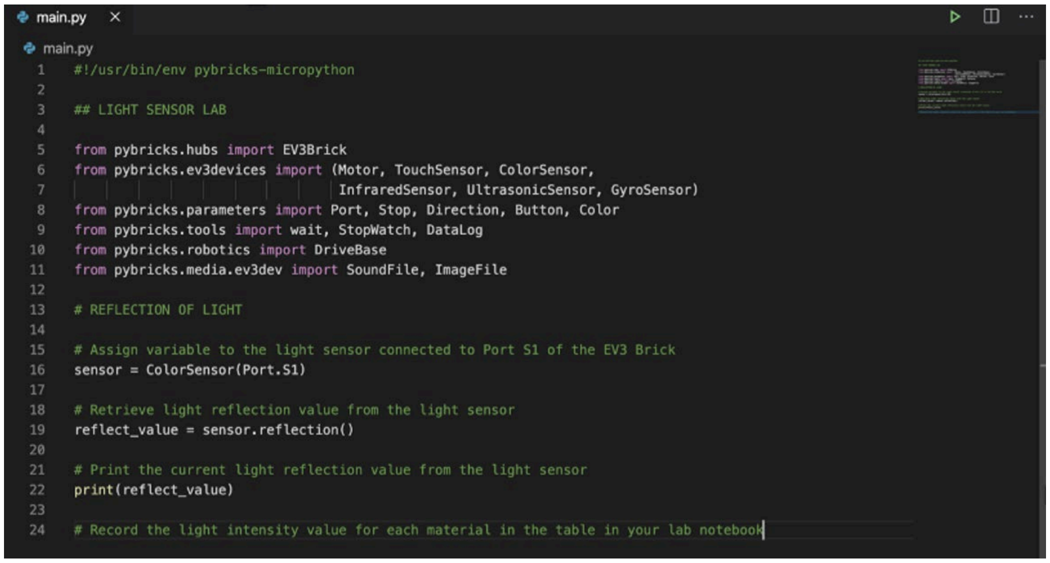


Figure 35. Light sensor experiment LabVIEW VI for controlling the rotary table.

5.3.2. Preliminary Assignment

As a preliminary assignment, the student is asked to prepare a data table containing a column with rows of each of the 10 materials, a column with rows for each corresponding light intensity value in units of Lux, and a column with rows for the corresponding calculated percent reflectivity of each material. The student will create this table either in a spreadsheet or in their laboratory notebook so that it can be populated with data during the experiment. In addition to making the table, the student is asked to make a preliminary guess ranking the materials from most reflective to least reflective. This will serve to expose any misconceptions about how materials reflect light when they are performing the experiment.

To retrieve data from the light sensor, students will need to run a MicroPython program in Visual Studio Code. Students will prepare the MicroPython program individually using a detailed instructional video as an aid. The lines of code are included for reference in Figure 36, and the instructional video's transcript and presentation is included in Appendix A for reference. This portion of the preliminary assignment will provide students with an understanding of general Python programming as well as how sensors operate off of a code. In the main.py file of the program, the first line sets up the pybricks-micropython environment that the code will run in. This environment is an optimized version of Python that LEGO MINDSTORMS uses for its EV3 functions. The next seven lines import functions from their respective libraries to be used in the code. These seven lines of imports cover all of the bases for the functions used to control the light sensor in this experiment.



```python
#!/usr/bin/env pybricks-micropython

## LIGHT SENSOR LAB

from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# REFLECTION OF LIGHT

# Assign variable to the light sensor connected to Port S1 of the EV3 Brick
sensor = ColorSensor(Port.S1)

# Retrieve light reflection value from the light sensor
reflect_value = sensor.reflection()

# Print the current light reflection value from the light sensor
print(reflect_value)

# Record the light intensity value for each material in the table in your lab notebook
```

Figure 36. The MicroPython code used to retrieve data from the light sensor.

A connection to the light sensor needs to be established in the MicroPython program. The light sensor is connected to the EV3 brick through Port S1. To implement this in the code, a variable is assigned to the light sensor to indicate the port being used.

This is done by creating a variable named “sensor” and setting it equal to LEGO’s light sensor function called “ColorSensor()”; note that Color and Light are used interchangeably when describing this type of sensor. Within the “ColorSensor()” function, the port is indicated by placing “Port.S1” into the parentheses of the function.

The reflected light intensity value from each material needs to be retrieved from the sensor and recorded by the remote student. To retrieve a light intensity value from the light sensor, a variable named “reflect_value” is created. To grab “reflect_value” from the light sensor, an action is performed on the previously defined variable, “sensor.” The “sensor” variable is the gateway to the light sensor. To retrieve this value from the light sensor, the “reflect_value” variable is set equal to “sensor.reflection(),” where the reflection function acts on the “sensor” variable to retrieve the light intensity value. The parentheses remain empty because no further input is needed by the function. Once the light intensity value has been retrieved from the sensor, the value is printed to the output of the code through “print(reflect_value)”.

5.3.3. Experiment Procedure

The experiment begins with a light intensity measurement of aluminum. The piece of aluminum is set in place underneath the light sensor so that its ambient light is shining onto the surface. After retrieving its light intensity value, the student will record the data and cycle the rotary table to the next material. This procedure is repeated until each material has a recorded light intensity value. The procedures for this remote experiment are included in Appendix A.

After the data is collected, the student calculates a reflectivity ratio of each material normalized to the light intensity value of aluminum. The percent reflectivity of each material is based on the amount of light reflected by aluminum because it reflects the most light out of all of the materials tested. The equation for calculating the percent reflectivity is shown in Eq. 1, and an example of collected data for the experiment is shown in Table 5.

$$\text{Reflectivity Ratio} = \frac{\text{Amount of Light from Sample}}{\text{Amount of Light from Aluminum}} \times 100 \qquad \text{(Eq. 1)}$$

Table 5. Collected data from a run through of the light sensor experiment.

Material	Light Intensity (Lux)	Reflectivity Ratio (%)
Aluminum	93	100
Copper	58	62
Lead	22	24
Red Paper	42	45
Yellow Paper	63	68
Green Paper	10	11
Blue Paper	13	14
Purple Paper	34	37
White Paper	76	82
Black Paper	3	3

Several discussion questions are provided to the student to expand on the concepts they learned:

- How might materials be difficult to reimage in a security application?
    - Darker colored materials would be difficult to reimage if there is a collection of shades of dark colors in one area. It would require a much more sensitive sensor to distinguish the varying low value light intensities from each other. Materials like aluminum and white paper, which reflect much more light, would produce a similar issue but with distinguishing high value light intensities from each other.
- How would the results change for metallic materials if they were more or less polished?
    - If metallic materials were more polished, more light would be reflected from the surface of the materials resulting in less heat retained in the material. Polished materials reflect more light because they have a more even, smooth surface. Therefore, the reflected light rays travel parallel to each other away from the surface.

When the student is preparing to leave the remote laboratory, they are asked to do several things out of courtesy for the next student. First, they will position the rotary table back to aluminum. Next, they will delete their MicroPython program from the Brick using the “EV3DEV DEVICE BROWSER” drop-down menu. Lastly, they will close out of all windows and sign out of the host computer. The teaching assistant will verify that the system is ready before the next remote student arrives to their scheduled laboratory time.

5.4. Ultrasonic Sensors for Velocity Determination

5.4.1. Introduction

In security and surveillance scenarios, ultrasonic sensors are advantageous because they track moving objects by sending and receiving ultrasonic sounds. The time between these sound waves is indicative of the distance between the sensor and the object being tracked (LEGO Education, 2019). Therefore, if an ultrasonic sensor sends and receives these sounds continuously, a plot of the object's position over time can be produced to ascertain its velocity.

The ultrasonic sensor experiment incorporates the 3D printed ultrasonic sensor wall, a linear slide, and the EV3 Ultrasonic Sensor. The wall's base was designed to seamlessly slide onto the linear slide's tracks, and the linear slide is used to move the wall on a one-dimensional track to and from the ultrasonic sensor. As the wall is moving, the sensor sends and receives ultrasonic sounds to and from the wall using the MicroPython program the student created for the experiment with the supplemental instructional video. The experiment apparatus is shown in Figure 37. The experiment is conducted in parts A, B, and C, each with a different velocity to determine.

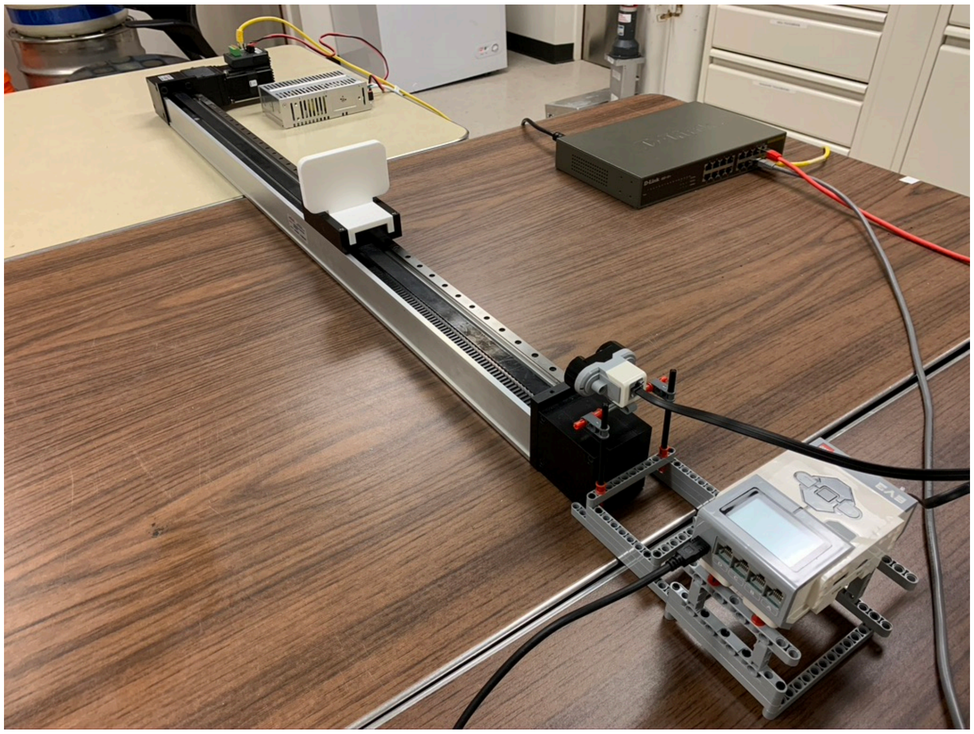


Figure 37. The ultrasonic sensor experiment apparatus.

5.4.2. Preliminary Assignment

Before starting the remote laboratory, the student should first prepare (3) tables in a spreadsheet. The tables will have “Time (s)” and “Distance (mm)” as columns with an indefinite number of rows. The student will also watch the ultrasonic sensor instructional video to prepare the experiment’s MicroPython program. Throughout the experiment, the program will output distance measurements as well as the time of each measurement. These values are entered into the appropriate spreadsheet for post-experiment analysis.

The student will develop a MicroPython program to collect data from the ultrasonic sensor. An instructional video will be provided to help them in the development. The presentation and video transcript are included in Appendix A. To begin, a “time_step” variable is defined, and represents how often to retrieve a distance value from the ultrasonic sensor in units of milliseconds. The time step value can be adjusted based on the needs of the experiment; however, for all parts of this experiment it is set equal to 500 milliseconds for consistency. Next, a “step” number is defined so that the program knows how many times to retrieve a distance value. This value can be changed based on the needs of the experiment.

The next portion of the MicroPython program involves outputting time and distance so that the student can enter these values into the spreadsheet for post-experiment analysis. First, labels are created for the program output for “Time (s)” and “Distance (mm)”. Next, for-loops are set up to perform a number of iterations equal to the number of steps defined by the user. A for-loop is an iterative structure that performs a process a specified number of times. In the for-loop for the time output, the time is calculated by multiplying the loop’s current iteration number by the time step. This value is then divided by 1000 to convert from milliseconds to seconds. The time value of that for-loop iteration is then printed to the output of the program. This is repeated for all iterations of the for-loop. To retrieve and output the distance value from the ultrasonic sensor, an additional for-loop is implemented in the program. To retrieve a distance measurement from the ultrasonic sensor, a variable named dist_value is created. To get this value, an action is performed on the defined variable, sensor. The sensor variable is the gateway to the ultrasonic sensor. To retrieve this value from the ultrasonic sensor, the dist_value variable is set equal to sensor.distance(), where the distance function acts on the sensor variable to retrieve the distance measurement. This distance measurement is then displayed in the output of the program by using the "print()" function, and the for-loop does not iterate again until an amount of time equal to the time step has passed.

5.4.3. Experiment Procedure

The experiment begins with the student ensuring the correct starting position of the ultrasonic sensor wall on the linear slide. Using the LabVIEW VI interface shown in Figure 38 for Part A of the experiment, the student selects "Position 1" in the object position drop-down menu. The student procedures are provided in Appendix A for reference. With the wall set in place, the student's next step is to determine the amount of time it takes for the wall to travel along the length of the linear slide towards the ultrasonic sensor; this value is used for Δtime in Eq. 2. The student then moves the object back to the starting position, sets the time_step variable equal to 500 milliseconds, and uses the ratio shown in Eq. 2 to calculate the number of steps needed to collect data with the MicroPython program. If the number of steps is not adequately determined, the student may not collect enough data points to calculate the wall's velocity within the uncertainties of the expected value. The student will know this after seeing the MicroPython program end before the wall reaches the end of the linear slide. The variable should then be updated in the MicroPython program after calculating the number of steps.

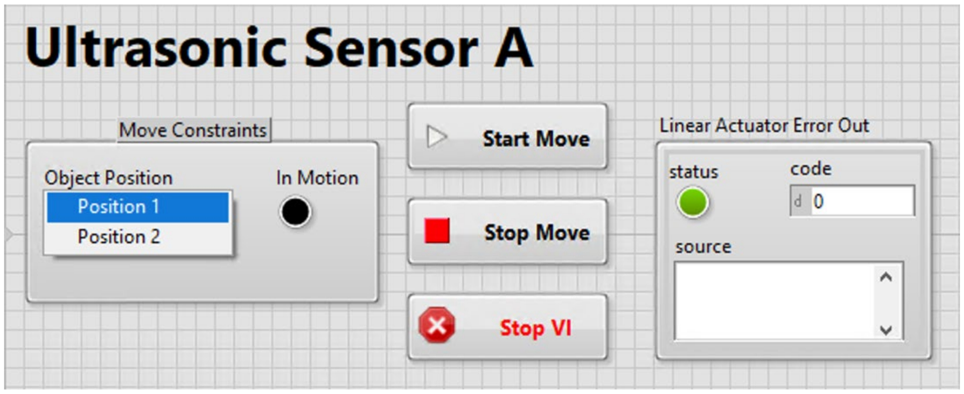


Figure 38. Front panel of the LabVIEW VI for Part A of the ultrasonic sensor experiment.

$$\text{Steps} = \frac{\Delta\text{time}}{\text{time\_step}} \qquad \text{(Eq. 2)}$$

Where:

Steps = The number of instances that a distance measurement will be retrieved from the ultrasonic sensor

Δtime = The amount of time it takes for the object to travel from Position 1 to Position 2 on the linear slide [ms]

time_step = The amount of time in between each step [ms]

The experiment begins with the student collecting position versus time data. The wall is moved from Position 1 to Position 2 on the linear slide using the LabVIEW VI while the MicroPython program collects data. If the program stops running before the object stops at Position 2, the student will need to adjust their Δtime value in Eq. 2 to allow for more movement time, thereby resulting in an increased number of steps. When the program stops running, columns of time and the corresponding distance measurements are provided to the student in the output. These columns are then recorded in the prepared spreadsheet.

Once the data is in the spreadsheet, the velocity of the wall is determined by plotting the distance measurements with respect to time. The student may have extra data points at the beginning and end of the columns for one or two reasons: 1) a delay in the MicroPython program beginning to collect distance measurements and the wall leaving Position 1 or 2) the wall reached Position 2 before the MicroPython program stopped collecting distance measurements. The student will remove these data points to improve the velocity estimation and its uncertainty. With the knowledge that velocity is equal to distance as a function of time, the student is able to ascertain the wall's velocity from the slope of a generated trend line in a plot. Excel's LINEST function is used to provide the uncertainty in the slope of the linear trendline.

These steps are repeated for Part B and Part C using their respective LabVIEW VIs. The same MicroPython program is used to collect data for all three parts of the experiment; however, the student must remember to update the number of steps in each part using Eq. 2. The magnitude of the velocity increases across Parts A-C, thereby decreasing the value of Δtime and the number of steps. The value of time_step remains constant at 500 milliseconds in each of the three parts of the experiment. The results of a completed ultrasonic sensor experiment are shown in Figure 39 – Figure 41 and in Table 6. These results were compared to the expected results in Table 4.

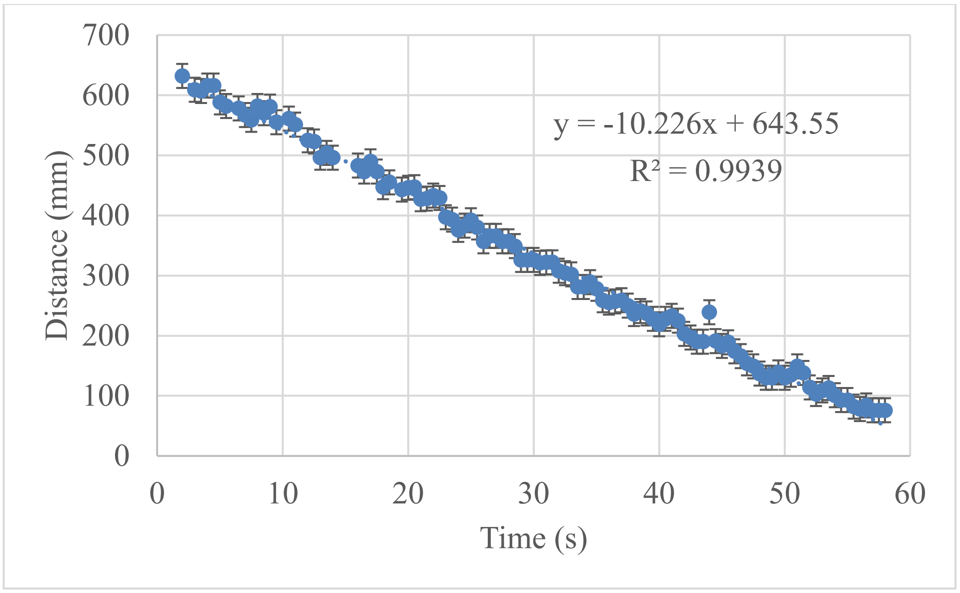


Figure 39. Results from Part A of the ultrasonic sensor experiment. Vertical error bars are shown to 2σ to account for the distance measurement uncertainty.

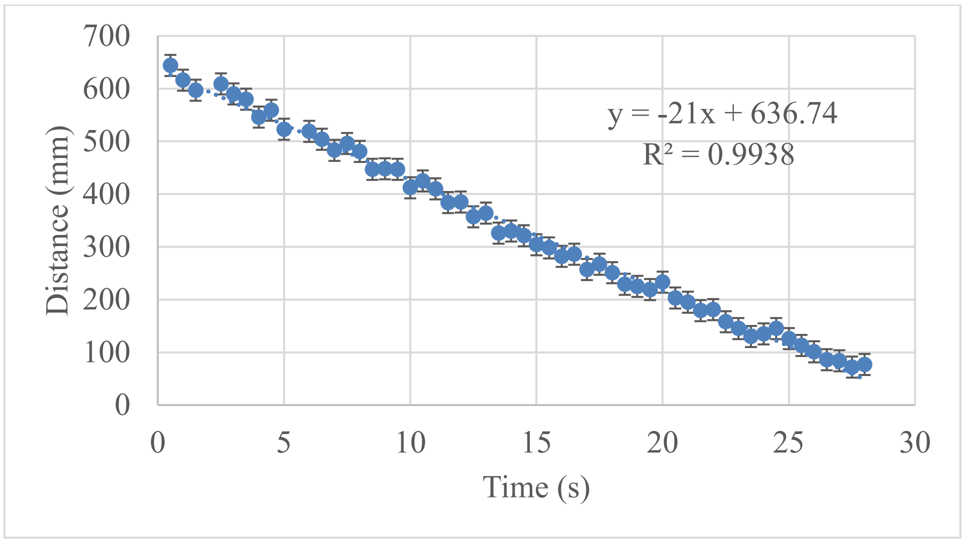


Figure 40. Results from Part B of the ultrasonic sensor experiment. Vertical error bars are shown to 2σ to account for the distance measurement uncertainty.

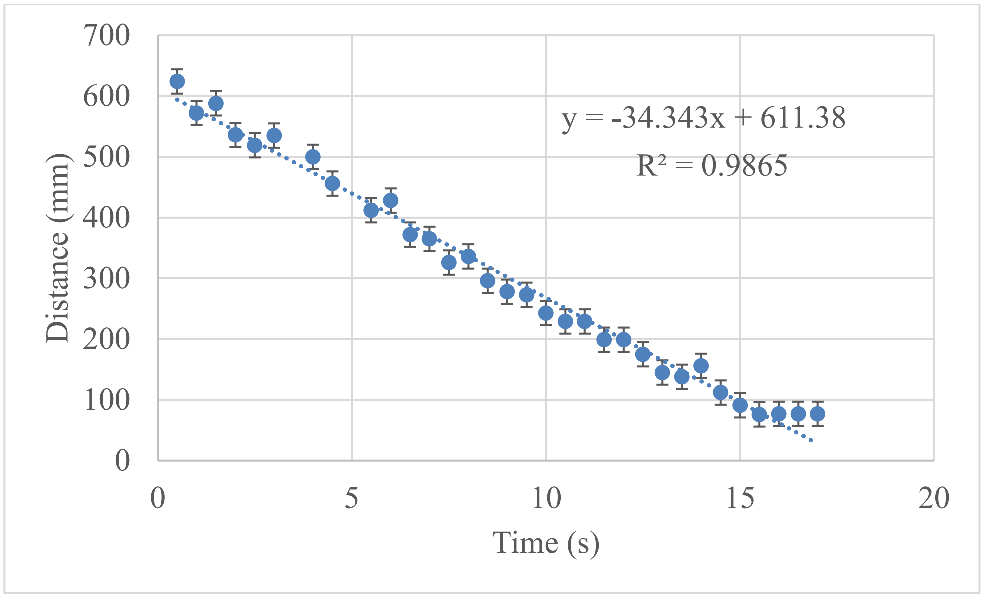


Figure 41. Results from Part C of the ultrasonic sensor experiment. Vertical error bars are shown to 2σ to account for the distance measurement uncertainty.

Table 6. Collected data from a run through of the ultrasonic sensor experiment. Uncertainties are shown to 2σ (LEGO Education, 2020).

Experiment Part	Calculated Velocity (mm s^{-1})	Expected Velocity (mm s^{-1})
A	-10.2 ± 0.16	-10.6 ± 0.25
B	-21.0 ± 0.46	-21.6 ± 0.63
C	-34.3 ± 1.46	-33.0 ± 1.36

Several discussion questions are provided to the student at the end of the experiment as a post-laboratory assignment:

- What does the slope of the line in each plot represent? What does the R^2 value imply?

  - The slope of the line is the velocity of the wall as it travelled towards the ultrasonic sensor, and the R^2 value shows the student that the data closely aligns with the linear relationship of distance over time.
- Are the calculated velocities statistically equivalent to the expected velocities given by the TA?
  - As seen in Table 6, the velocity values from each part of the experiment were able to be reproduced with statistically equivalent values. The calculated values are within 2σ of the expected values in Table 4. After the laboratory, the TA will provide the velocity of the object in each part of the experiment to the student. The uncertainty of the ultrasonic sensor was reported to be ± 1 cm (LEGO Education, 2020).
- What happened to the velocity between Part A and Part C? Do the results support the observations?
  - The velocity noticeably increases in each successive part of the experiment. The student should have noticed that the velocity increased in a couple of ways. First, the Δtime and steps values used in the MicroPython program should have been observed to have decreased across Part A through Part C. Second, the student should have noticed how much more quickly the wall arrived at the end of the linear slide by visually observing the wall with the live-time camera feed.
- How could an ultrasonic sensor be programmed to alert a facility when there is movement in a restricted area?

    - This sensor can be used for this capability by implementing conditional statements in the MicroPython program that check whether or not an object is within a pre-defined boundary. For instance, if the sensor detects that an object is 100 cm away with a pre-defined boundary of 150 cm, the program can alert the user that there has been a security breach and that there may be an unwanted presence in a secure area.

5.5. Infrared Sensors for Remote Object Measurements

5.5.1. Introduction

A question at the end of the ultrasonic sensor experiment asks students how an ultrasonic sensor could be used in a security or surveillance application. This question serves as the basis of the culmination of the series of physical security experiments. If the velocity of an object is determined, how can its magnitude be used to acquire other quantifiable data? In the infrared sensor experiment, the student will use a known velocity to measure the length of a moving object and its individual constituents.

Infrared sensors function similarly to that of ultrasonic devices in security and surveillance scenarios. Infrared sensors operate by sending an infrared signal towards an object and measuring the reflection of its signal (LEGO Education, 2019). Whereas ultrasonic sensors can more generally determine that motion is occurring in a large area, infrared sensors must be closer to the object of interest. A scenario presented to the student is an unattended monitoring system supplemented with an infrared sensor placed at a key measurement point (KMP) between two material balance areas (MBAs). This could help automatically verify material inventory by tracking movement from one

MBA to another. If itemized forms of nuclear material, such as storage tanks or containers, were transported between the two MBAs on a conveyor belt of a known speed, the length and number of the objects being transported could be determined based on the total amount of time that the infrared sensor detects each object. Even without a conveyor belt with a known speed, an ultrasonic sensor could be used to determine the speed of the objects as they are moved between MBAs before being used for object measurement.

The infrared sensor experiment aligns well with the scenario of the KMP between two item MBAs. In this experiment, an irregularly-shaped, comb-like object was designed to seamlessly attach to the linear slide. The comb-like object passes by the infrared sensor at a known velocity that was previously determined by the student in Part A of the ultrasonic sensor experiment. The infrared sensor is placed 2 mm away from the comb wall. This apparatus is shown in Figure 42. This experiment operates using a single infrared sensor to give a one-dimensional measurement of the object based on the proximity readings recorded by the infrared sensor. If possible, a grid of infrared sensors would give a more-encompassing, two-dimensional measurement of objects passing through the KMP to estimate the length and height of the passing objects; however, this experiment only utilizes a single infrared sensor for simplicity.

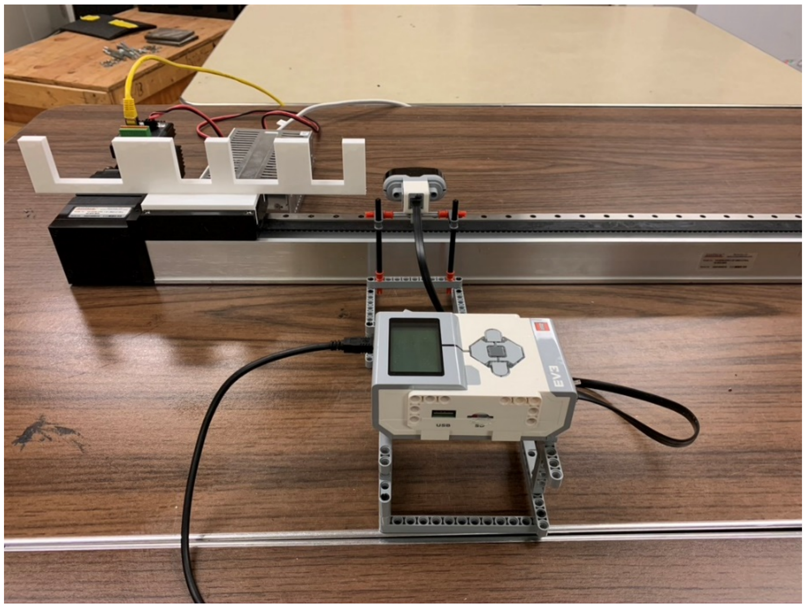


Figure 42. The infrared sensor experiment apparatus.

By completing this experiment, the student will be able to recognize the importance of material reflectivity as they learned in the light sensor experiment. Since infrared sensors operate by detecting reflected light, the type and color of material of the transported items will affect the performance of the infrared sensor. Thus, the comb-like object used in this experiment was designed to be white to reflect light back to the infrared sensor more easily. Another trial of the experiment can be performed by the student using a black comb-like object. This will show the operational difference in how the infrared sensor responds to a change in color. Since the black object will absorb more of the infrared light, less of this light will return to the sensor resulting in skewed measurements.

5.5.2. Preliminary Assignment

Before entering the remote laboratory, the student will create the table shown in Table 7. The student will also watch the infrared sensor instructional video to prepare the experiment's MicroPython program. Throughout the experiment, the program will output values of distance as a measure of proximity as well as the time each measurement was recorded. These values will be recorded in a spreadsheet for post-experiment analysis. Similar to the ultrasonic sensor experiment, the infrared sensor experiment is conducted in parts A, B, and C; there is a different time step used in each part, as seen in the three columns of Table 7. More on the variation of the time step will be discussed shortly.

Table 7. Table provided to the students for the infrared sensor experiment.

ROI	Actual Measurement (mm)	Estimated Measurement w/ 500 ms Time Step (mm)	Estimated Measurement w/ 200 ms Time Step (mm)	Estimated Measurement w/ 100 ms Time Step (mm)
1	20.19 ± 0.04			
2	30.19 ± 0.04			
3	20.25 ± 0.04			
4	50.10 ± 0.04			
5	20.04 ± 0.04			

The student will develop a MicroPython program to collect data from the infrared sensor with the instructional video as an aid; the presentation and video transcript are included in Appendix A as a reference. To begin, a “time_step” variable is defined, and it defines how often to retrieve a distance value from the infrared sensor in units of milliseconds. In the ultrasonic sensor experiment, the time step was held constant throughout each of the three parts. For this experiment, however, the time step is the variable changed in each part in order to show the student how results are affected by how often measurements are taken. Next, the number of steps is defined so that the program knows how many times to retrieve a distance value, and this value can also be changed based on the needs of the experiment. Similar to the ultrasonic sensor MicroPython program, the time and distance measurements are collected using for-loops before being displayed in the program output.

5.5.3. Experiment Procedure

The experiment begins with the student ensuring the correct starting position of the comb-like object on the linear slide; from here on, the comb-like object will be referred to as the comb wall. Using the LabVIEW VI interface shown in Figure 43, the student selects “Position 1” in the object position drop-down menu for the linear slide to position the comb wall at its starting point on the left-hand side of the infrared sensor. The student procedures are provided in Appendix A for reference.

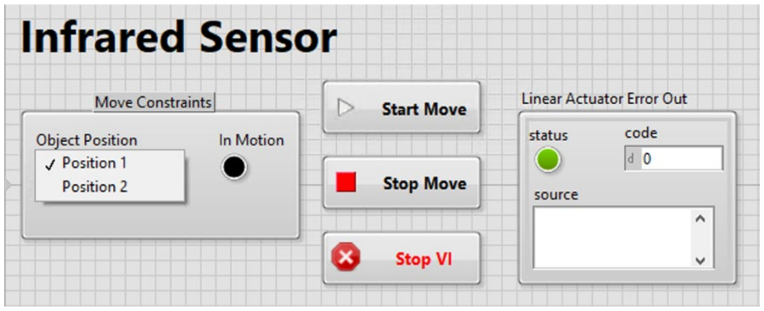


Figure 43. Front panel of the LabVIEW VI for the infrared sensor experiment.

Next, the student determines the amount of time for the wall to travel the length of the linear slide and completely pass the infrared sensor; this value is Δtime in Eq. 2. The student then moves the object back to the starting position, sets the time_step variable equal to 500 milliseconds for Part A, and uses the ratio shown in Eq. 2 to calculate the number of steps needed to collect data with the MicroPython program. If the number of steps is not adequately determined, the student may not collect enough data points to measure the entire length of the comb wall. The student will know this after seeing the MicroPython program end before the comb wall fully passed in front of the infrared sensor. The variable should then be updated in the MicroPython program after calculating the number of steps.

The experiment begins with the student collecting position versus time data. The wall is moved from Position 1 to Position 2 on the slide using the LabVIEW VI, and the MicroPython program is simultaneously running in Visual Studio Code to collect data points. If the program stops running before the object comes to a stop at Position 2, the student will need to adjust their Δtime value in Eq. 2. This will allow for more movement time and result in an increased number of steps. When the program stops running, columns of time and the corresponding distance measurements are provided to the student in the output. These columns are recorded in a spreadsheet that was prepared before beginning the experiment. This is repeated for Part B and Part C with time step values of 200 ms and 100 ms, respectively.

Once the data is in the spreadsheet, the length of the comb wall and its constituents is determined based on the Boolean values of each distance measurement. Boolean values are used for expressions that have a yes-or-no answer, where an outcome of "true" is typically assigned a value of 1 and an outcome of "false" is typically assigned a value of 0. If a constituent of the comb wall is covering the infrared light emitted by the sensor at a certain time then the distance measurement for that time will be 0, the Boolean value will be "true", and a value of 1 will be assigned. Alternatively, if there is a gap between the comb wall's teeth the distance measurement will be greater than 0, the Boolean value will be "false", and a value of 0 will be assigned. This threshold can be adjusted based on the application, such as if the infrared sensor needs to be positioned further away from the moving object. In the spreadsheet, a new column will be created for the values of 0 and 1. In each cell, an IF function will be used to identify whether or not a constituent of the comb wall is present or absent for certain periods of time. Once this column is created and values of 0 and 1 are assigned to each distance measurement, a scatter plot with straight lines of assigned values with respect to time will be used to show the overall shape of the moving object; this is shown in Figure 44 – Figure 46. There are visual shifts along the x-axis in these plots because of when the MicroPython program was initiated with respect to when the comb wall began its travel; however, this has no effect on the results. Between Figure 44 – Figure 46, there is also a noticeable change in conciseness of the shape of each constituent. As the time step is decreased, the infrared sensor retrieves distance measurements much more frequently. Therefore, the shape of each constituent becomes less trapezoidal and more rectangular since the infrared sensor detects the presence of the start and end of each constituent sooner.

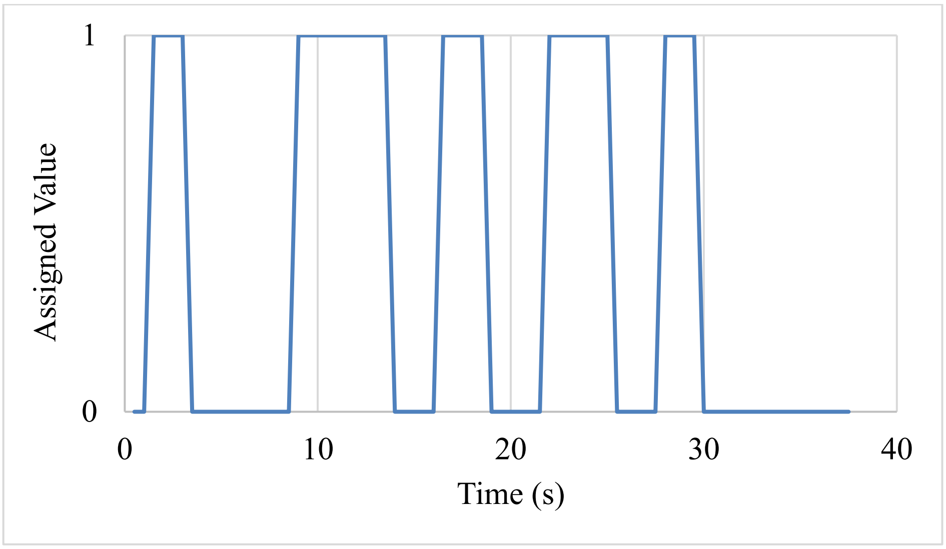


Figure 44. Example of scatter plot with straight lines showing the overall shape of the comb wall using a time step of 500 ms.

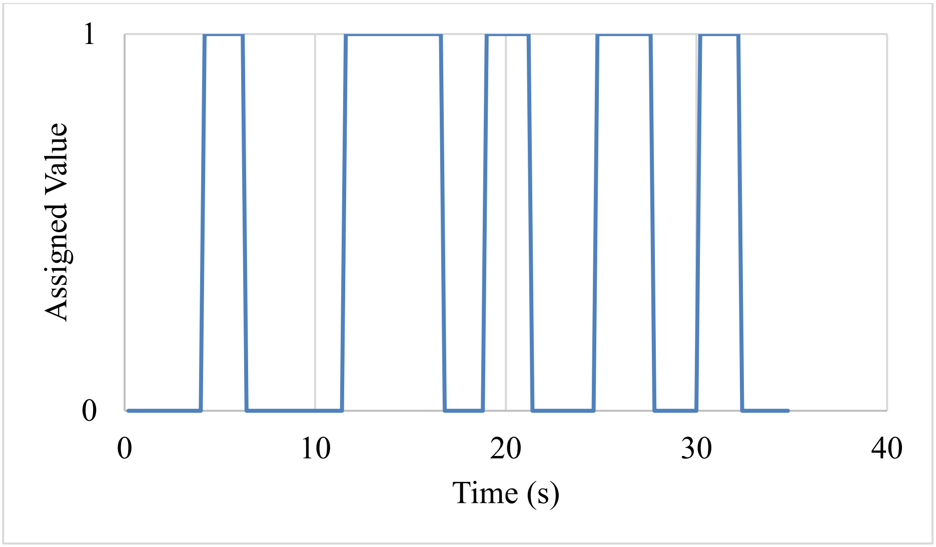


Figure 45. Example of scatter plot with straight lines showing the overall shape of the comb wall using a time step of 200 ms.

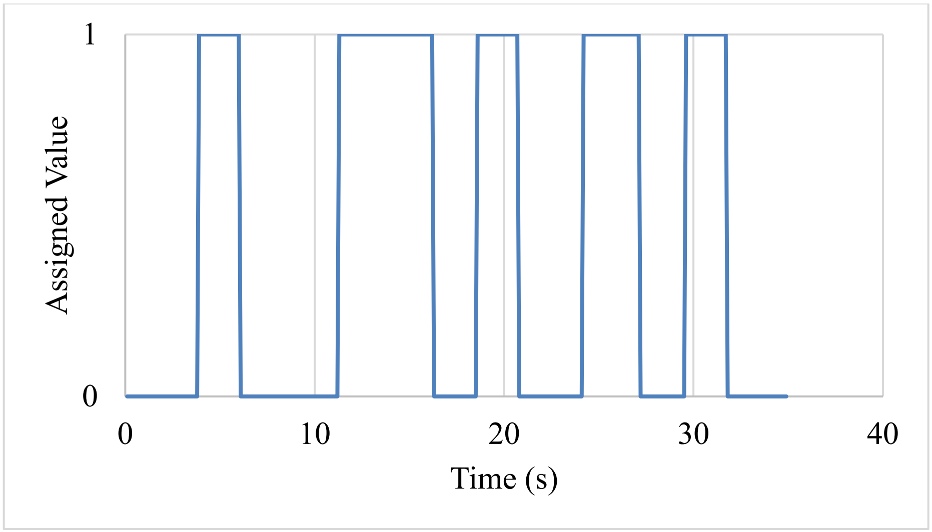


Figure 46. Example of scatter plot with straight lines showing the overall shape of the comb wall using a time step of 100 ms.

Next, the student is directed to determine the length of each constituent. This is accomplished by multiplying the amount of time that each tooth was detected by the velocity of the linear slide as determined in Part A of the ultrasonic sensor experiment. This results in an estimate of the length of the constituent, and it is then repeated for the remaining (4) ROIs. The student is also advised to propagate the error from the velocity found in the ultrasonic sensor experiment (LEGO Education, 2020). The students will compare their calculations to the actual width of each ROI. They will be instructed to discuss any discrepancies between their results and the expected measurements as well as how they would improve their measurements in the future.

Table 8. Collected data from a run through of the infrared sensor experiment. Uncertainties are shown to 2σ (LEGO Education, 2020) (General, 2020).

ROI	Actual Measurement (mm)	Estimated Measurement for Part A: 500 ms Time Step (mm)	Estimated Measurement for Part B: 200 ms Time Step (mm)	Estimated Measurement for Part C: 100 ms Time Step (mm)
1	20.19 ± 0.04	14.49 ± 0.53	19.32 ± 0.70	20.29 ± 0.74
2	30.19 ± 0.04	24.15 ± 0.88	28.98 ± 1.05	28.98 ± 1.05
3	20.25 ± 0.04	19.32 ± 0.70	19.32 ± 0.70	20.29 ± 0.74
4	50.10 ± 0.04	43.47 ± 1.58	46.37 ± 1.68	47.33 ± 1.72
5	20.04 ± 0.04	19.32 ± 0.70	19.32 ± 0.70	19.32 ± 0.70

Several questions are asked to the student as a post-laboratory assignment. For reference, the results of a run through of the infrared sensor experiment are shown in Table 8:

- How are the results affected by the change in the time step value?
    - It can be seen that the majority of the constituents had an improvement in the estimation of their length as the time step was decreased. This is expected because the infrared sensor retrieved distance measurements much more frequently as the time step decreased. As a result, the calculated results became more accurate since the increased number of steps meant that the infrared sensor made measurements closer together.
- Is there a disadvantage associated with decreasing the time step?
    - There are going to be limitations with respect to how often any sensor can retrieve data. This is a result of a dead time that affects sensors similarly to that of radiation detectors. Much like the limit to how quickly a person can contract their muscle after a previous contraction, the dead time is the amount of time in which some sort of event cannot be completed or recorded. Eventually, the time step will reach this limit and the sensor will not be able to retrieve data at that frequency.
- How do the estimated measurements compare to the actual measurements?
    - Three of the ROIs eventually had estimated measurements that were statistically equivalent to the actual measurements with the time step of 100 ms in Part C. This can be improved by continuing to decrease the time step until the actual and estimated measurements are within 2σ for all ROIs.

As previously discussed, the same procedures can be used to complete an additional version of this experiment with a black comb wall. This structure is shown in Figure 47, has the same dimensions as the white comb wall, and is placed the same distance away from the infrared sensor. The purpose of this supplementary portion of the experiment is to demonstrate material reflectivity and how it affects data collection for physical protection systems.

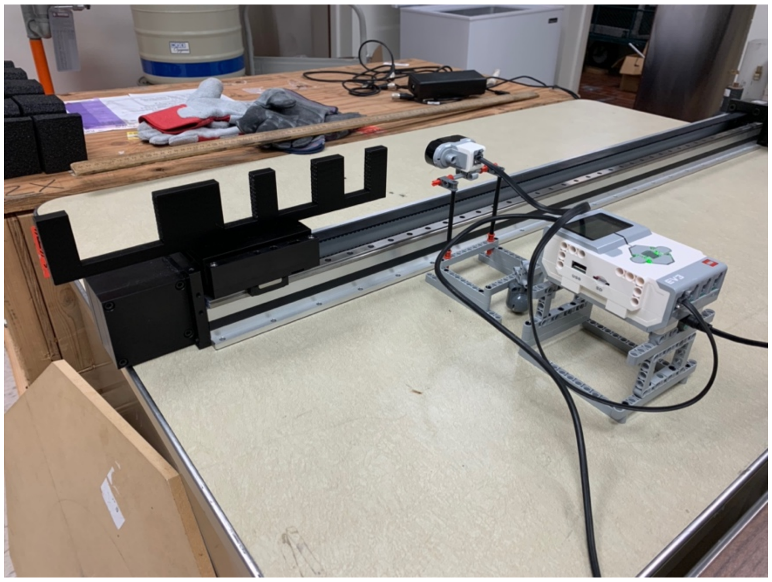


Figure 47. The black comb wall used for a supplementary portion of the infrared sensor experiment.

Table 9. A sample of raw data from the white and black comb wall versions of the infrared sensor experiment using a time step of 500 ms. This sample corresponds to ROI 1 of the comb wall.

Time (s)	Proximity w/ White Comb Wall (cm)	Proximity w/ Black Comb Wall (cm)
0.5	61	100
1	51	100
1.5	38	100
2	27	100
2.5	21	100
3	16	100
3.5	12	100
4	4	74
4.5	0	20
5	0	16
5.5	0	16
6	0	19
6.5	0	47

A sample of the raw data collected by the infrared sensor using both the white and black comb walls is shown in Table 9. A comparison of the two columns of data shows that the infrared sensor registered the black comb wall further away from the sensor than the white comb wall. The answer to this phenomenon is that the reflectivity of the black PLA material is substantially lower than that of the white PLA material. This causes more infrared light from the infrared sensor to be absorbed by the comb wall as heat. Since this light becomes "lost" from the system, it is unable to be reflected back to the sensor. The infrared sensor then confuses the "lost" infrared light as light that is being reflected off of an object that is farther away than it actually is, resulting in the measurements not approaching 0 mm. Despite the lack of physical contact between the sensor and the comb wall, the distance measurements reach 0 mm with the white comb wall because of the material’s high reflectivity and actual separation distance of 2 mm. Lastly, another observation to point out is that the proximity measurements decrease gradually with the white wall. This is because infrared signals are emitted in all directions, so as the wall approaches the sensor the emitted infrared signals are gradually reflected back to the sensor. This results in the gradual decrease in proximity measurements for that wall. The black wall, however, absorbs the emitted infrared signals as it approaches the sensor. This results in the black wall being able to sneak up on the sensor as it does not contribute to a gradual decrease in proximity measurements. The culmination of the physical security experiments results in the student experiencing firsthand how the light reflectivity of various materials will affect measurements when using light-emitting sensors. With this experience, students will be able to recognize scenarios in which the material type, velocity, and dimensions of an object will all need to be considered.

6. RADIATION DETECTION EXPERIMENTS

The proof-of-concept remote laboratory included the following experiments: gamma source identification, uranium enrichment quantification, and GM dead time determination. Improvements were made to the existing experiments, and several new radiation detection experiments were added to the remote laboratory. Procedures were written for the remote laboratories, including a version for students and a version for teaching assistants. The version for teaching assistants details how to set up the experiment apparatus and ensure that the equipment is functioning properly. The student procedures for the following remote experiments are included in Appendix B for reference.

6.1. GM Dead Time Determination and Counting Statistics

The dead time experiment was updated from Emery's proof-of-concept version to improve its functionality. Instead of the duct tape being used to secure sources and detectors, custom made platforms were constructed (Figure 48 and Figure 49). This improved functionality allows for consistent and improved source-detector geometry. Lastly, the Ludlum Model 2200 rate meter is now interfaced with a computer and students now can vary count time in addition to controlling start/stop times. The graphical user interface (GUI) for this program is shown in Figure 50 (Ludlum Measurements Inc., 2020).

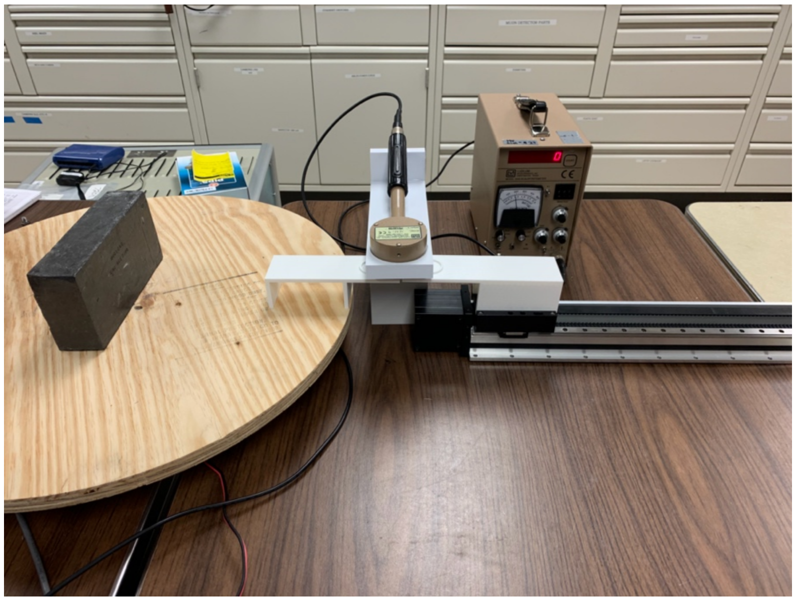


Figure 48. Apparatus of the dead time and counting statistics experiment.

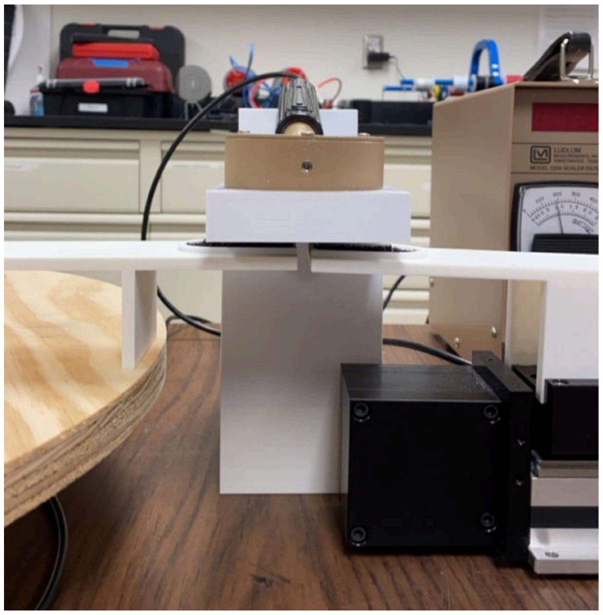
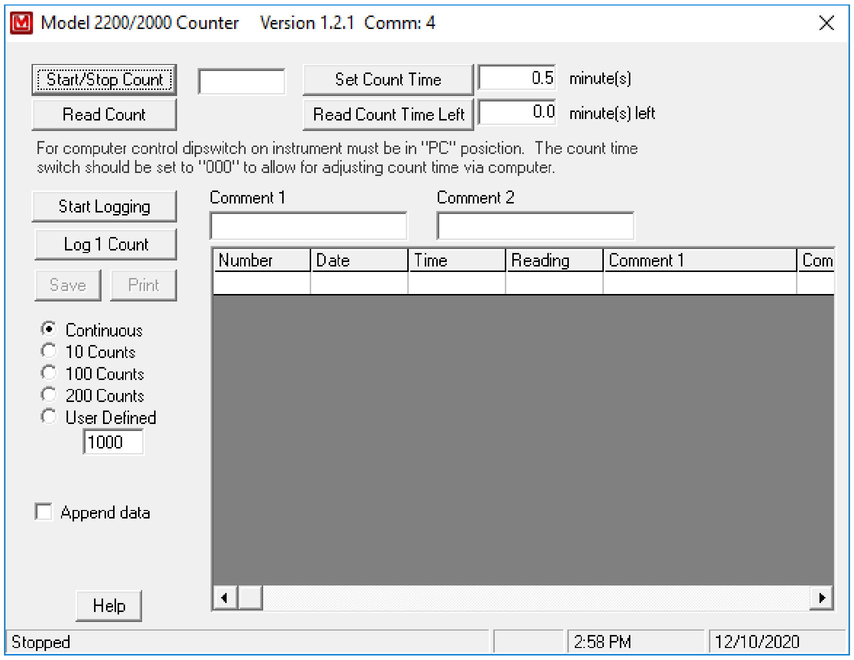


Figure 49. Close up view of the GM probe holder and source platforms.


Model 2200/2000 Counter Version 1.2.1 Comm: 4

Set Count Time 0.5 minute(s)

Read Count Time Left 0.0 minute(s) left

For computer control dipswitch on instrument must be in "PC" posiction. The count time switch should be set to "000" to allow for adjusting count time via computer.

Number	Date	Time	Reading	Comment 1	Com

[✓ Continuous] 10 Counts 100 Counts 200 Counts User Defined 1000

Stopped 2:58 PM 12/10/2020

Figure 50. The Ludlum Model 2200 rate meter computer program.

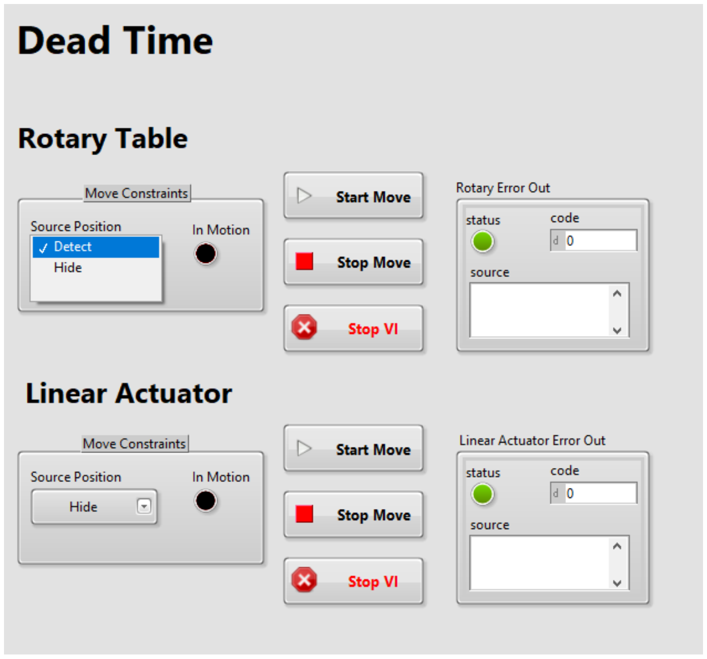


Figure 51. LabVIEW VI for the dead time and counting statistics experiment.

The purpose of the experiment is to introduce students to gas-filled detectors, dead time, and counting statistics. The dead time is the amount of time after a detection event in which a detector is unable to record a new detection event (Knoll, 2010). To find the dead time of the GM detector, the student will first separately measure two ^{204}Tl half sources. The student switches out the half sources from under the GM probe holder with the LabVIEW VI that controls the motors of the rotary table and linear slide, as shown in Figure 51. Next, the student will adjust the rotary table and linear slide so that the two half sources are placed under the GM probe holder for measurement as a whole source. Lastly, the sources will both be moved away from the detector for a measurement of the room's background radiation. In the counting statistics portion of the experiment, the student measures the two half sources as a whole source 40 times for 6 seconds each time. Once obtaining these measurements, the student will calculate the sum of the residuals around the experimental mean to determine the variance and ultimately the standard deviation σ. The student should find that the data follows a Gaussian distribution with 68% of the data points deviating from the experimental mean by 1σ.

6.2. Inverse Square Law

This experiment introduces students to the inverse square law, scintillation detectors, and simple gamma spectroscopy. The inverse square law is a fundamental radiation physics concept in which the intensity of radiation is reduced by a factor of one over the distance squared (Knoll, 2010). For example, if the distance between a detector and a source is doubled then the intensity of the radiation at the location of the detector will be reduced by a factor four. The detector in this experiment is a 2"x 2" NaI detector with a 30 mm aperture lead collimator, giving the student an introduction to scintillation detectors. The source is a 10 μCi ^{137}Cs check source so that it can be placed on the linear slide source platform. The apparatus of the experiment is shown in Figure 52 for reference. To implement this experiment without a student physically measuring the distance between the detector and the source, the linear slide was programmed to move in steps of 10 cm with the LabVIEW VI shown in Figure 53. At each step, the student uses GENIE-2000 Gamma Acquisition & Analysis to record the peak area from a user defined ROI highlighting the 662 keV peak from ^{137}Cs. A plot of the peak area with respect to distance shows that the data does not truly reduce by a factor of one over the distance squared. The relationship only applies to a geometry consisting of a point detector and a point source. Therefore, as the detector and source become more collimated the relationship will more closely apply to the geometry.

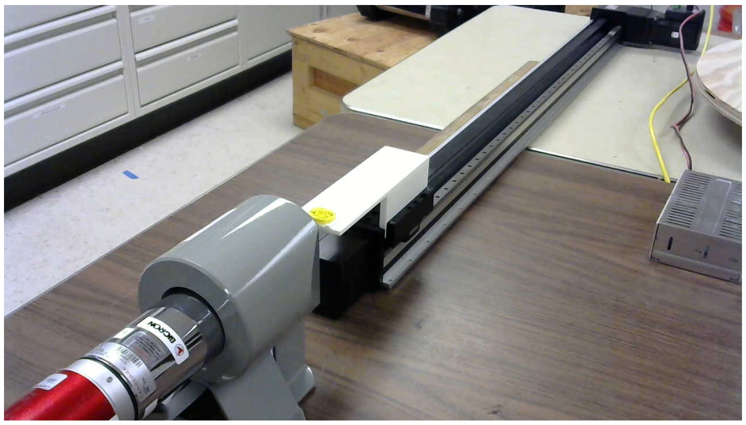


Figure 52. Apparatus of the inverse square law experiment.

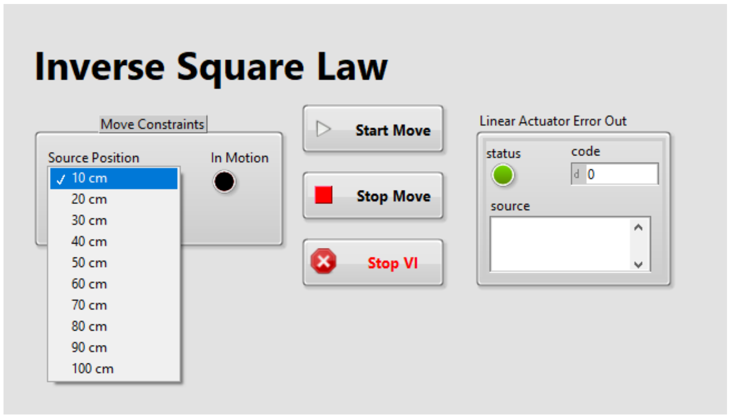


Figure 53. LabVIEW VI for the inverse square law experiment.

6.3. Alpha Spectroscopy and Attenuation

An alpha spectrometer is used to introduce students to alpha spectroscopy and attenuation. The Alpha Analyst, shown in Figure 54, has six chambers held under vacuum, and a passivated implanted planar silicon (PIPS) detector is used to detect and determine the energy of the emitted alpha particles in each chamber. It is necessary to note that the rotary table and linear slide are not used, and the camera is not required; however, the camera is pointed at the Alpha Analyst so that the student has a visual of the chambers and associated hardware.

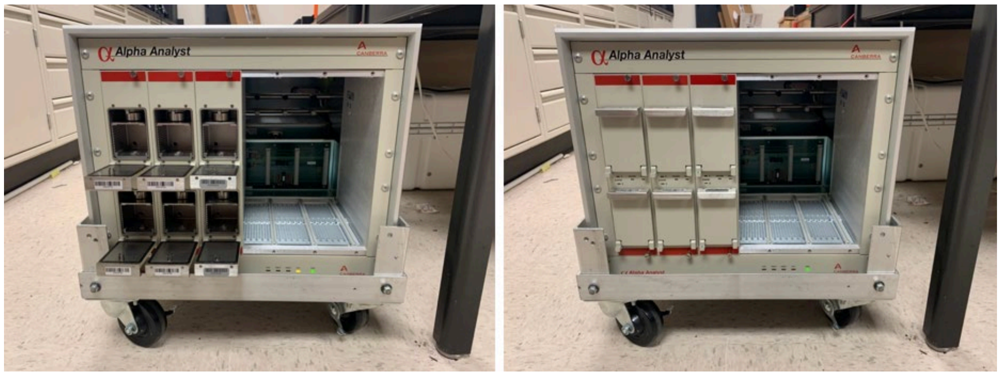


Figure 54. Apparatus of the alpha spectroscopy and attenuation experiment with chamber doors open (left) and closed (right).

The stopping power of Mylar film on alpha particles of varying energies is determined using the Mirion Technologies Alpha Analyst (Mirion Technologies, 2017). The student begins the experiment by performing an energy calibration with a mixed alpha source in the first chamber. Next, the student will record the count of mixed alpha sources in the next three chambers with a count time of at least 300 s. The mixed alpha sources will have the same activity and be made up of the same constituents as the source in the first chamber. In each successive chamber, the mixed alpha source will have an increased thickness of Mylar between it and the PIPS detector. The student will observe a noticeable shift downwards in each constituent's alpha particle's peak centroid energy as the thickness of the Mylar film is increased in each successive chamber. The slope of the plot's linear trendline of the peak centroid energy with respect to Mylar thickness will provide the stopping power of Mylar. This relationship is shown in Eq. 3.

$$S = -\frac{dE}{dx} \qquad \text{(Eq. 3)}$$

Where:

S = Stopping power of the Mylar film on alpha particles (MeV μm^{-1})

dE = Change in alpha energy (MeV)

dx = Change in position of the alpha particle (μm)

6.4. Compton Scattering

The Compton scattering experiment introduces students to the relationship between incident photon energy and the scattering angle of the photon with a free electron; this relationship is shown in Eq. 4 (Knoll, 2010). In the remote experiment, the scattering angle is adjusted in increments of 10° with the rotary table by rotating a 0.4 mCi ^{137}Cs source around an aluminum scattering rod that is fixed to the center of the rotary table. The detector responsible for recording the energies of the scattered photons is a 2"x 2" NaI detector that is positioned on top of a platform that crosses over the rotary table. Both the source and detector are collimated by lead with a 30 mm aperture. For reference, the apparatus is shown in Figure 55 and the LabVIEW VI is shown in Figure 56. The recorded scattered photon energy is observed using the ROI feature in GENIE-2000 Gamma Acquisition & Analysis (Mirion Technologies, 2016). At each angle, an ROI is created around the peak of interest to record its centroid and the FWHM of the peak. The energy of the scattered photon decreases as the scattering angle increases. The uncertainty in the energy of the scattered photon is found using Eq. 5.

$$E_{\gamma'} = \frac{E_\gamma}{1 + \frac{E_\gamma}{m_e c^2}(1 - \cos(\theta))} \qquad \text{(Eq. 4)}$$

Where:

$E_{\gamma'}$ = Energy of the scattered photon (MeV)

E_γ = Incident energy of the photon (MeV)

$m_e c^2$ = Rest mass of an electron (0.511 MeV)

θ = Scattering angle of the incident photon with a free electron (°)

$$\sigma_E = \frac{\text{FWHM}}{2.33} \qquad \text{(Eq. 5)}$$

Where:

σ_E = Uncertainty in the peak's centroid energy (MeV)

FWHM = Full width at half of the maximum of the peak (MeV)

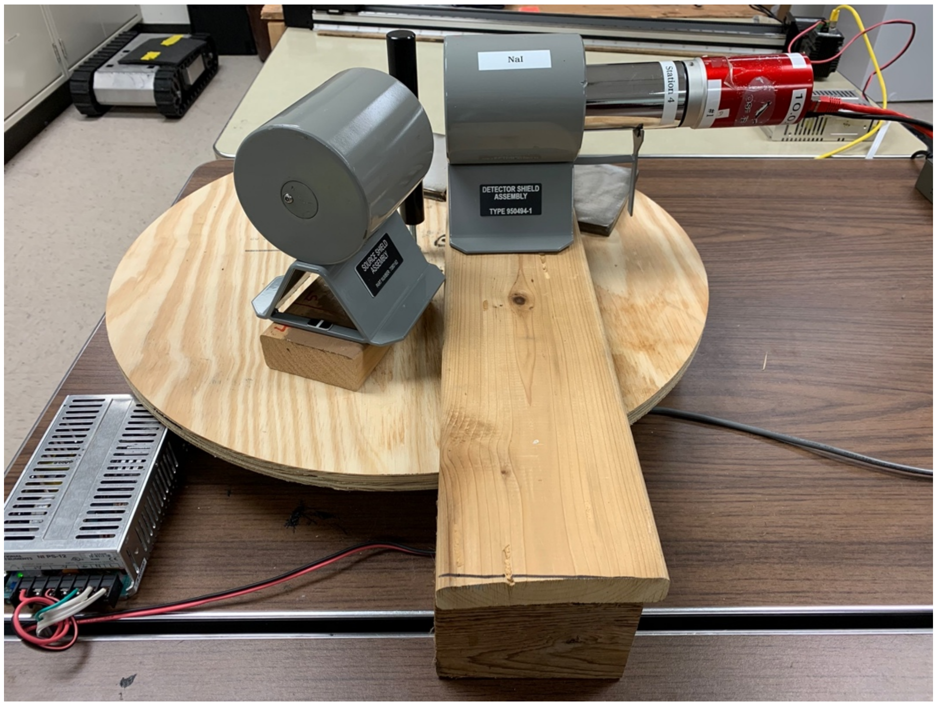


Figure 55. Apparatus of the Compton scattering experiment.

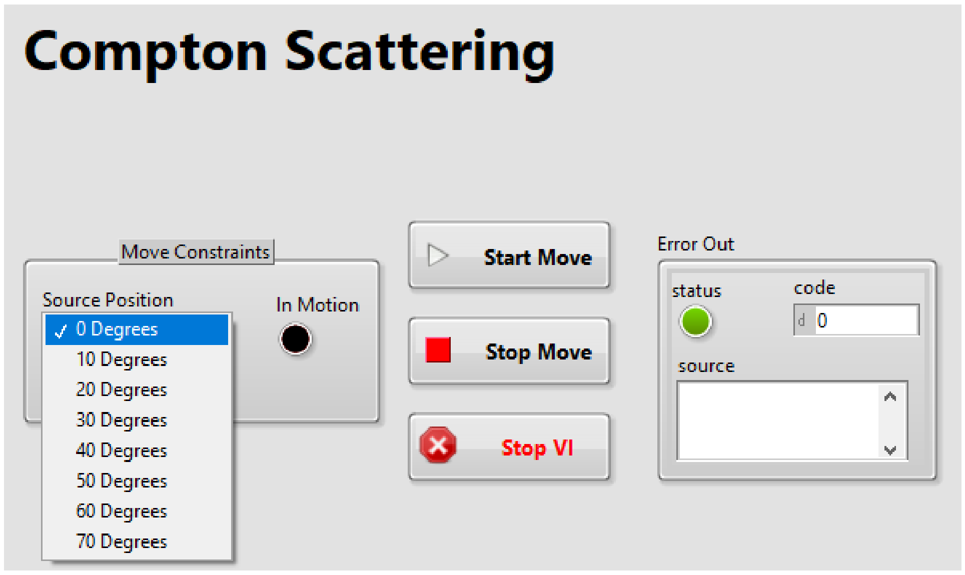


Figure 56. LabVIEW VI for the Compton scattering experiment.

6.5. Gamma Spectroscopy with Scintillation Detectors

The Gamma Spectroscopy with Scintillation Detectors experiment is an updated version of Emery’s proof-of-concept gamma source identification experiment. Updates to the apparatus include the circular rotary table to avoid the obstruction of corners with the detectors, the inclusion of both NaI and LaBr detectors for resolution comparison, custom source platforms for holding check sources, and increased lead shielding partitions to isolate each source of interest for detection and identification. The apparatus is shown in Figure 57 and the LabVIEW VI is shown in Figure 58.

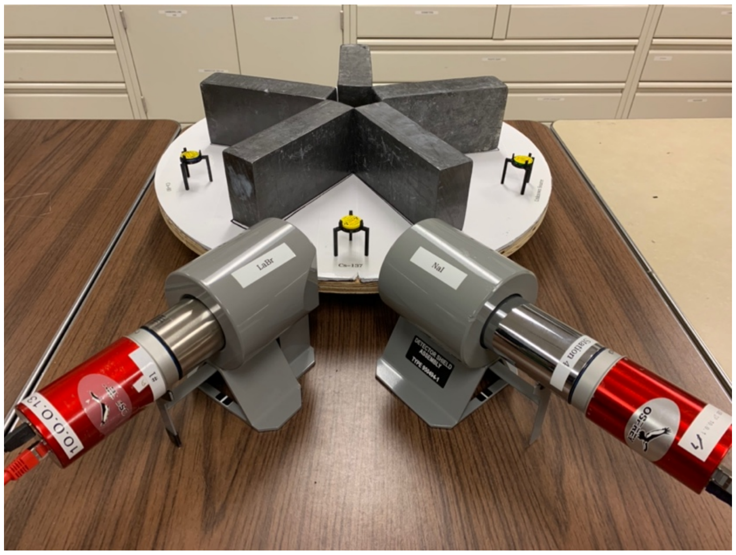


Figure 57. Apparatus of the spectroscopy with scintillation detectors experiment.

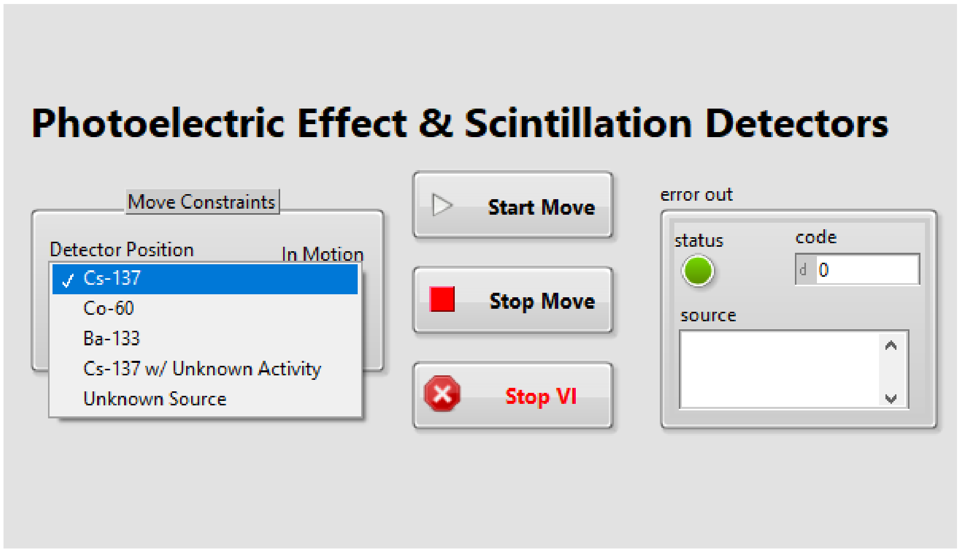


Figure 58. LabVIEW VI for the scintillation detectors experiment.

The apparatus consists of the rotary table partitioned into five sections by lead bricks. The first four sections contain a single 1 μCi check source on a source platform. The fifth partitioned section contains multiple unknown check sources on the source platform for source identification; the unknown sources are chosen by the teaching assistant or professor. Each in their own lead collimator with a 30 mm aperture, the NaI and LaBr detectors are positioned so that they can isolate the source of interest with the detector face at the same height as the source platform.

The objectives of the experiment are to observe spectroscopic features, identify unknown sources and determine their activity, and compare the detection differences between NaI and LaBr detectors. The student will learn how to use GENIE by energy and efficiency calibrating the NaI detector using the 1-μCi check sources in the first four sections of the rotary table. In this process, the student will also develop source libraries and observe spectroscopic features such as the photopeak produced by the photoelectric effect. Once the detector is energy and efficiency calibrated, the student will identify the unknown sources in the fifth section of the rotary table and determine their activities. The student will achieve this by learning how to customize GENIE analysis sequence and develop source libraries. The experiment is repeated twice so that students can observe detection differences between the two scintillators.

6.6. Attenuation Coefficients

Photon attenuation is an important phenomenon to consider in nuclear security because all materials will reduce the intensity of gamma radiation by some factor. Unlike alpha attenuation, photon attenuation results in a reduction in the peak area rather than a shift downwards in peak centroid energy. The relationship that governs this phenomenon is shown in Eq. 6 (Knoll, 2010).

$$I = I_o e^{-\mu t} \quad \text{(Eq. 6)}$$

Where:

I = Intensity of attenuated photons

I_o = Intensity of photons without an attenuator

μ = Linear attenuation coefficient (cm^{-1} or mm^{-1})

t = Thickness of material (cm or mm)

As shown in Figure 59, the experiment involves the use of the rotary table with a 10 μCi ^{137}Cs check source on a platform in the center, and the source is surrounded by varying thicknesses of lead, aluminum, polyethylene, and copper. These attenuators are held by custom attenuator holders with screws that keep the attenuators upright. To detect the attenuated photons, a lead collimated NaI detector with a 30 mm aperture is positioned on the side of the rotary table facing the ^{137}Cs check source. The student controls the rotary table with the experiment's LabVIEW VI to cycle the attenuators in between the ^{137}Cs source and the NaI detector, as shown in Figure 60. The attenuation coefficient of each material can then be determined by plotting the relationship of attenuated intensity with respect to material thickness, where the slope of the line is the linear attenuation coefficient in units of inverse material thickness.

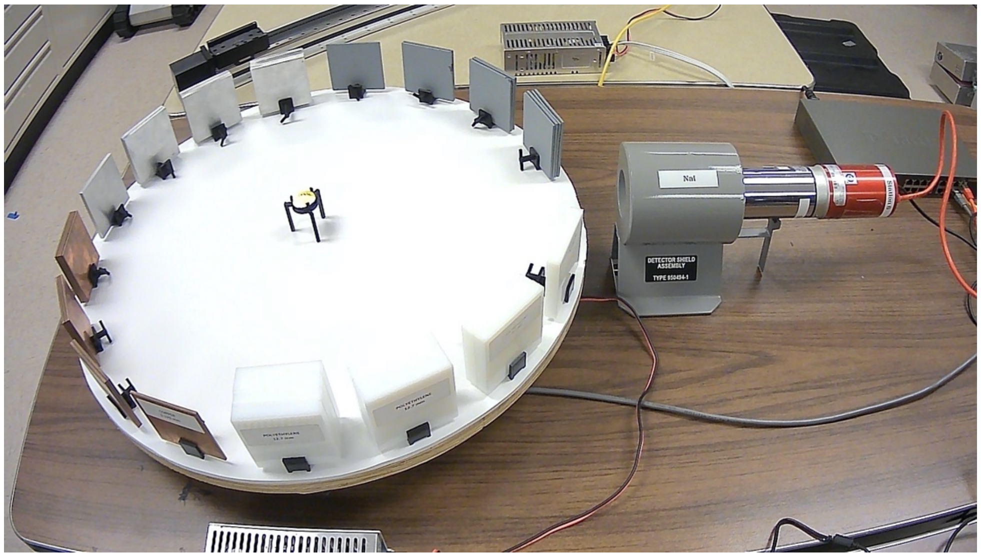


Figure 59. Apparatus of the attenuation coefficients experiment.

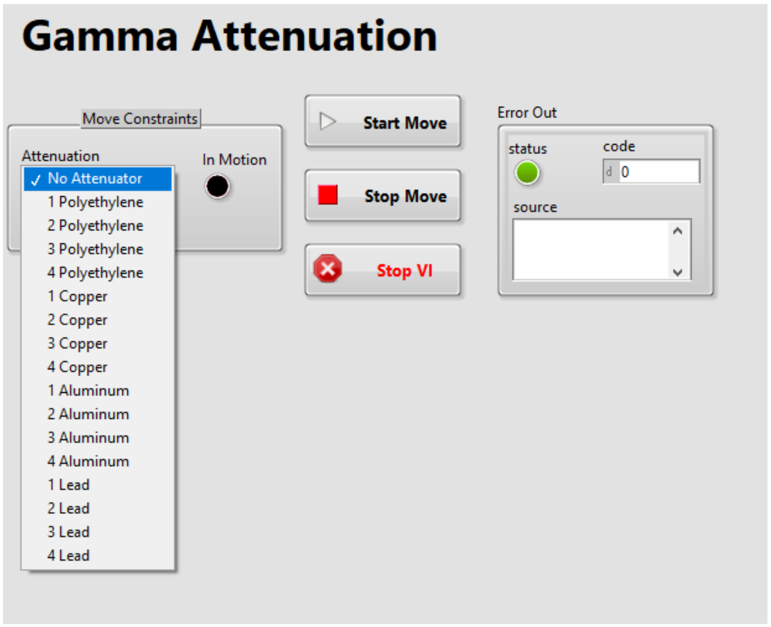


Figure 60. LabVIEW VI for the attenuation coefficients experiment.

6.7. HPGe Gamma Spectroscopy and Uranium Enrichment Determination

Students are further exposed to resolution differences between detectors and gamma spectroscopy by using the Mirion Technologies Falcon 5000 for energy and efficiency calibration (Mirion Technologies, 2017). As shown in Figure 61, the rotary table is set up with 1 μCi ^{137}Cs, ^{60}Co, and ^{133}Ba check sources positioned on custom source platforms to be vertically aligned with the center of the detector face. The student uses these sources one at a time with the LabVIEW VI shown in Figure 62 to calibrate the detector by energy and efficiency. At the time this experiment is completed in the semester, the student should be familiar with the calibration process. The main takeaway, therefore, is to notice the resolution of the HPGe detector and record gamma spectroscopy features such as the photopeak, Compton edge, and backscatter peak.

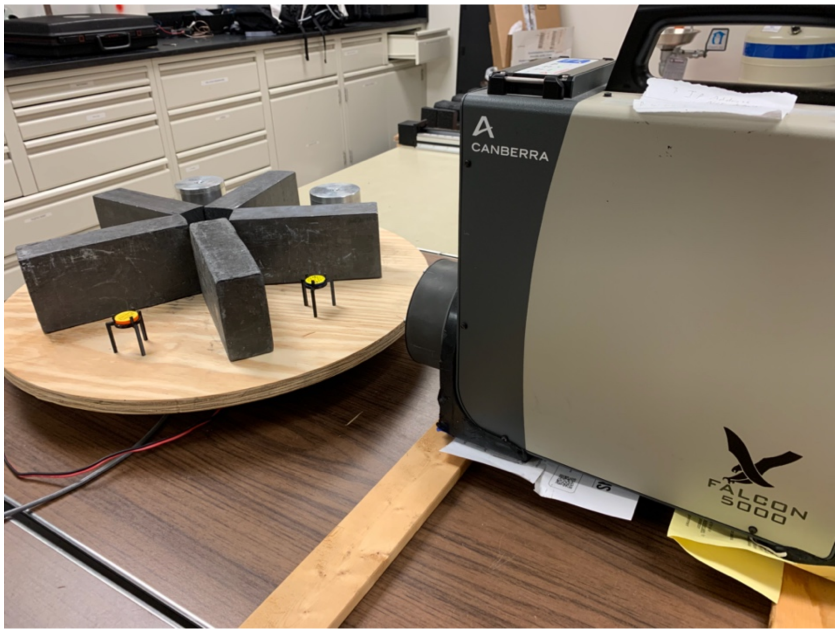


Figure 61. Apparatus of the uranium enrichment determination experiment.

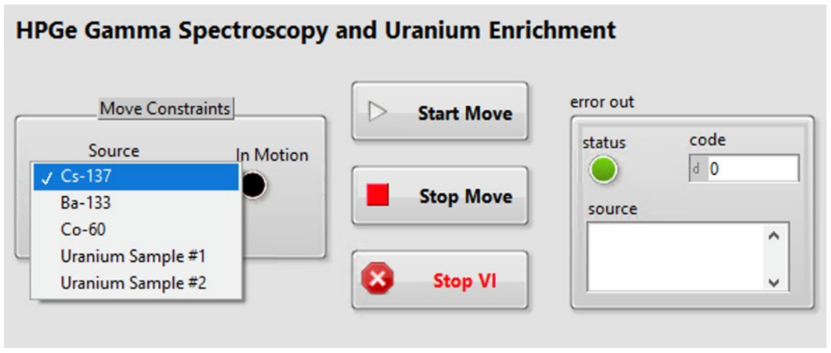


Figure 62. LabVIEW VI for the uranium enrichment determination experiment.

The latter portion of the experiment uses the comparator method to determine the enrichment of an unknown sample of uranium. The comparator method, shown in Eq. 7, uses the 186 keV and 1,001 keV peaks characteristic of ^{235}U and ^{238}U, respectively, to estimate the enrichment of an unknown sample of uranium (Marianno, 2019). The samples are shown in Figure 61 to be partitioned by lead bricks, and they cycle on the rotary table after the energy and efficiency calibration.

$$E_{unk} = E_k \times \frac{\frac{C_{unk\text{-}235}}{C_{unk\text{-}238}}}{\frac{C_{k\text{-}235}}{C_{k\text{-}238}}} \qquad \text{(Eq. 7)}$$

Where:

E_{unk} = Enrichment of the unknown source (%)

E_k = Enrichment of the known source (%)

$C_{unk\text{-}235}$ = Area of the 186 keV peak in the unknown source

$C_{unk\text{-}238}$ = Area of the 1,001 keV peak in the unknown source

$C_{k\text{-}235}$ = Area of the 186 keV peak in the known source

$C_{k\text{-}238}$ = Area of the 1,001 keV peak in the known source

7. RECOMMENDATIONS FOR FUTURE WORK

With the addition of physical security experiments and the expansion upon radiation detection experiments, the remote laboratory has become a functioning interdisciplinary educational tool that could be used by many more departments and institutions; however, there are still some improvements to be made. First, the student experience would benefit from having videos embedded in the LabVIEW VIs. These videos would provide the student with guided instruction to more closely resemble the traditional laboratory format. After certain items are selected from the drop-down menu of a LabVIEW VI, a video would begin to play to explain certain physics phenomena, remind the student of important details to notice, and make the remote experience more engaging. Next, the addition of neutron experiments would provide students experience with neutron moderation and using the various types of neutron detectors. The inclusion of these experiments is a challenge, however, because neutron sources are difficult to shield to avoid unwanted interference. These experiments would require a completely separate room with only one source of interest, and it would have to be away from all other stored neutron sources. The scheduling of students would be another challenge with neutron experiments. An experiment involving different geometries of neutron moderating materials would need its logistics carefully planned out since only one source and geometry could be set up at a time to avoid detection interference.

8. CONCLUSION

A remote laboratory for radiation detection and physical security education was developed to provide nuclear security education and training to students and professionals. Beginning with the proof-of-concept remote laboratory for radiation detection, a series of physical security experiments was developed to provide a nuclear security component to the remote laboratory. The experiments explored important fundamental concepts for understanding how sensors can be utilized in a facility's physical protection system. Using custom-made components, a rotary table, and a linear slide, the concepts of light reflectivity, motion sensing, and remote object measurement were taught in a distance laboratory setting. After completing the experiments, the student will have gained an understanding of how material properties affect the measurements taken by light-emitting sensors as well as how motion sensing can complement the function of an infrared sensor. The remote laboratory also saw an expansion of its radiation detection curriculum to include alpha spectroscopy, Compton scattering, and gamma attenuation experiments. It is hoped that other institutions will be able to use this remote laboratory for the education of students and professionals in nuclear security.

REFERENCES

Center for Innovative Research in Cyberlearning. (2020). *Remote Labs*. Retrieved from CIRCL Center: https://circlcenter.org/remote-labs/

Chennakesava Sai, P., & Shivraj Narayan, Y. (2014, December). Fused Deposition Modeling - Insights. Bonfring.

Clemson University. (2020). *Programs*. Retrieved from Clemson University: http://www.clemson.edu/online/programs/index.html

Dassault Systèmes. (2019). Introducing SolidWorks. Waltham, Massachusetts.

Emery, G. (2018). Remotely Accessible Radiation Detection Laboratory for Distance Education. College Station, Texas: Texas A&M University.

Etcher. (2019, November 28). *Etcher Installation Guide*. Retrieved from Etcher: https://etcher.download/etcher-installation-guide/

FlashForge Corporation. (2017). FlashForge Guider II 3D Printer.

FlashForge Corporation. (2020). *FlashPrint*. Retrieved from FlashForge: https://www.flashforge.com/product-detail/40

General. (2020). *Digital Fractional Caliper with Extra-Large LCD Screen, 3 Mode Display, 6-Inches*. Retrieved from https://www.generaltools.com/6-in-steel-digital-caliper-1

George, D. (2014). *MicroPython*. Retrieved from MicroPython: https://micropython.org/

Grimm, T. (2004). *User's Guide to Rapid Prototyping.* Society of Manufacturing Engineers.

Hyder, A. C. (2010). Design and implementation of remotely controlled laboratory experiments. Atlanta, Georgia: Georgia Institute of Technology.

KAMU. (2020). *Radioactivity iLab*. Retrieved from KAMU PBS Learning Media: https://kamu.pbslearningmedia.org/resource/cyb11.pd.cyber.ilab/radioactivity-ilab/

Knoll, G. F. (2010). *Radiation Detection and Measurement* (4 ed.). Wiley.

Kopp, D. G. (2011). Development of an On-line Radiation Detection and Measurements Laboratory Course. Clemson, South Carolina: Clemson University.

LEGO Education. (2019, March 26). Getting Started with EV3 MicroPython.

LEGO Education. (2020). *LEGO MINDSTORMS Education EV3 Color Sensor*. Retrieved from LEGO Education: https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-color-sensor/45506

LEGO Education. (2020). *LEGO Mindstorms Education EV3 Core Set*. Retrieved from LEGO Education: https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-core-set/5003400#lego-mindstorms-education-ev3

LEGO Education. (2020). *LEGO MINDSTORMS Education EV3 Infrared Sensor*. Retrieved from LEGO Education: https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-infrared-sensor/45509

LEGO Education. (2020). *LEGO MINDSTORMS Education EV3 Intelligent Brick*. Retrieved from LEGO Education: https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-intelligent-brick/45500

LEGO Education. (2020). *LEGO MINDSTORMS Education EV3 Ultrasonic Sensor*. Retrieved from LEGO Education: https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-ultrasonic-sensor/45504

LinTech Motion Control Inc. (2020). *I40 Series Belt Driven Linear slides*. Retrieved 2020, from LinTech Motion Control: https://www.lintechmotion.com/products2.cfm?ModelNo=140&t=Group11

Logitech. (2020). *C270 HD Webcam*. Retrieved from Logitech: https://www.logitech.com/en-us/products/webcams/c270-hd-webcam.960-000694.html

Ludlum Measurements Inc. (2018). Ludlum 44-9 Alpha, Beta, Gamma Detector. Stillwater, Texas.

Ludlum Measurements Inc. (2020). Ludlum 2200 Scaler Ratemeter. Stillwater, Texas.

Marianno, C. (2019). Lecture 7: Gamma Ray Spectroscopy with Semiconductor Detectors. College Station, Texas, USA.

Microsoft. (2020). *Getting Started*. Retrieved from Visual Studio Code: https://code.visualstudio.com/docs

Mirion Technologies. (2016). GENIE 2000 Basic Spectroscopy Software. San Ramon, California.

Mirion Technologies. (2016). GENIE 2000 Gamma Analysis Software. San Ramon, California.

Mirion Technologies. (2017). Falcon 5000.

Mirion Technologies. (2017). InSpector™ 2000 DSP Portable Spectroscopy Workstation. San Ramon, California.

Mirion Technologies. (2017). Osprey™ Universal Digital MCA Tube Base for Scintillation Spectrometry. San Ramon, California.

National Instruments. (2016). LabVIEW 2018 User Manual. Austin, Texas.

National Instruments. (2016). User Manual NI ISM-7411/7412 Ethernet Integrated Stepper. Austin, Texas.

Ramya, M. V., Purushothama, G. K., & Prakash, K. R. (2020, March). Design and Implementation of IoT Based Remote Laboratory for Sensor Experiments. *Journal of Interactive Mobile Technologies, 14*(9), 227-238.

Schmidt, R., & Umetani, N. (2014). Branching Support Structures for 3D Printing. *ACM DL Digital Library, 2014*(9), 1.

School Smart. (2020). *View Larger Image By:School Smart School Smart Railroad Board, 22 x 28 Inches, 6-Ply, White, Pack of 25*. Retrieved from School Specialty: https://www.schoolspecialty.com/school-smart-railroad-board-22x28-inches-6-ply-white-pack-of-25-1485742

Scientific American. (2020). *How exactly does light transform into heat--for instance, when sunlight warms up a brick wall? I understand that electrons in the atoms in the wall absorb the light, but how does that absorbed sunlight turn into thermal energy?* Retrieved from Scientific American: https://www.scientificamerican.com/article/how-exactly-does-light-tr/

Simplify3D. (2020). *Materials Guide: ABS*. Retrieved from Simplify3D: https://www.simplify3d.com/support/materials-guide/abs/

Simplify3D. (2020). *Materials Guide: PLA*. Retrieved from Simplify3D: https://www.simplify3d.com/support/materials-guide/pla/

Simplify3D. (2020). *Working with File Types*. Retrieved from Simplify3D: https://www.simplify3d.com/support/articles/working-with-file-types/

Walmart. (2020). *Elmer's Tri-fold Foam Display Board, White, 28"x40"*. Retrieved from https://www.walmart.com/ip/Elmer-s-Tri-fold-Foam-Display-Board-White-28-x-40/17011434

APPENDIX A

PHYSICAL SECURITY EXPERIMENT PROCEDURES, INSTRUCTIONAL VIDEO SLIDES, AND INSTRUCTIONAL VIDEO TRANSCRIPTS.

Light Sensors and Material Reflectivity

1. Purpose

This experiment is designed to demonstrate how to use a light sensor to study the color-dependent reflectivity of light. The amount of light reflected from a surface depends on the type of material and the color of the surface. Light that is not reflected off of a surface is absorbed as heat in the material. Understanding the way light is reflected off of different materials allows for the use of light sensors in security applications, such as by processing images based on varying reflectivity and designing appropriate electronic housings to avoid overheating.

2. Lab Prep

- Create the following table in your laboratory notebook:

Material	Light Intensity (Lux)	Percent Reflectivity (%)
Aluminum		100
Copper		
Lead		
Red Paper		
Yellow Paper		
Green Paper		
Blue Paper		
Purple Paper		
White Paper		
Black Paper		

- Make a preliminary list of the provided materials from most reflective to least reflective and explain your rationale.

3. Materials Needed

The following will be provided by the instructor:

- Rotary Table
- LEGO Mindstorms EV3 Brick
- LEGO Mindstorms EV3 Color Sensor
- Aluminum, copper, and lead squares
- Set of Assorted-Color Paper Squares

Students should have the following:

- Calculator
- Laboratory Notebook
- Pen

4. Experimental Procedure

4.0 Set Up the Remote Desktop Connection to the Laboratory Computer

1. VPN into the TAMU network using the Cisco AnyConnect Secure Mobility Client. Use these instructions if you have not done this before: https://agrilifelaserfiche.tamu.edu/documents/tamu-connect-vpn.pdf/ https://it.tamu.edu/services/network-and-internet-access/virtual-private-networks/virtual-private-network-vpn/
2. Open the Remote Desktop client on your computer. For Windows users, you should already have the remote desktop program installed on your computer. For Mac users, use “Microsoft Remote Desktop” in the app store.
3. Your instructor should have given you an IP address of the laboratory computer. Use this IP address for connection.
4. Sign in with your institution’s login information.
5. Open the “camera” application and change the camera to the Logitech webcam in the upper right-hand corner.
6. Download and open the Light Sensor LabVIEW program given to you by the professor/TA. **DO NOT EDIT ANY OF THE CODE. If you edit the code, please return to an original version of the program.**
7. Organize these windows to ensure you can see everything on your screen.

4.1 Establish the Connection with the EV3 Brick

1. Launch Visual Studio Code
2. In the taskbar, navigate to File ➔ Open Folder and select the “LastName_LS_Lab” folder that contains the code you prepared.
3. Under the explorer tab on the left-hand side of Visual Studio Code, find the EV3DEV DEVICE BROWSER and click on its drop-down arrow.

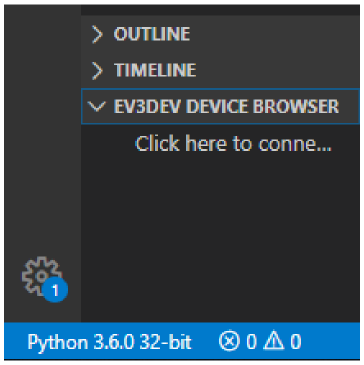


Figure 1. Explorer tab used to connect the EV brick to Visual Studio Code.

4. Select "Click here to connect to a device" and a searchable window should appear at the top of the window. Select "ev3dev" to connect to the EV3 Brick. If you receive a Windows Security Alert, click "cancel" and continue with the procedures.
5. Ensure that there is a green circular light in the EV3DEV DEVICE BROWSER drop-down menu. In this menu, you should see all of the contents of the EV3 Brick.

4.2 Controlling the Rotary Table

1. Run the Light Sensor LabVIEW program by selecting the right-pointing arrow in the upper left-hand corner of the LabVIEW window.
2. In the Material drop-down menu, there is a list of all of the materials that will be used in the experiment. Ensure that Aluminum is already in place by selecting it in the drop-down menu and pressing "Start Move". If you receive an error and the program stops running, do not worry; Aluminum is already set in place.

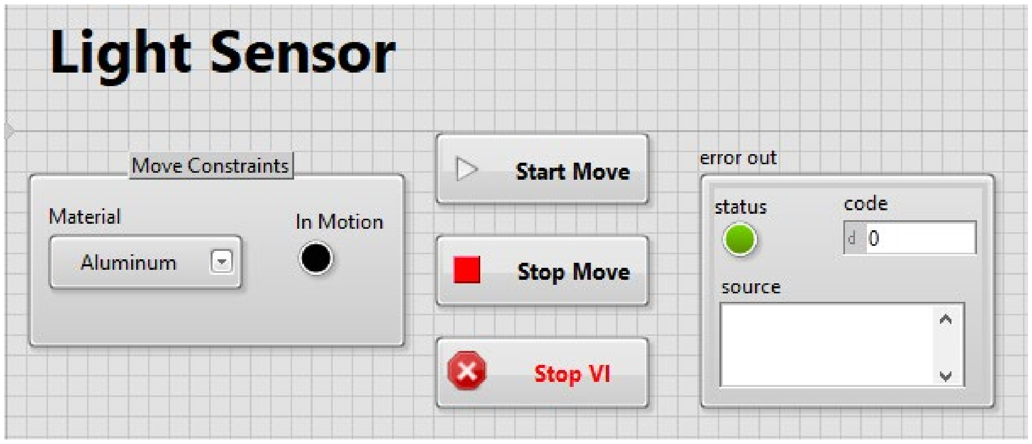


Figure 2. Light sensor LabVIEW front panel with the controls to run the program.

4.3 Collecting the Data

1. Run the Python program by pressing F5 on your keyboard to retrieve the light intensity value. Alternatively, in the toolbar you can go to Run ➔ Start Debugging to run the program.
2. Record the light intensity for Aluminum in the Light Intensity (Lux) column of the table you created in your lab notebook.
3. Repeat steps 4.3.1 – 4.3.2 by cycling the rotary table to measure the remaining materials
4. Observe the sensor values you have recorded. Do the values make sense when compared to each other?
5. Return the rotary table back to Aluminum as a courtesy to the next student.
6. Stop the LabVIEW program by clicking on red stop symbol in the taskbar or the Stop VI button.
7. Record any sources of error in your lab notebook. How might these sources of error affect the data?

4.4 Analyzing the Data

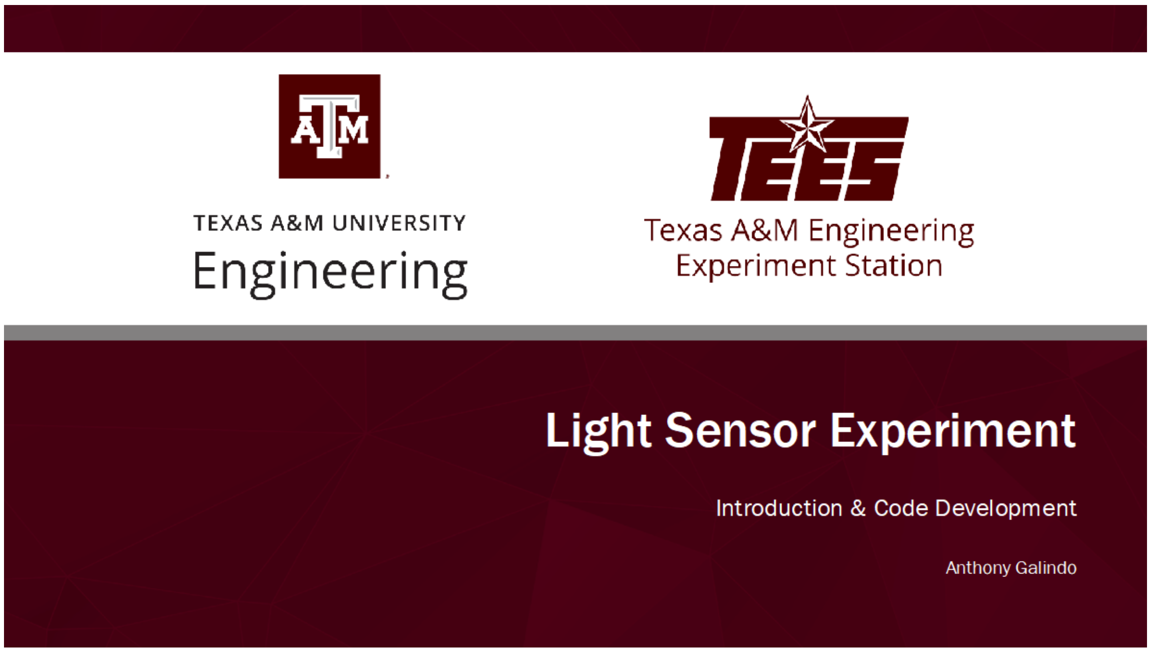
1. Calculate the percent reflectivity of each material using Eq. 1. The percent reflectivity of each material will be based on the amount of light reflected by Aluminum. Write these values in the appropriate column of the table you made in your lab notebook.

$$\% \text{ Reflectivity} = \frac{\text{Amount of Light from Sample}}{\text{Amount of Light from Aluminum}} \times 100 \qquad \text{(Eq. 1)}$$

2. Based on the results, discuss which materials absorb the most light. When would materials be difficult to reimage in a security application?
3. Now, imagine a scenario in which an unattended monitoring system needs to be used in an outdoor environment where it is dry and arid. What color material would need to be chosen as the casing of the system to ensure that it does not overheat?
4. How would the results change for aluminum, copper, and lead if the materials were more polished? What about if they were less polished? Discuss the implications of material degradation on reimaging these materials.

4.5 Leaving the Remote Laboratory

1. Close out of all windows.
2. Be sure to sign out of the computer before exiting out of the remote desktop client.



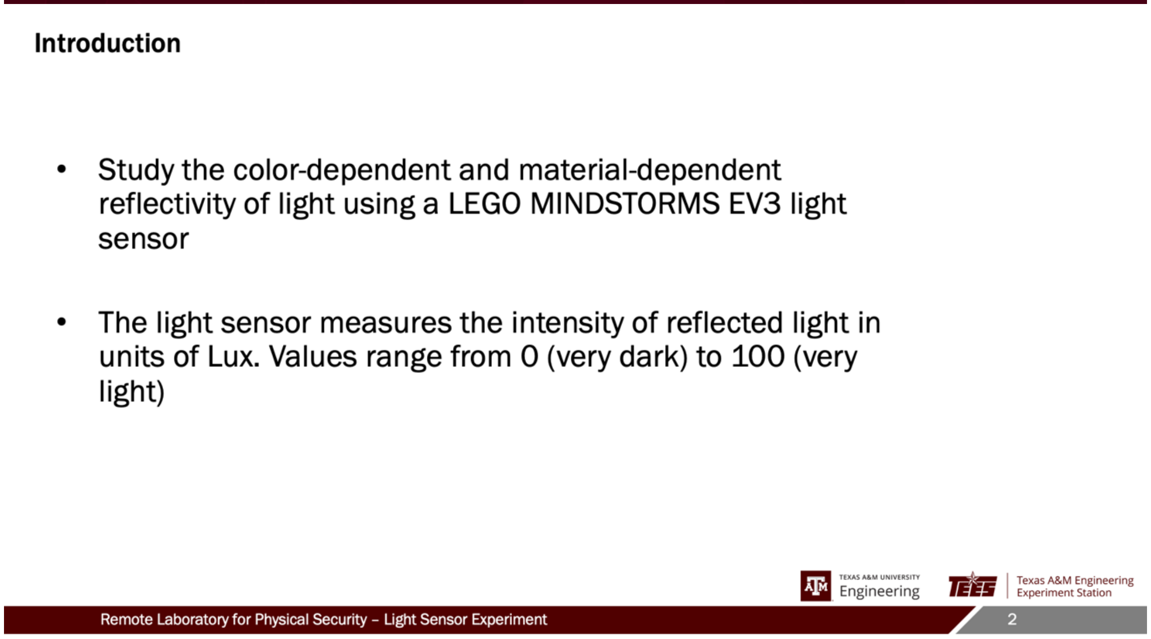
TEXAS A&M UNIVERSITY
Engineering

Texas A&M Engineering
Experiment Station

Light Sensor Experiment

Introduction & Code Development

Anthony Galindo



Introduction

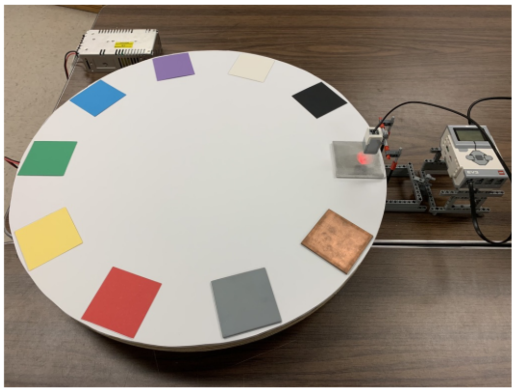
- Study the color-dependent and material-dependent reflectivity of light using a LEGO MINDSTORMS EV3 light sensor

- The light sensor measures the intensity of reflected light in units of Lux. Values range from 0 (very dark) to 100 (very light)

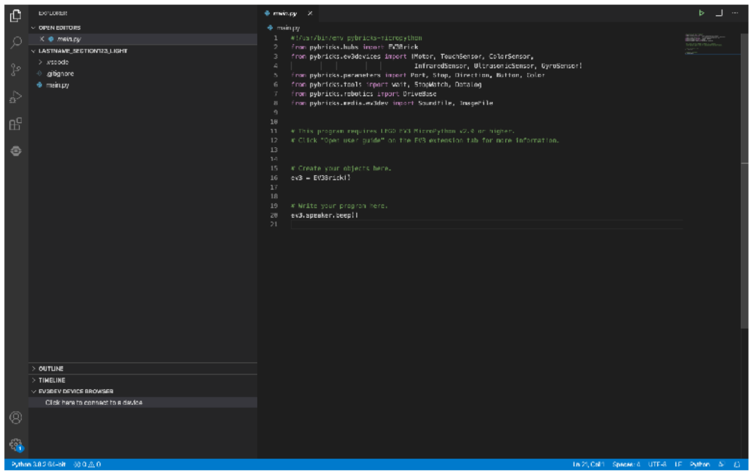
Remote Laboratory for Physical Security – Light Sensor Experiment 2

Introduction

- Rotary table allows for circular rotation of colors and materials
- Fixed light sensor hovers over the rotary table
- An intensity value will be measured and recorded for each color and material in units of Lux

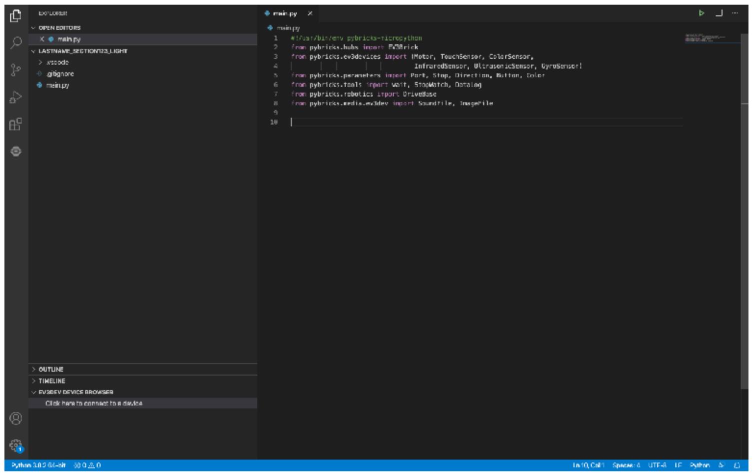


Code Development



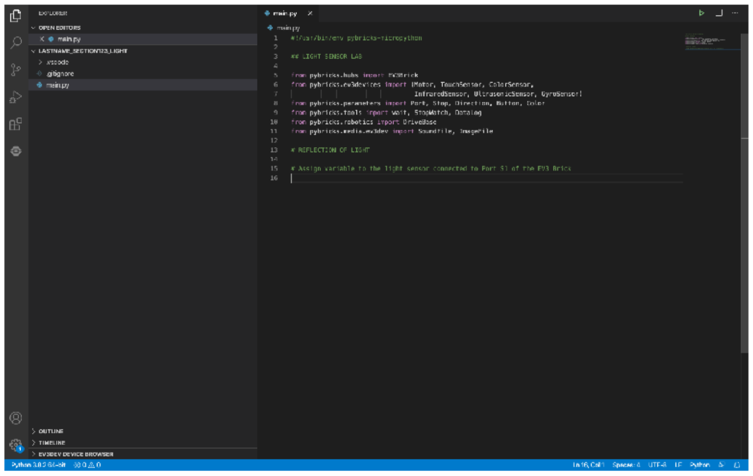
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile


# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.


# Create your objects here.
ev3 = EV3Brick()


# Write your program here.
ev3.speaker.beep()
```

Code Development



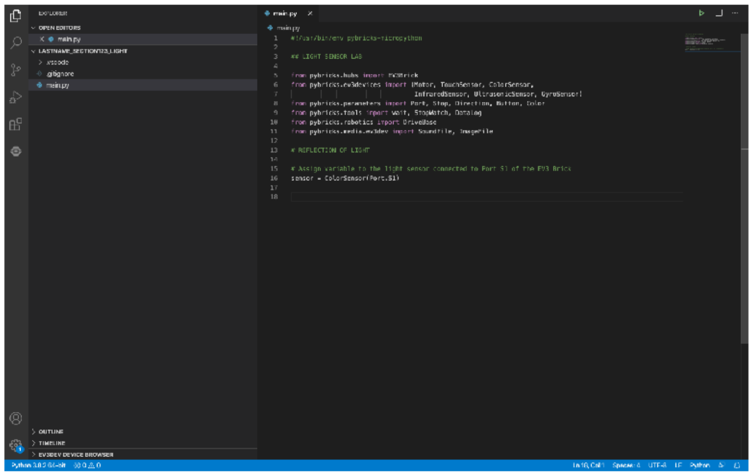
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

```

Code Development



```
#!/usr/bin/env pybricks-micropython

## LIGHT SENSOR LAB

from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# REFLECTION OF LIGHT

# Assign variable to the light sensor connected to Port S1 of the EV3 Brick

```

Code Development

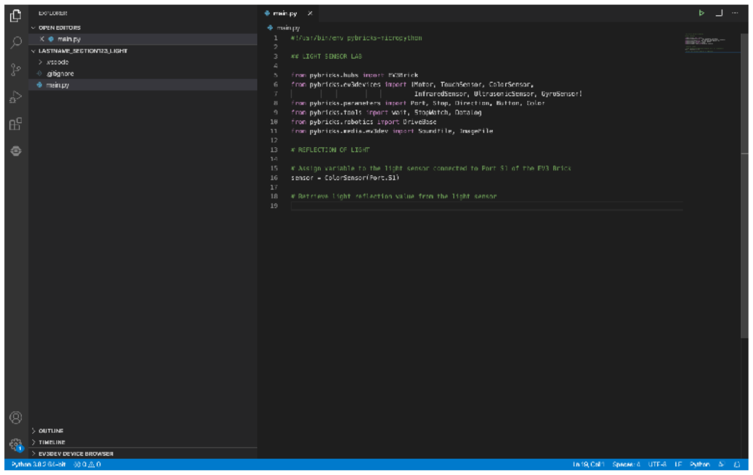


```python
#!/usr/bin/env pybricks-micropython

## LIGHT SENSOR LAB

from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# REFLECTION OF LIGHT

# Assign variable to the light sensor connected to Port S1 of the EV3 Brick
sensor = ColorSensor(Port.S1)

```

Texas A&M University Engineering | TEES Texas A&M Engineering Experiment Station

Remote Laboratory for Physical Security – Light Sensor Experiment 7

Code Development

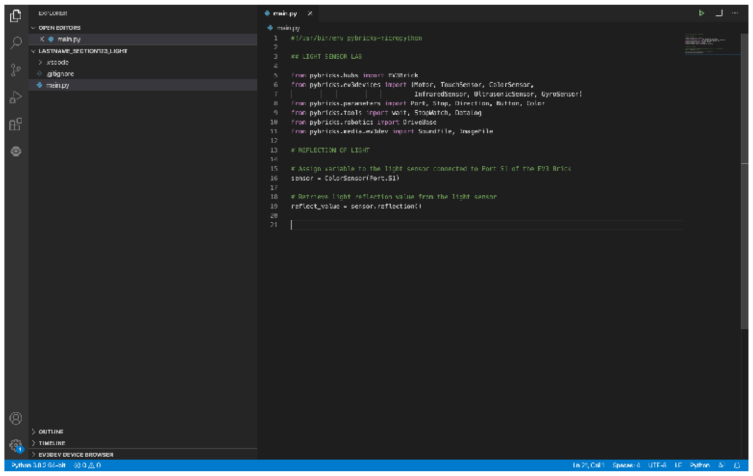


```python
#!/usr/bin/env pybricks-micropython

## LIGHT SENSOR LAB

from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# REFLECTION OF LIGHT

# Assign variable to the light sensor connected to Port S1 of the EV3 Brick
sensor = ColorSensor(Port.S1)

# Retrieve light reflection value from the light sensor

```

Texas A&M University Engineering | TEES Texas A&M Engineering Experiment Station

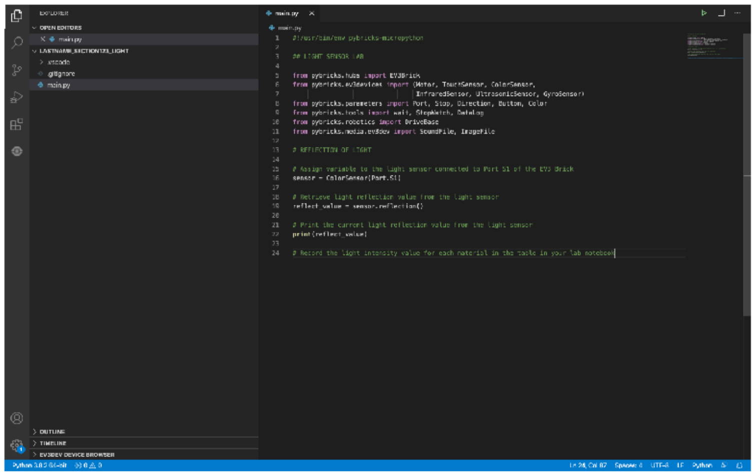
Remote Laboratory for Physical Security – Light Sensor Experiment 8

Code Development



```python
#!/usr/bin/env pybricks-micropython

## LIGHT SENSOR LAB

from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# REFLECTION OF LIGHT

# Assign variable to the light sensor connected to Port S1 of the EV3 Brick
sensor = ColorSensor(Port.S1)

# Retrieve light reflection value from the light sensor
reflect_value = sensor.reflection()

```

Code Development



```python
#!/usr/bin/env pybricks-micropython

## LIGHT SENSOR LAB

from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# REFLECTION OF LIGHT

# Assign variable to the light sensor connected to Port S1 of the EV3 Brick
sensor = ColorSensor(Port.S1)

# Retrieve light reflection value from the light sensor
reflect_value = sensor.reflection()

# Print the current light reflection value from the light sensor
print(reflect_value)

# Record the light intensity value for each material in the table in your lab notebook
```

TEXAS A&M UNIVERSITY
Engineering

TEES
Texas A&M Engineering
Experiment Station

Questions?

Contact: anthgalindo@tamu.edu

Light Sensor Video Transcript

INTRODUCTION

SLIDE 1
This video will walk through the setup of the code needed to complete the light sensor experiment.

SLIDE 2
In this experiment, a light sensor will be used to study the color-dependent and material-dependent reflectivity of light. You have often heard to avoid wearing dark colors on a sunny day. Why is that? Light that is not reflected off of a surface is absorbed as heat in the material. Darker colors do not reflect light as well as lighter colors; thus, they absorb more heat in the material. The LEGO MINDSTORMS EV3 light sensor outputs light intensity values in units of Lux.

SLIDE 3
This concept of light reflectivity will be demonstrated in the experiment by measuring a variety of colors and materials with a LEGO MINDSTORMS EV3 light sensor. A rotary table will be used to cycle the different colors and materials under a fixed light sensor. A value of intensity will then be measured and recorded for each color and material.

CODE WALK THROUGH

SLIDE 4
In main.py, the first line sets up the environment that the code will run in. This environment is pybricks-micropython, an optimized version of Python that LEGO MINDSTORMS uses for its EV3 functions. The next seven lines import functions from their respective libraries to be used in the code. These seven lines of imports cover all of our bases for the functions we will use to control the light sensor in this experiment.

SLIDE 5
First, let's delete the remaining code in this file since it's not needed for this experiment. We'll start by commenting some sections of the code.

SLIDE 6
Let's start with "## LIGHT SENSOR LAB" to label what this code is for. Next, "# REFLECTION OF LIGHT" to focus on the purpose of this code. Commenting your code helps to organize your thoughts and explain the purpose of each line. The light sensor is connected to the EV3 brick through Port S1. Let's go ahead and comment this into the code. To implement this in the code, we have to assign a variable to the light sensor that indicates the port being used.

SLIDE 7
We will do this by creating a variable named "sensor" and setting it equal to LEGO's light sensor function. It is called "ColorSensor()"; note that Color and Light are used interchangeably

when describing this type of sensor. Within the “ColorSensor()” function, we must indicate the port that the sensor is using, so we will put “Port.S1” in the function.

SLIDE 8
Next, we want to retrieve the light intensity value from the light sensor, so let’s start with commenting that into the code. The light intensity value is a measure of how much light is reflected by the color/material using units of Lux. The lower the value, the less light that is reflected off of the material.

SLIDE 9
To get this intensity value, we need to create another variable, so let’s call it “reflect_value.” To get this value, we need to perform an action on our previously-defined variable, “sensor.” The “sensor” variable is our gateway to the light sensor. To retrieve this value from the light sensor, we will set the “reflect_value” variable equal to “sensor.reflection(),” where the reflection function acts on the “sensor” variable to retrieve the light intensity value. The parentheses will remain empty because no further input is needed by the function. Now that the light intensity value has been retrieved from the light sensor, we need to be able to see it ourselves.

SLIDE 10
We will do this by printing the value to the output of the code. Let’s comment this and then use the print function on the next line, “print(reflect_value).” This code will need to be executed 10 times - once for each material - throughout the experiment. Each execution will yield a single light intensity value that characterizes the material being tested.

SLIDE 11
Details explaining how to run the code are included in the experiment procedures. Please contact a teaching assistant with any questions and concerns.

Ultrasonic Sensors for Velocity Determination

1. Purpose

Ultrasonic sensors can be used to detect motion in security and surveillance applications. To understand how ultrasonic sensors work, the student will use a linear actuator to measure a moving object’s position at certain time increments to determine its velocity. The LEGO MINDSTORMS ultrasonic sensor sends an ultrasonic sound from its transmitter and receives a reflection of this sound in its receiver after it has bounced off of nearby objects. This transmission of ultrasonic sound creates air pressure vibrations which are detected and converted into an electric signal. This signal is then used by the EV3 to estimate the distance between the sensor and the object in units of millimeters. The experiment will be conducted with three different velocities.

2. Lab Prep

- Create (3) of the following headers in an Excel spreadsheet, one for each of the three parts of the experiment. Preferably, there should be a separate sheet in the Excel file for each part.

Time (s)	Distance (mm)

- Watch the “Ultrasonic Sensor Code Development” instructional video to prepare the code that will be used to gather data in the experiment.

3. Materials Needed

The following will be provided by the instructor:

- Linear Actuator
- LEGO Mindstorms EV3 Brick
- LEGO Mindstorms EV3 Ultrasonic Sensor
- Flat Object

Students should have the following:

- Calculator
- Laboratory Notebook
- Pen

4. Experimental Procedure

4.0 Set Up the Remote Desktop Connection to the Laboratory Computer

8. VPN into the TAMU network using the Cisco AnyConnect Secure Mobility Client. Use these instructions if you have not done this before: https://agrilifelaserfiche.tamu.edu/documents/tamu-connect-vpn.pdf/ https://it.tamu.edu/services/network-and-internet-access/virtual-private-networks/virtual-private-network-vpn/
9. Open the Remote Desktop client on your computer. For Windows users, you should already have the remote desktop program installed on your computer. For Mac users, use “Microsoft Remote Desktop” in the app store.
10. Your instructor should have given you an IP address of the laboratory computer. Use this IP address for connection.
11. Sign in with your institution’s login information.
12. Open the “Camera” application and change the camera to the Logitech webcam in the upper right-hand corner.
13. Download the three Ultrasonic Sensor LabVIEW programs given to you by the professor/TA. **DO NOT EDIT ANY OF THE CODE. If you edit the code, please return to an original version of the program.**
14. Organize these windows to ensure you can see everything on your screen.

4.1 Establish the Connection with the EV3 Brick

6. Launch Visual Studio Code
7. In the taskbar, navigate to File ➔ Open Folder and select the "LastName_Ultrasonic_Lab" folder that contains the code you prepared.
8. Under the explorer tab on the left-hand side of Visual Studio Code, find the EV3DEV DEVICE BROWSER and click on its drop-down arrow. This is shown in Figure 1.

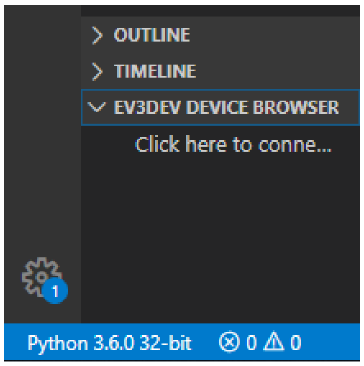


Figure 1. Explorer tab used to connect the EV brick to Visual Studio Code.

9. Select "Click here to connect to a device" and a searchable window should appear at the top of the window. Select "ev3dev" to connect to the EV3 Brick. If you receive a Windows Security Alert, click "cancel" and continue with the procedures.
10. Ensure that there is a green circular light in the EV3DEV DEVICE BROWSER drop-down menu. In this menu, you should see all of the contents of the EV3 Brick.

4.2 Controlling the Linear Actuator

3. Open the "Ultrasonic Sensor Lab A" LabVIEW program. Ensure you have the correct LabVIEW program open by comparing your screen to Figure 2.
4. Run the program by selecting the right-pointing arrow in the upper left-hand corner of the LabVIEW window.
5. In the Object Position drop-down menu, there is a Position 1 and a Position 2. Ensure that the object is in Position 1 by selecting it in the drop-down menu and pressing "Start Move." If you receive an error and the program stops running, do not worry; the object is already in Position 1.

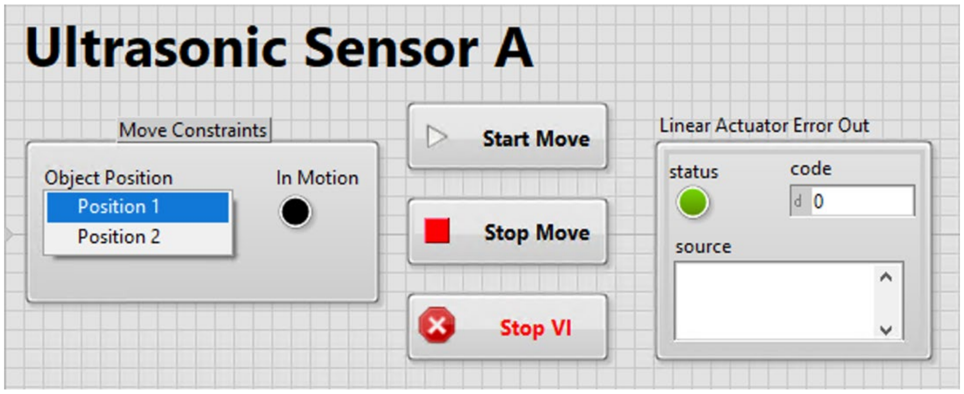


Figure 2. Front panel of the LabVIEW program for Part A of the experiment.

4.3 Collecting the Data

8. To begin, move the object from Position 1 to Position 2. Record the number of seconds it takes to complete the movement, ΔTime.
9. Move the object back to Position 1.
10. Set the time_step variable in the Python program to 500 (units are in milliseconds) and save the program.
11. Use the ratio shown in Eq. 1 to find the number of steps needed and update the "steps" variable in your program with this value.

$$\text{Steps} = \frac{\Delta\text{Time}}{\text{time\_step}} \qquad \text{(Eq. 1)}$$

12. Run the Python program by clicking F5 on your keyboard. Alternatively, in the toolbar you can go to Run ➔ Start Debugging to run the program. Next, move the object from Position 1 to Position 2. If the Python program stops running before the object comes to a stop at Position 2, adjust the ΔTime value in Eq. 1 to allow for more movement time and a larger calculated "steps" value.
13. Observe the sensor values as the object gets closer to the ultrasonic sensor. Note that the distance unit is in millimeters. Do the values make sense?
14. Copy and paste the time column and the ultrasonic sensor values column to an Excel spreadsheet.
15. Repeat steps 4.3.1 – 4.3.7 for Part B and Part C by running the respective LabVIEW programs. **Remember to update the number of steps using Eq. 1 in Part B and Part C**. Save the Python program before running.
16. Return the object back to Position 1 as a courtesy to the next student.
17. Stop the LabVIEW program by clicking on red stop symbol in the taskbar or the Stop VI button.

18. Record any sources of error in your lab notebook. How might these sources of error affect the data?

4.4 Analyzing the Data

1. In the Excel spreadsheet for Part A, create a column for the uncertainty of each sensor value. The uncertainty of these distance values is reported by LEGO to be ± 10 mm.
2. Create a “Scatter with Straight Lines” plot of distance vs. time. Add the appropriate trendline and R^2 value. Be sure to also include 2σ error bars that represent the uncertainty values.
3. Recall the relationship between velocity, distance, and time. What does the slope of the line in the plot represent? What does the R^2 value imply?
4. Calculate the uncertainty in the slope of the distance vs. time plot using Excel’s LINEST function. Does this calculated value make sense?
5. Repeat steps 4.4.1-4.4.4 for Part B and Part C.
6. What do you notice happens from Part A through Part C? Do your results support what you observed during the experiment?
7. Record the calculated velocity and its uncertainty in units of mm/s for each part in your lab notebook. The distance units given by the ultrasonic sensor are in mm, so be sure to make the appropriate conversion. You will need the velocity in mm/s for the infrared sensor laboratory.

4.5 Leaving the Remote Laboratory

3. Close out of all windows.
4. Be sure to sign out of the computer before exiting out of the remote desktop client.

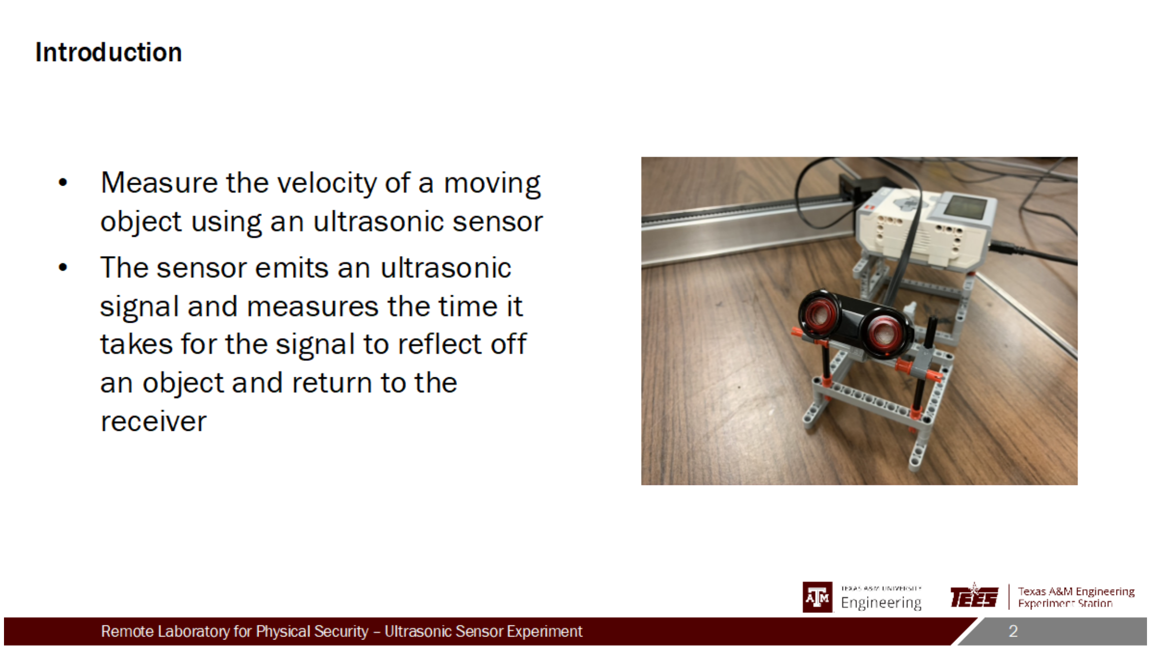
TEXAS A&M UNIVERSITY
Engineering

Texas A&M Engineering
Experiment Station

Ultrasonic Sensor Experiment

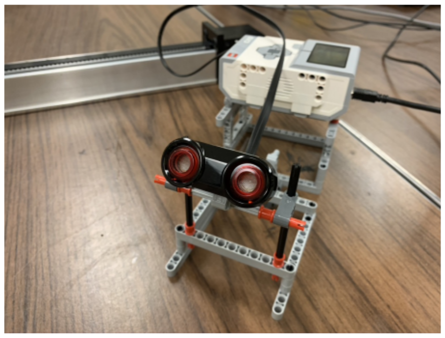
Introduction & Code Development

Anthony Galindo



Introduction

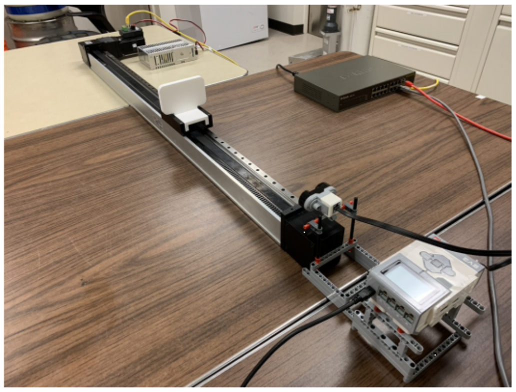
- Measure the velocity of a moving object using an ultrasonic sensor
- The sensor emits an ultrasonic signal and measures the time it takes for the signal to reflect off an object and return to the receiver



Remote Laboratory for Physical Security – Ultrasonic Sensor Experiment 2

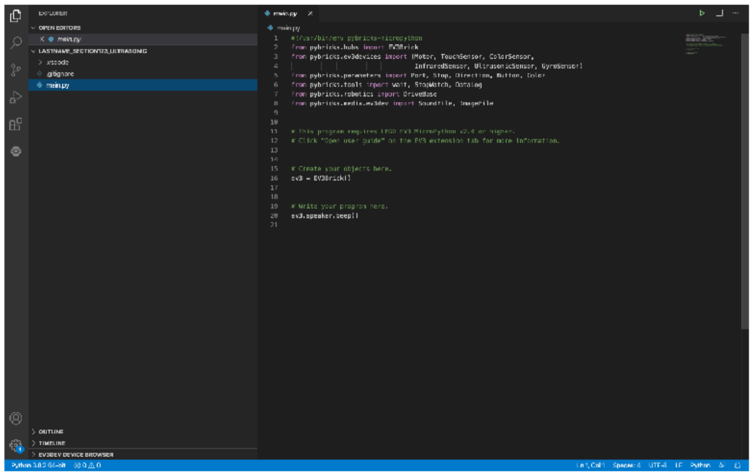
Introduction

- The wall will slide on the linear actuator towards the ultrasonic sensor
- A plot of distance vs time can be produced, and the slope of this line is the velocity
  - Distance units: millimeters
  - Time units: seconds
- There are parts A, B, and C of the experiment, each having a different velocity to determine



Texas A&M University Engineering | TEES | Texas A&M Engineering Experiment Station

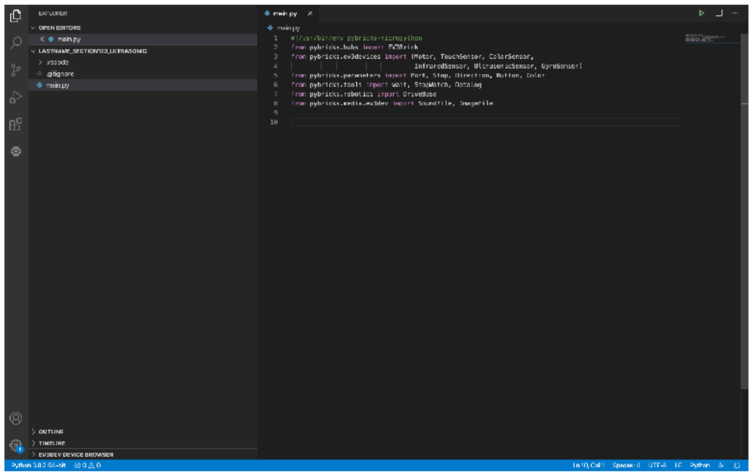
Code Development



```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile


# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.


# Create your objects here.
ev3 = EV3Brick()


# Write your program here.
ev3.speaker.beep()
```

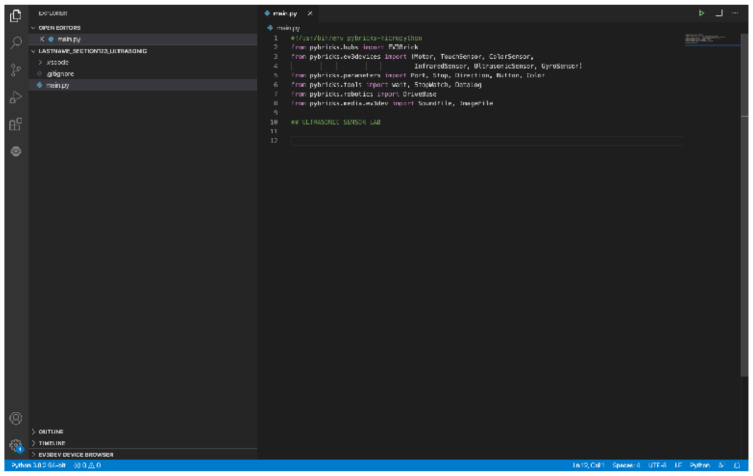
Texas A&M University Engineering | TEES | Texas A&M Engineering Experiment Station

Code Development



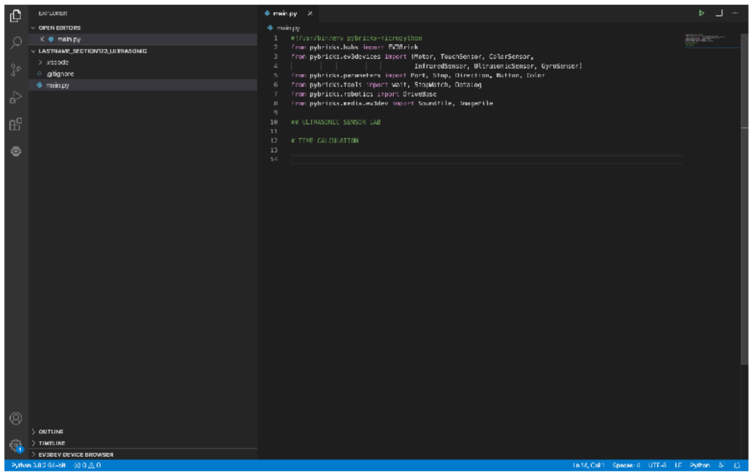
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

```

Code Development



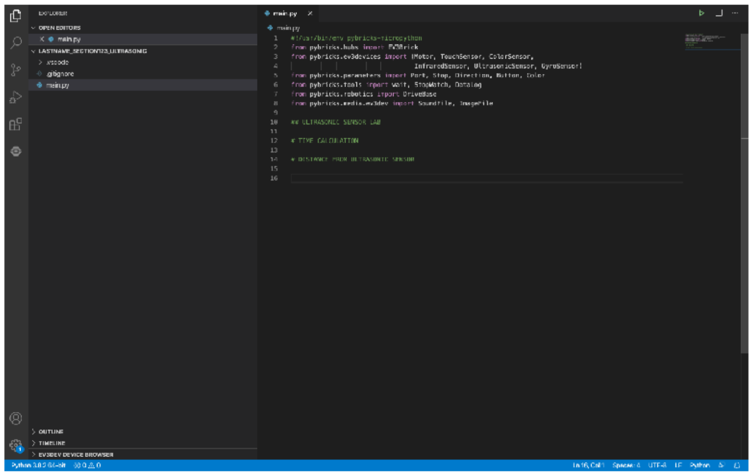
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## ULTRASONIC SENSOR LAB

```

Code Development



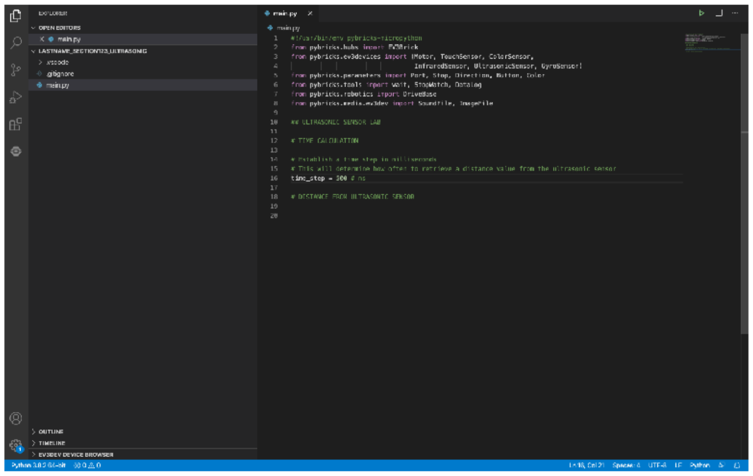
```python
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## ULTRASONIC SENSOR LAB

# TIME CALCULATION

```

Code Development



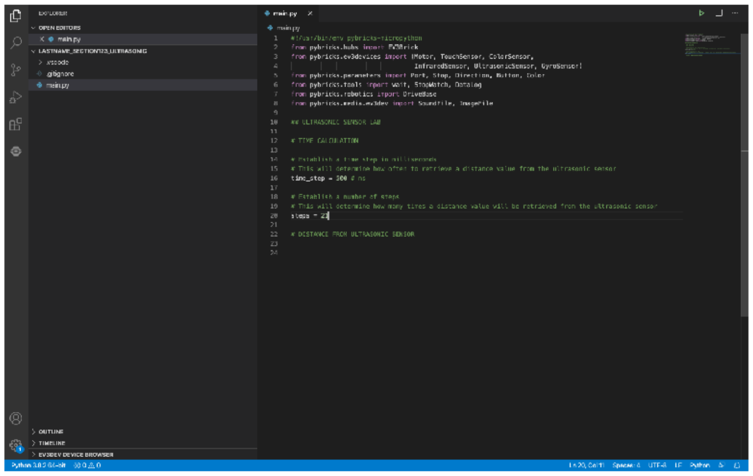
```python
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## ULTRASONIC SENSOR LAB

# TIME CALCULATION

# DISTANCE FROM ULTRASONIC SENSOR

```

Code Development



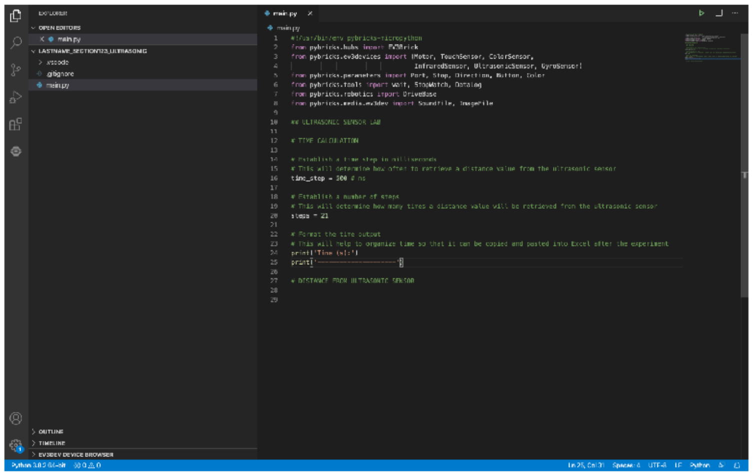
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## ULTRASONIC SENSOR LAB

# TIME CALCULATION

# Establish a time step in milliseconds
# This will determine how often to retrieve a distance value from the ultrasonic sensor
time_step = 500 # ms

# DISTANCE FROM ULTRASONIC SENSOR
```

Code Development



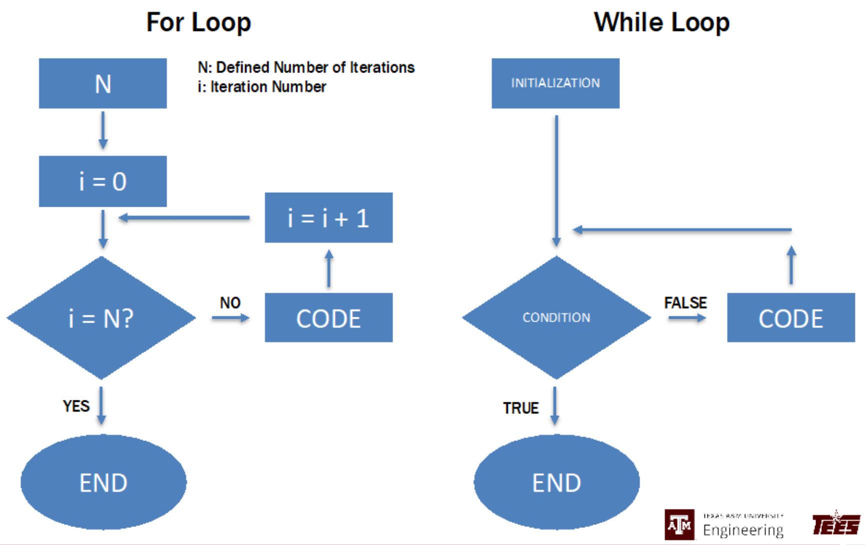
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## ULTRASONIC SENSOR LAB

# TIME CALCULATION

# Establish a time step in milliseconds
# This will determine how often to retrieve a distance value from the ultrasonic sensor
time_step = 500 # ms

# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the ultrasonic sensor
steps = 21

# DISTANCE FROM ULTRASONIC SENSOR
```

Code Development

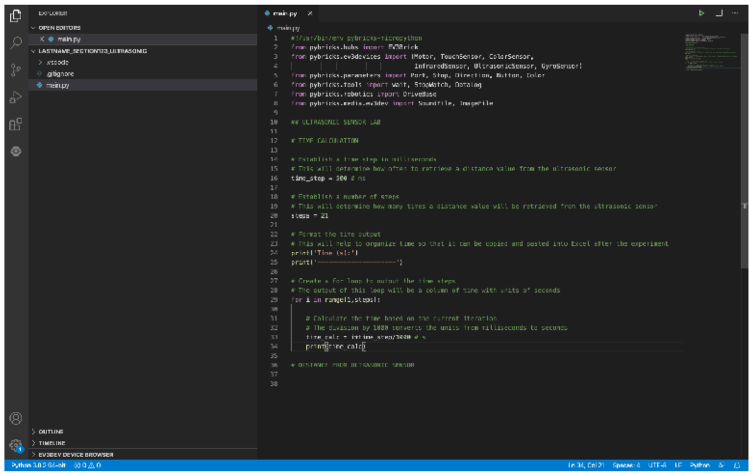


```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## ULTRASONIC SENSOR LAB

# TIME CALCULATION

# Establish a time step in milliseconds
# This will determine how often to retrieve a distance value from the ultrasonic sensor
time_step = 500 # ms

# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the ultrasonic sensor
steps = 21

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('------------------')

# DISTANCE FROM ULTRASONIC SENSOR
```

Code Development

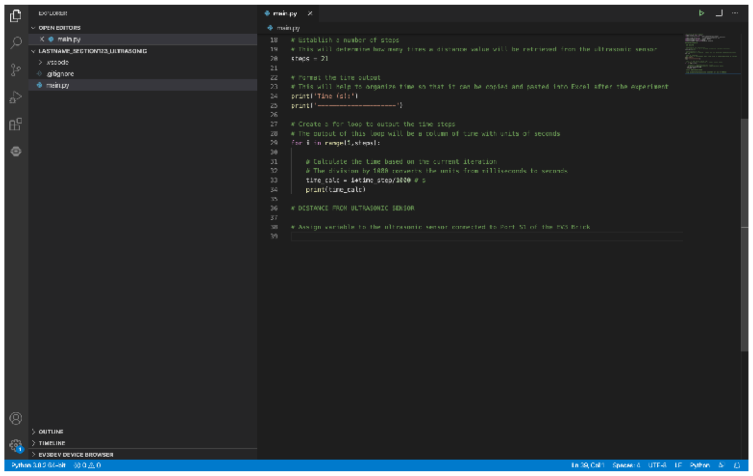


Code Development



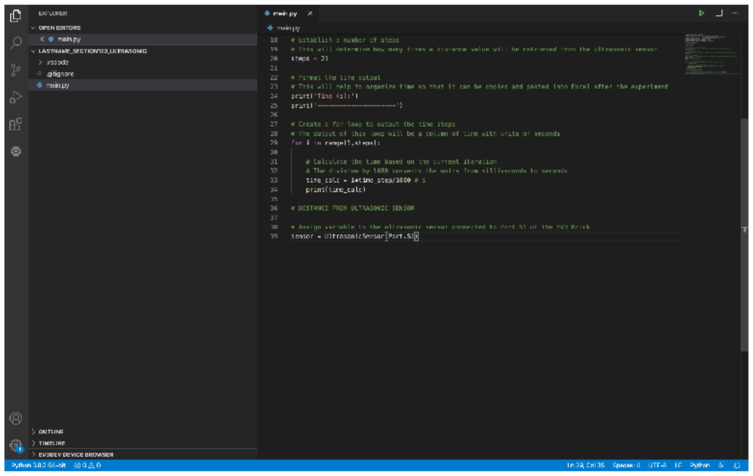
```python
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## ULTRASONIC SENSOR LAB

# TIME CALCULATION

# Establish a time step in milliseconds
# This will determine how often to retrieve a distance value from the ultrasonic sensor
time_step = 500 # ms

# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the ultrasonic sensor
steps = 21

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('-------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s
    print(time_calc)

# DISTANCE FROM ULTRASONIC SENSOR


```

Code Development



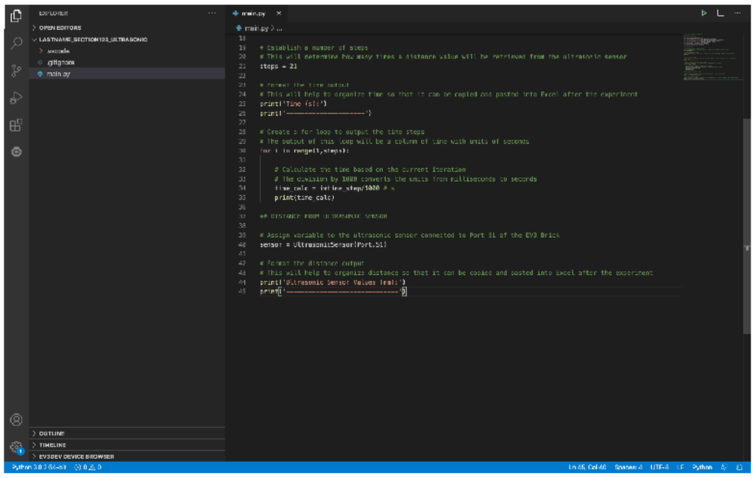
```python
# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the ultrasonic sensor
steps = 21

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('-------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s
    print(time_calc)

# DISTANCE FROM ULTRASONIC SENSOR

# Assign variable to the ultrasonic sensor connected to Port S1 of the EV3 Brick

```

Code Development



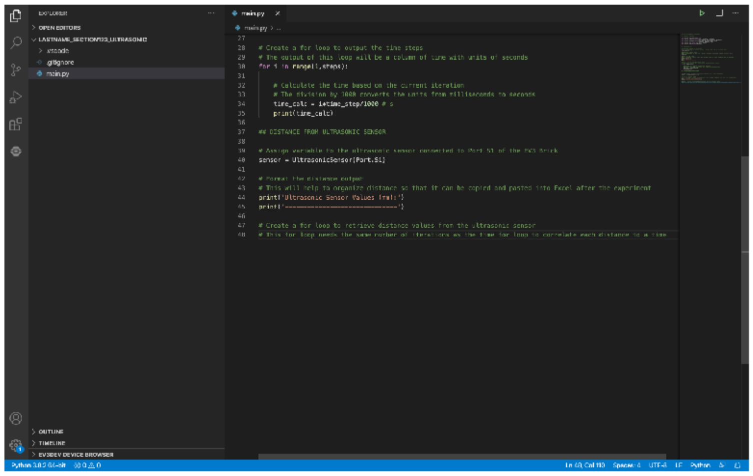
```
# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the ultrasonic sensor
steps = 21

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('-------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s
    print(time_calc)

# DISTANCE FROM ULTRASONIC SENSOR

# Assign variable to the ultrasonic sensor connected to Port S1 of the EV3 Brick
sensor = UltrasonicSensor(Port.S1)
```

Code Development



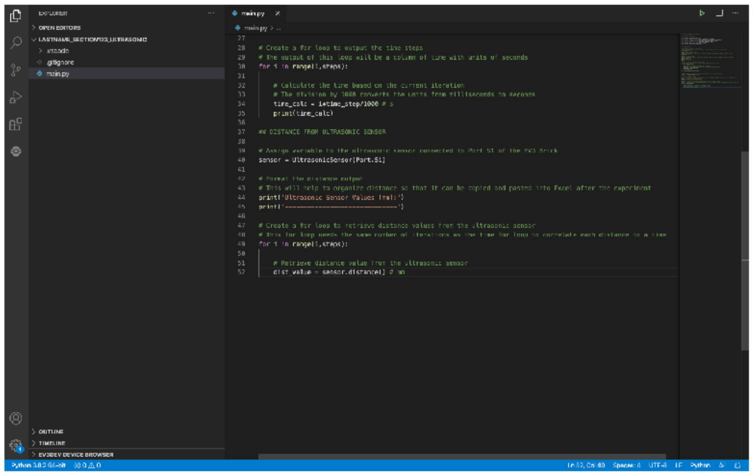
```

# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the ultrasonic sensor
steps = 21

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('-------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s
    print(time_calc)

## DISTANCE FROM ULTRASONIC SENSOR

# Assign variable to the ultrasonic sensor connected to Port S1 of the EV3 Brick
sensor = UltrasonicSensor(Port.S1)

# Format the distance output
# This will help to organize distance so that it can be copied and pasted into Excel after the experiment
print('Ultrasonic Sensor Values (mm):')
print('---------------------------')
```

Code Development



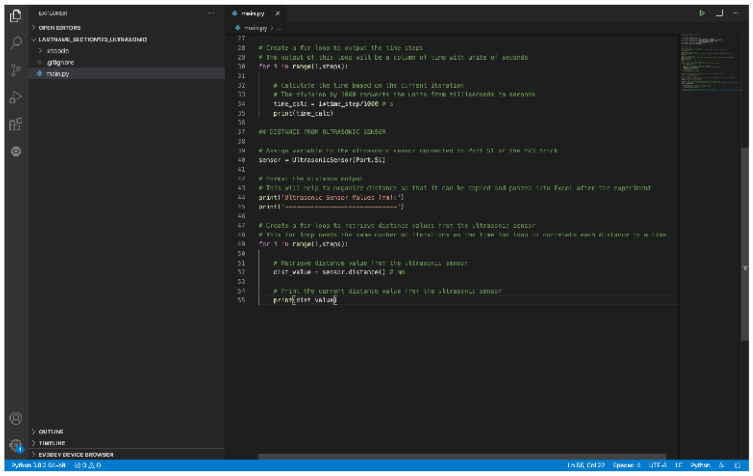
```
# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s
    print(time_calc)

## DISTANCE FROM ULTRASONIC SENSOR

# Assign variable to the ultrasonic sensor connected to Port S1 of the EV3 Brick
sensor = UltrasonicSensor(Port.S1)

# Format the distance output
# This will help to organize distance so that it can be copied and pasted into Excel after the experiment
print('Ultrasonic Sensor Values (mm):')
print('----------------------------')

# Create a for loop to retrieve distance values from the ultrasonic sensor
# This for loop needs the same number of iterations as the time for loop to correlate each distance to a time
```

Code Development



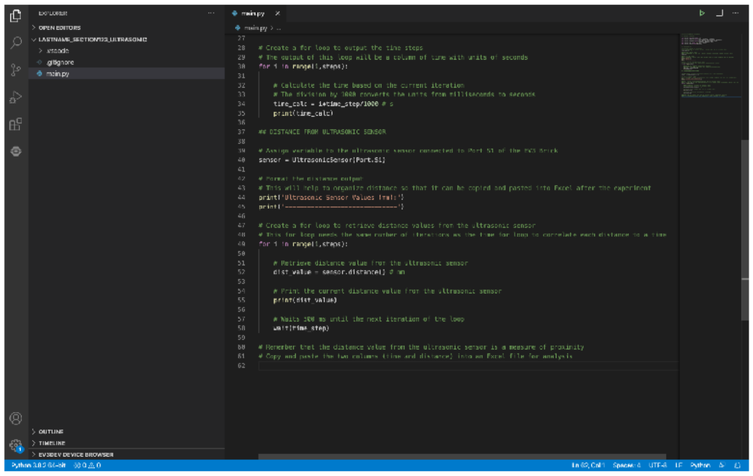
```
# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s
    print(time_calc)

## DISTANCE FROM ULTRASONIC SENSOR

# Assign variable to the ultrasonic sensor connected to Port S1 of the EV3 Brick
sensor = UltrasonicSensor(Port.S1)

# Format the distance output
# This will help to organize distance so that it can be copied and pasted into Excel after the experiment
print('Ultrasonic Sensor Values (mm):')
print('----------------------------')

# Create a for loop to retrieve distance values from the ultrasonic sensor
# This for loop needs the same number of iterations as the time for loop to correlate each distance to a time
for i in range(1,steps):

    # Retrieve distance value from the ultrasonic sensor
    dist_value = sensor.distance() # mm
```

Code Development



```python

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s
    print(time_calc)

## DISTANCE FROM ULTRASONIC SENSOR

# Assign variable to the ultrasonic sensor connected to Port S1 of the EV3 Brick
sensor = UltrasonicSensor(Port.S1)

# Format the distance output
# This will help to organize distance so that it can be copied and pasted into Excel after the experiment
print('Ultrasonic Sensor Values (mm):')
print('-----------------------------')

# Create a for loop to retrieve distance values from the ultrasonic sensor
# This for loop needs the same number of iterations as the time for loop to correlate each distance to a time
for i in range(1,steps):

    # Retrieve distance value from the ultrasonic sensor
    dist_value = sensor.distance() # mm

    # Print the current distance value from the ultrasonic sensor
    print(dist_value)
```

Code Development



```python

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s
    print(time_calc)

## DISTANCE FROM ULTRASONIC SENSOR

# Assign variable to the ultrasonic sensor connected to Port S1 of the EV3 Brick
sensor = UltrasonicSensor(Port.S1)

# Format the distance output
# This will help to organize distance so that it can be copied and pasted into Excel after the experiment
print('Ultrasonic Sensor Values (mm):')
print('-----------------------------')

# Create a for loop to retrieve distance values from the ultrasonic sensor
# This for loop needs the same number of iterations as the time for loop to correlate each distance to a time
for i in range(1,steps):

    # Retrieve distance value from the ultrasonic sensor
    dist_value = sensor.distance() # mm

    # Print the current distance value from the ultrasonic sensor
    print(dist_value)

    # Waits 100 ms until the next iteration of the loop
    wait(time_step)

# Remember that the distance value from the ultrasonic sensor is a measure of proximity
# Copy and paste the two columns (time and distance) into an Excel file for analysis

```

TEXAS A&M UNIVERSITY
Engineering

Texas A&M Engineering
Experiment Station

Questions?

Contact: anthgalindo@tamu.edu

Ultrasonic Sensor Video Transcript

INTRODUCTION

SLIDE 1
This video will walk through the setup of the code needed to complete the ultrasonic sensor experiment.

SLIDE 2
In this experiment, an ultrasonic sensor will be used to measure the velocity of a moving object. The LEGO MINDSTORMS ultrasonic sensor sends an ultrasonic sound from its transmitter and receives a reflection of this sound in its receiver after it has bounced off of a nearby object.

SLIDE 3
To measure the velocity of the object, you will need to know its position over time as it travels on a linear actuator. By continuously recording the distance between the sensor and the object, its position over time will be determined. Be sure to note that the distance output from the ultrasonic sensor is in units of millimeters.

CODE WALK THROUGH

SLIDE 4
In main.py, the first line sets up the environment that the code will run in. This environment is pybricks-micropython, an optimized version of Python that LEGO MINDSTORMS uses for its EV3 functions. The next seven lines import functions from their respective libraries to be used in the code. These seven lines of imports cover all of our bases for the functions we will use to control the ultrasonic sensor in this experiment.

SLIDE 5
First, let’s delete the remaining code in this file since it’s not needed for this experiment. We’ll start by commenting some sections of the code.

SLIDE 6
Let’s start with “## ULTRASONIC SENSOR LAB” to label what this code is for.

SLIDE 7
Next, “# TIME CALCULATION” to focus on the time iterations that will determine how often to retrieve a value from the ultrasonic sensor.

SLIDE 8
Additionally, a “# DISTANCE FROM ULTRASONIC SENSOR” section to focus on retrieving the distance value from the ultrasonic sensor. Commenting your code helps to organize your thoughts and explain the purpose of each line.

SLIDE 9

In the ## TIME CALCULATION section of the code, let's establish a time step in milliseconds. This will determine how often to retrieve a distance value from the ultrasonic sensor. The time step can be adjusted to whatever value you think is appropriate for the experiment. For now, let's set it at 500 milliseconds.

SLIDE 10
Next, we need to establish a number of steps which will be the number of times that data will be collected in the experiment. The number of steps can be changed to however many data points are needed, but for now let's set it at 21 steps. I will explain the reason for 21 steps rather than an even 20 steps soon.

SLIDE 11
Next, we will want to write code that outputs a column of these time steps so that it's easy to copy and paste the values into an Excel spreadsheet. We will do this by printing two lines: one line with a label for Time and its units in seconds, and another line that prints out a dashed line for formatting purposes.

SLIDE 12
Lastly, we need to implement a structure that will calculate and output the time steps. There are for loops and there are while loops. For loops will repeat for a specified number of iterations whereas while loops will repeat a set of actions until a condition is met. Since we have a set number of time steps already defined, we will use a for loop.

SLIDE 13
In the line setting up the for loop, the i is used to represent the iteration passing through the loop. In the first iteration, the first value in the defined range of "1 through steps", (where steps = 21), will pass through the loop. Thus, the time_calc value will yield 1*time_step/1000 which equals 0.5 seconds. This will then be printed in the next line. The next iteration will pass i = 2 through the loop for a time_calc value = 1 second. This will continue until the last iteration. The last iteration will be i = 20 because the range function ignores the very last value in the range. In our case, the range is from 1 through 21, so the loop's last iteration will be for i = 20. This is the reason why the steps variable was assigned 21, to avoid having the loop end on i = 19 if we set steps = 20.

SLIDE 14
The ultrasonic sensor is connected to the EV3 brick through Port S1. Let's go ahead and comment this into the code.

SLIDE 15
To implement this in the code, we have to assign a variable to the ultrasonic sensor that indicates the port being used. We will do this by creating a variable named "sensor" and setting it equal to LEGO's ultrasonic sensor function. It is called "UltrasonicSensor()". Within the "UltrasonicSensor()" function, we must indicate the port that the sensor is using, so we will put "Port.S1" in the function.

SLIDE 16

Next, we will want to write code that outputs a column of the ultrasonic sensor values so that it's easy to copy and paste the values into an Excel spreadsheet. We will do this by printing two lines: one line with a label for Ultrasonic Sensor Values in units of millimeters, and another line that prints out a dashed line for formatting purposes.

SLIDE 17
Next, we want to retrieve the distance value from the ultrasonic sensor, so let's start by commenting that into the code. The distance value is a measure of how far away the object is from the ultrasonic sensor in units of millimeters. We need to implement another for loop to retrieve the same number of distance values as there are time steps.

SLIDE 18
The for loop will be very similar to the for loop that calculates the time steps. To get the distance value, we need to create another variable, so let's call it "dist_value." To get this value, we need to perform an action on our previously-defined variable, "sensor." The "sensor" variable is our gateway to the ultrasonic sensor. To retrieve the distance value, we will set the "dist_value" variable equal to "sensor.distance()," where the distance function acts on the "sensor" variable to retrieve the distance value. The parentheses will remain empty because no further input is needed by the function.

SLIDE 19
It's also important to print the distance value. In each iteration, the print function will output the distance value for that time and it will appear in the ultrasonic sensor value column we set up.

SLIDE 20
Lastly, the for loop needs to wait for the amount of time equal to the time step before iterating again to retrieve another distance value. Luckily, the wait function will do this for us.

SLIDE 21
Details on running the code are included in the experiment procedures. Please contact a teaching assistant with any questions and concerns.

Infrared Sensors for Remote Object Measurements

1. Purpose

This experiment will introduce students to infrared sensors and how they can be used for security applications. With the knowledge of an object's velocity, it is possible to determine the dimensions of an object as it passes by an infrared sensor.

The EV3 infrared (IR) sensor can measure distance or detect signals that are sent from an IR beacon. The IR sensor can be used in three different modes:

1) Proximity: The IR sensor sends an IR signal and detects the reflection of this signal by an object in front of the sensor. The strength of the reflected signal can be used to estimate the distance to the object. The EV3 IR sensor maximum range is approximately 100 cm.
2) Beacon: The IR sensor can detect an IR beacon. The IR sensor can detect the beacon's proximity and its heading (angle from the direction the sensor is pointing).
3) Remote: The infrared sensor can detect button presses on the IR beacon. The IR sensor can detect which button on the remote IR beacon is pressed.

9. In this experiment, the size and shape of an irregular object as it passes by on a linear actuator will be determined based on the proximity readings recorded by the infrared sensor. An image of the irregular object is shown in Figure 1 with its Regions of Interest (ROIs) labeled. The actual measurements of the object are included in Table 1 with a 2 σ uncertainty.

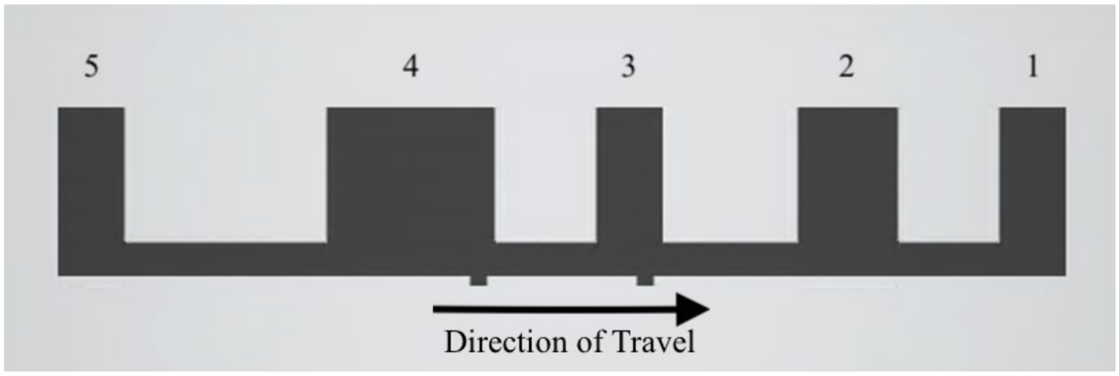


Figure 1. Labeled image of the irregular object used in the infrared sensor experiment.

2. Lab Prep

- Answer the following questions in your lab notebook:
    - In what security situations can knowing the dimensions of a moving object prove useful?
    - Provide an equation for finding the length of an object if the velocity of the object and the time that has passed are known.

- Watch the "Infrared Sensor Code Development" instructional video to prepare the code that will be used to gather data in the experiment.

Create the following table in your lab notebook:

Table 1. Table for actual and estimated ROI measurements of the irregular object.

ROI	Actual Measurement (mm)	Estimated Measurement w/ 500 ms Time Step (mm)	Estimated Measurement w/ 200 ms Time Step (mm)	Estimated Measurement w/ 100 ms Time Step (mm)
1	20.19 ± 0.04			
2	30.19 ± 0.04			
3	20.25 ± 0.04			
4	50.10 ± 0.04			
5	20.04 ± 0.04			

3. Materials Needed

The following will be provided by the instructor:

- Linear Actuator
- LEGO Mindstorms EV3 Brick
- LEGO Mindstorms EV3 Infrared Sensor
- Irregular Comb-Like Object

Students should have the following:

- Calculator
- Laboratory Notebook
- Pen

4. Experimental Procedure

5.0 Set Up the Remote Desktop Connection to the Laboratory Computer

15. VPN into the TAMU network using the Cisco AnyConnect Secure Mobility Client. Use these instructions if you have not done this before: https://agrilifelaserfiche.tamu.edu/documents/tamu-connect-vpn.pdf/

https://it.tamu.edu/services/network-and-internet-access/virtual-private-networks/virtual-private-network-vpn/

16. Open the Remote Desktop client on your computer. For Windows users, you should already have the remote desktop program installed on your computer. For Mac users, use "Microsoft Remote Desktop" in the app store.
17. Your instructor should have given you an IP address of the laboratory computer. Use this IP address for connection.
18. Sign in with your institution's login information.
19. Open the "camera" application and change the camera to the Logitech webcam in the upper right-hand corner.
20. Download and open the Infrared Sensor LabVIEW program on ecampus. **DO NOT EDIT ANY OF THE CODE. If you edit the code, please return to an original version of the program.**
21. Run the program by selecting the right-pointing arrow in the upper left-hand corner of the LabVIEW window.
22. Organize these windows to ensure you can see everything on your screen.

4.1 Establish the Connection with the EV3 Brick

1. Launch Visual Studio Code.
2. In the taskbar, navigate to File ➔ Open Folder and select the "LastName_IR_Lab" folder that contains the code you prepared.
3. Under the explorer tab on the left-hand side of Visual Studio Code, find "EV3DEV DEVICE BROWSER" and click on its drop-down arrow. This is shown in Figure 2.

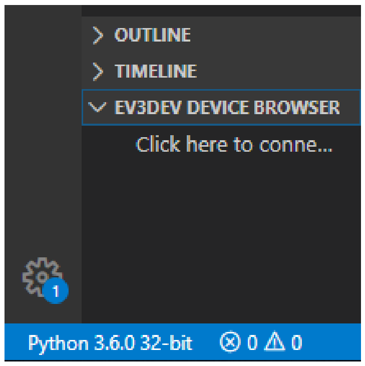


Figure 2. Explorer tab used to connect the EV brick to Visual Studio Code.

4. Select "Click here to connect to a device" and a searchable window should appear at the top of the window. Select "ev3dev" to connect to the EV3 Brick. If you receive a Windows Security Alert, click "cancel" and continue with the procedures.

5. Ensure that there is a green circular light in the EV3DEV DEVICE BROWSER drop-down menu. In this menu, you should see all of the contents of the EV3 Brick.

4.2 Controlling the Linear Actuator

6. Launch the “Infrared Sensor Lab” LabVIEW program. Ensure you have the correct LabVIEW program open by comparing your screen to Figure 3.
7. Run the program by selecting the right-pointing arrow in the upper left-hand corner of the LabVIEW window.
8. In the Object Position drop-down menu, there is a Position 1 and a Position 2. Ensure that the object is in Position 1 by selecting it in the drop-down menu and pressing “Start Move”. If you receive an error and the program stops running, do not worry; the object is already in Position 1.

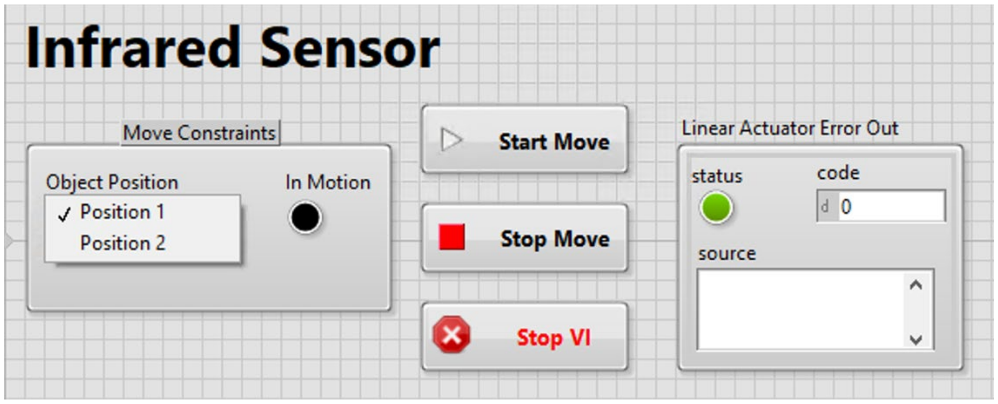


Figure 3. Infrared sensor LabVIEW front panel with the controls to run the program.

4.3 Collecting the Data

19. To begin, move the object from Position 1 to Position 2. The object will stop on its own. Record the number of seconds it takes to complete the movement, ΔTime.
20. Move the object back to Position 1.
21. Set the time_step variable in the Python program to 500 (units are in milliseconds) and save the program.
22. Use the ratio in Eq. 1 to find the number of steps needed:

$$\text{Steps} = \frac{\Delta\text{Time}}{\text{time\_step}} \qquad \text{(Eq. 1)}$$

23. Run the Python program by clicking F5 on your keyboard. Alternatively, in the toolbar you can go to Run ➔ Start Debugging to run the program. Next,

move the object from Position 1 to Position 2. If the Python program stops running before the object has completely passed by the infrared sensor, adjust your ΔTime value to allow for more transition time.

24. Observe the sensor values as the object is passing in front of the infrared sensor. Do the values make sense?
25. Copy and paste the time column and the infrared sensor values column to an Excel spreadsheet.
26. Repeat steps 4.3.3 – 4.3.7 for a time_step of 200 and 100 milliseconds. **Remember to update the number of steps using Eq. 1 for the time_step values of 200 and 100 milliseconds.** Save the Python program before running it with F5.
27. Return the object back to Position 1 as a courtesy to the next student.
28. Stop the LabVIEW program by clicking on red stop symbol in the taskbar or the Stop VI button.
29. Record any sources of error in your lab notebook. How might these sources of error affect the data?

4.4 Analyzing the Data

8. In the Excel spreadsheet for time_step = 500 milliseconds, make a third column next to the infrared sensor values called "Boolean Values." Recall that if the object is completely covering the infrared light emitted by the sensor, the output value will be 0. Otherwise, the output value will be greater than zero. In the first cell of this column, type in an IF statement that will leave a "0" if the object is covering the infrared sensor or a "1" if the object is not covering the infrared sensor. Copy this equation into the rest of the column's cells.
9. Create a "Scatter with Straight Lines" plot of the boolean value versus time. What does the plot show? Is this what you expected?
10. Mark the ROIs by assigning a color to the cells of each group of boolean values equal to 1. Do the number of ROIs match the number of components that the object consisted of?
11. Calculate ΔT_i , the amount of the time that passed in each ROI_i. Assume no uncertainty in ΔT_i.
12. Using your calculated velocity from Part A of the ultrasonic sensor lab, calculate an estimated measurement of each ROI. Be sure to propagate uncertainty.
13. How do the estimated measurements compare to the actual measurements? Be sure to compare using uncertainty.
14. Repeat steps 4.4.1 – 4.4.6 for the remaining time_step trials.
15. What do you notice happens as the time_step is decreased? Compare plots and estimated measurements. What are the advantages and disadvantages of decreasing the time_step?

4.5 Leaving the Remote Laboratory

5. Close out of all windows.
6. Be sure to sign out of the computer before exiting out of the remote desktop client.

TEXAS A&M UNIVERSITY
Engineering

Texas A&M Engineering
Experiment Station

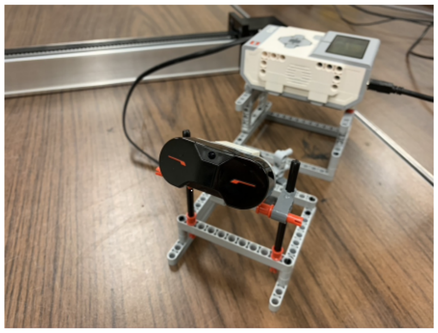
Infrared Sensor Experiment

Introduction & Code Development

Anthony Galindo

Introduction

- Measure a moving object's dimensions using a LEGO MINDSTORMS EV3 infrared sensor
- The sensor measures the reflection of its infrared signal to output a distance value (between the sensor and the object) in units of millimeters

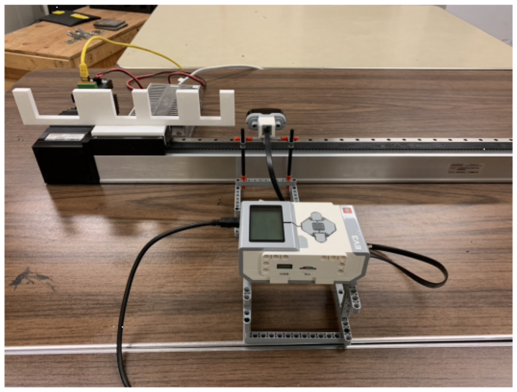


Engineering | Texas A&M Engineering Experiment Station

Remote Laboratory for Physical Security – Infrared Sensor Experiment 2

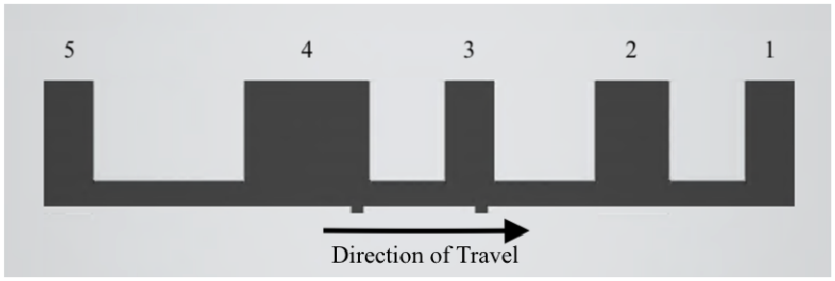
Introduction

- Distance value = 0 mm:
  - Constituent is in front of the infrared sensor
  - Assigned a "1" Boolean value in Excel
- Distance value > 0 mm:
  - Gap between constituents
  - Assigned a "0" Boolean value in Excel
- Plot of Boolean values vs time will show the shape of the object



Texas A&M University Engineering | TEES | Texas A&M Engineering Experiment Station

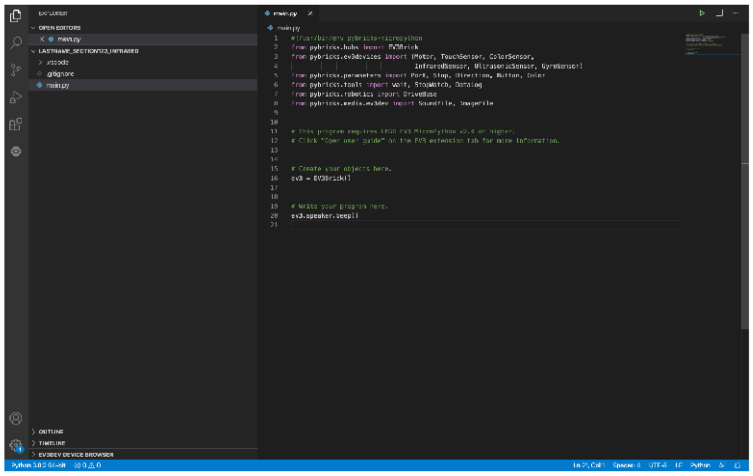
Introduction



- There are 5 constituents of the overall object
- The constituents have varying horizontal measurements that you will need to determine

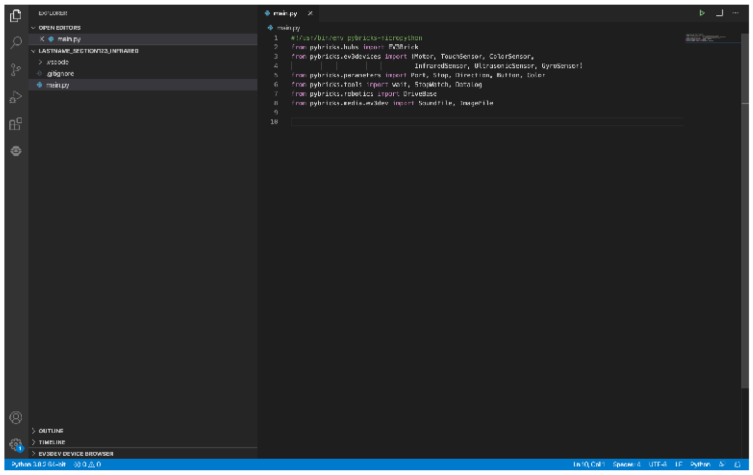
Texas A&M University Engineering | TEES | Texas A&M Engineering Experiment Station

Code Development



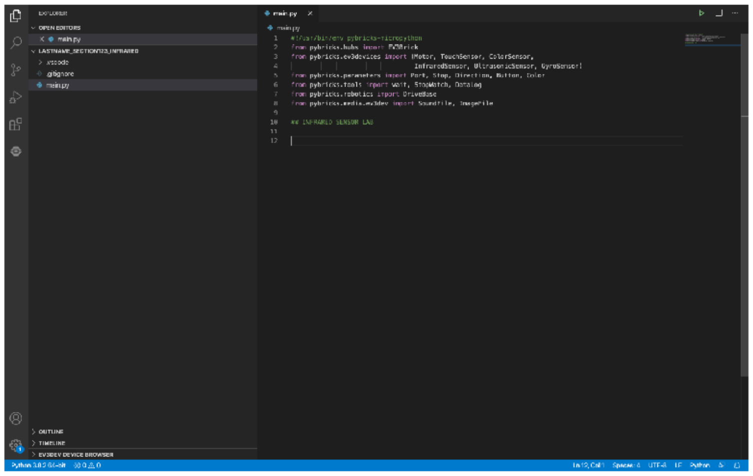
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile


# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.


# Create your objects here.
ev3 = EV3Brick()


# Write your program here.
ev3.speaker.beep()
```

Code Development



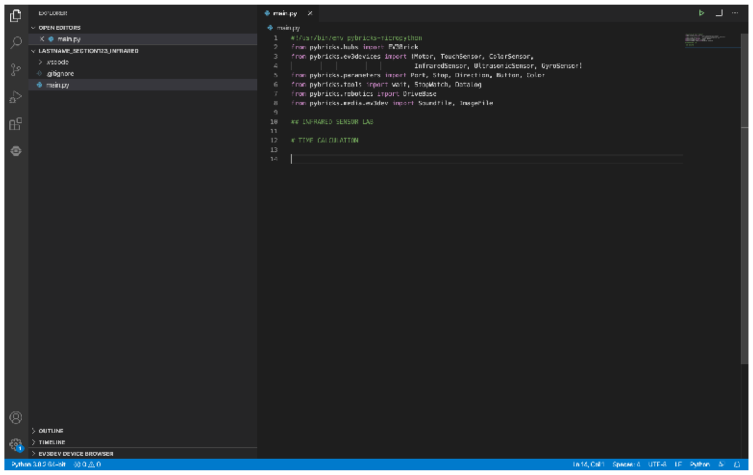
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

```

Code Development



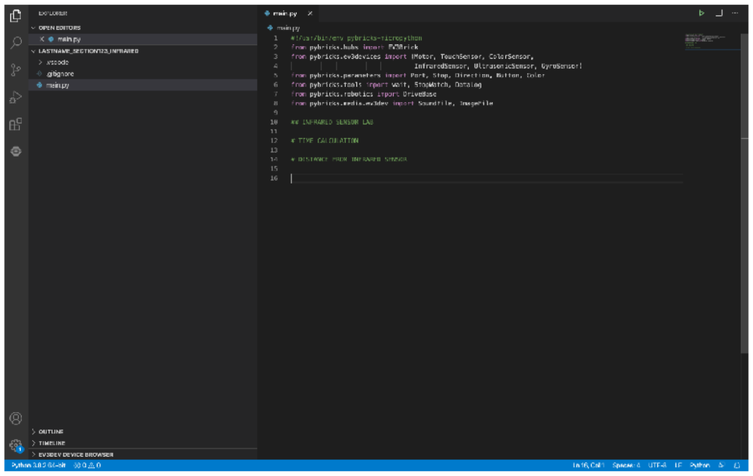
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## INFRARED SENSOR LAB

```

Code Development



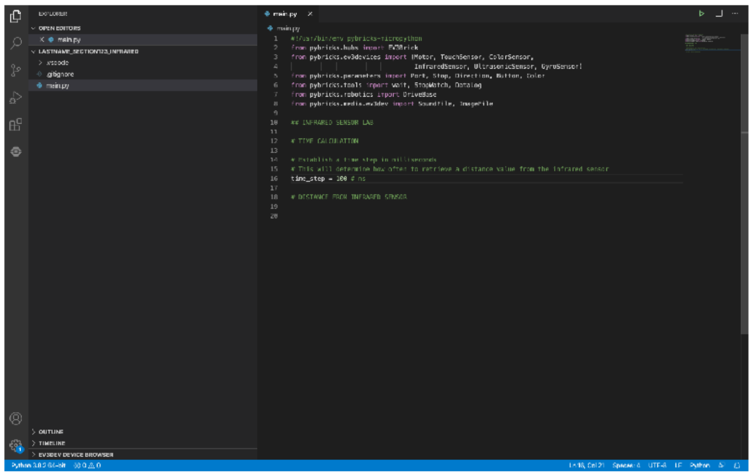
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## INFRARED SENSOR LAB

# TIME CALCULATION

```

Code Development



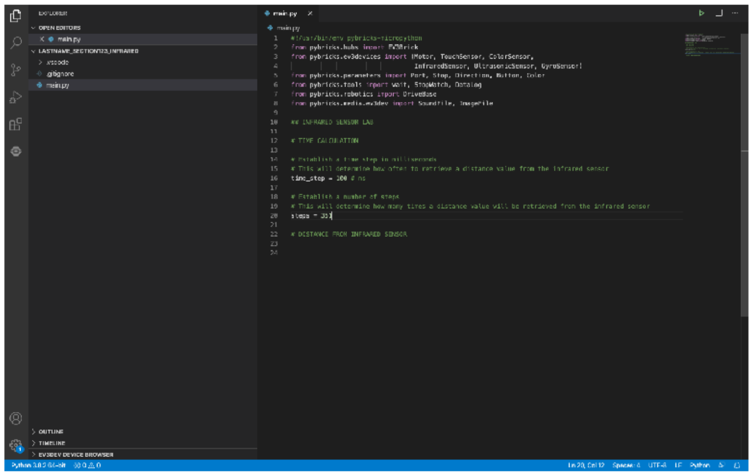
```python
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## INFRARED SENSOR LAB

# TIME CALCULATION

# DISTANCE FROM INFRARED SENSOR

```

Code Development



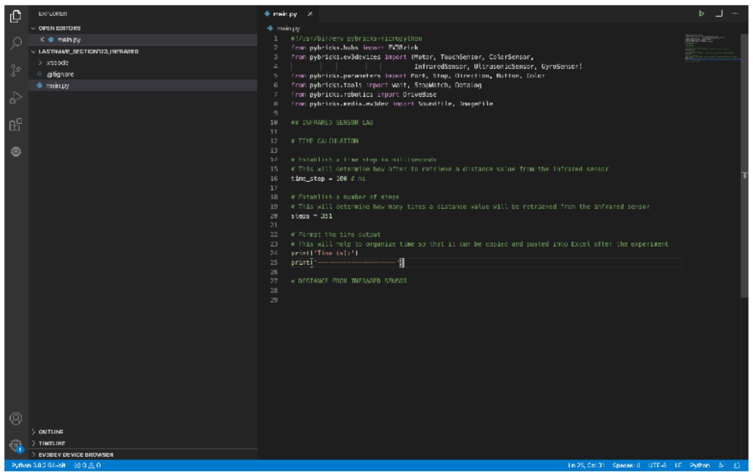
```python
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## INFRARED SENSOR LAB

# TIME CALCULATION

# Establish a time step in milliseconds
# This will determine how often to retrieve a distance value from the infrared sensor
time_step = 100 # ms

# DISTANCE FROM INFRARED SENSOR


```

Code Development



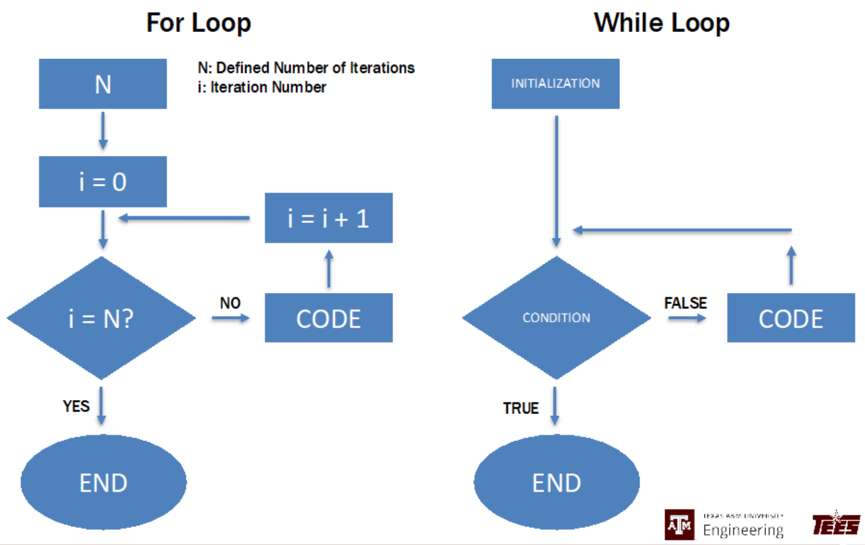
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## INFRARED SENSOR LAB

# TIME CALCULATION

# Establish a time step in milliseconds
# This will determine how often to retrieve a distance value from the infrared sensor
time_step = 100 # ms

# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the infrared sensor
steps = 351

# DISTANCE FROM INFRARED SENSOR
```

Code Development

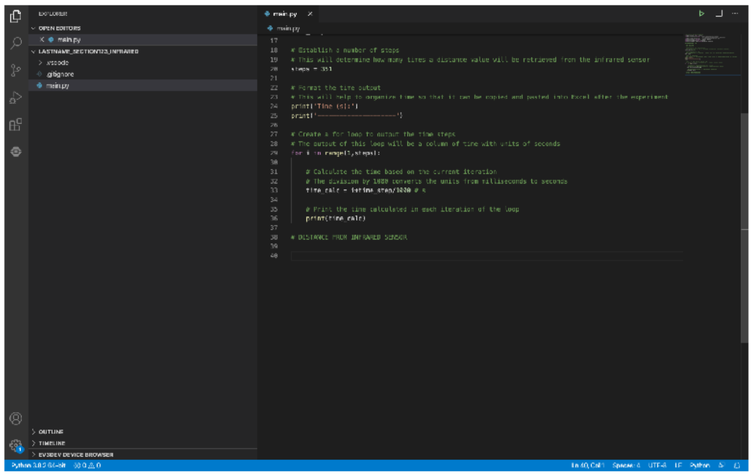


```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

## INFRARED SENSOR LAB

# TIME CALCULATION

# Establish a time step in milliseconds
# This will determine how often to retrieve a distance value from the infrared sensor
time_step = 100 # ms

# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the infrared sensor
steps = 351

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('-------------------')

# DISTANCE FROM INFRARED SENSOR
```

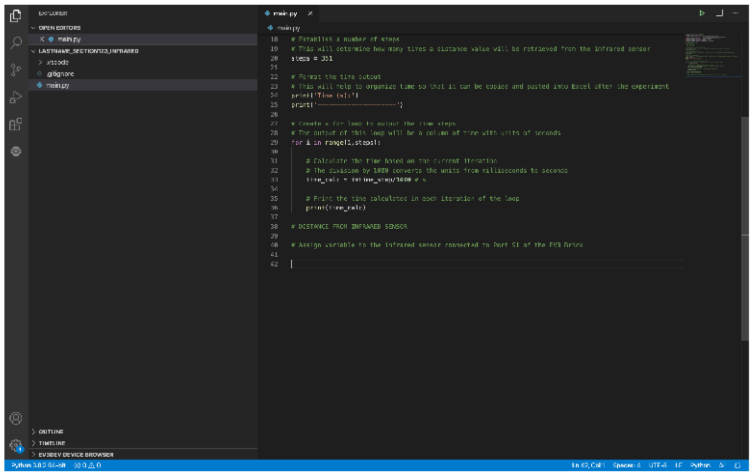
Code Development



Code Development

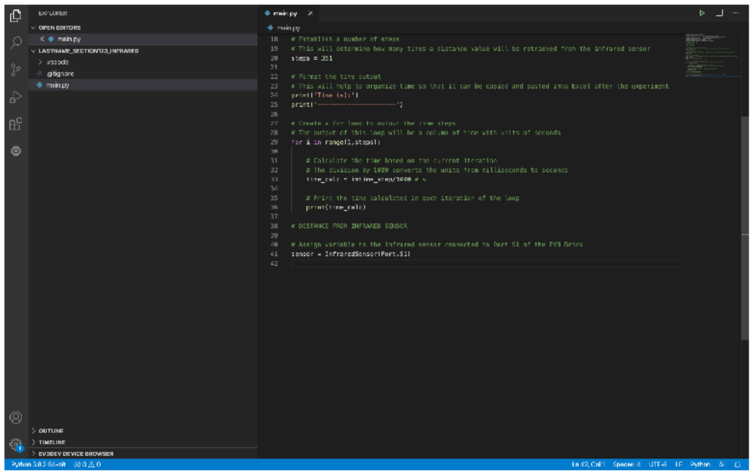

```python
# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the infrared sensor
steps = 351

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('-------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s

    # Print the time calculated in each iteration of the loop
    print(time_calc)

# DISTANCE FROM INFRARED SENSOR

```

Code Development



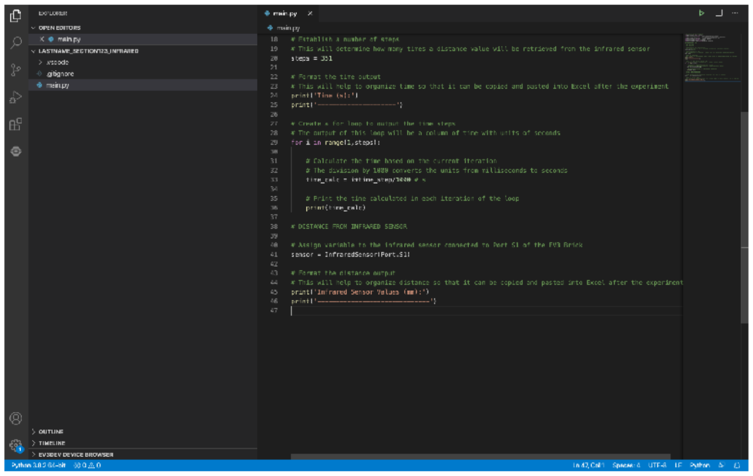
```python
# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the infrared sensor
steps = 351

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('--------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s

    # Print the time calculated in each iteration of the loop
    print(time_calc)

# DISTANCE FROM INFRARED SENSOR

# Assign variable to the infrared sensor connected to Port S1 of the EV3 Brick

```

Code Development



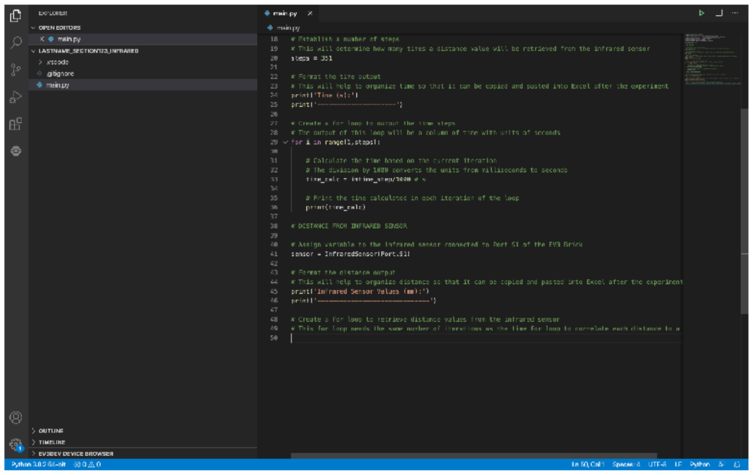
```python
# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the infrared sensor
steps = 351

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('--------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s

    # Print the time calculated in each iteration of the loop
    print(time_calc)

# DISTANCE FROM INFRARED SENSOR

# Assign variable to the infrared sensor connected to Port S1 of the EV3 Brick
sensor = InfraredSensor(Port.S1)

```

Code Development



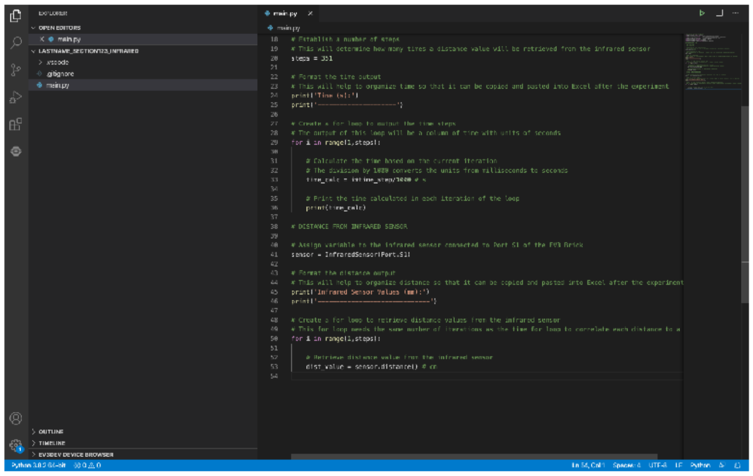
```python
# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the infrared sensor
steps = 351

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('--------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s

    # Print the time calculated in each iteration of the loop
    print(time_calc)

# DISTANCE FROM INFRARED SENSOR

# Assign variable to the infrared sensor connected to Port S1 of the EV3 Brick
sensor = InfraredSensor(Port.S1)

# Format the distance output
# This will help to organize distance so that it can be copied and pasted into Excel after the experiment
print('Infrared Sensor Values (mm):')
print('------------------------------')

```

Code Development



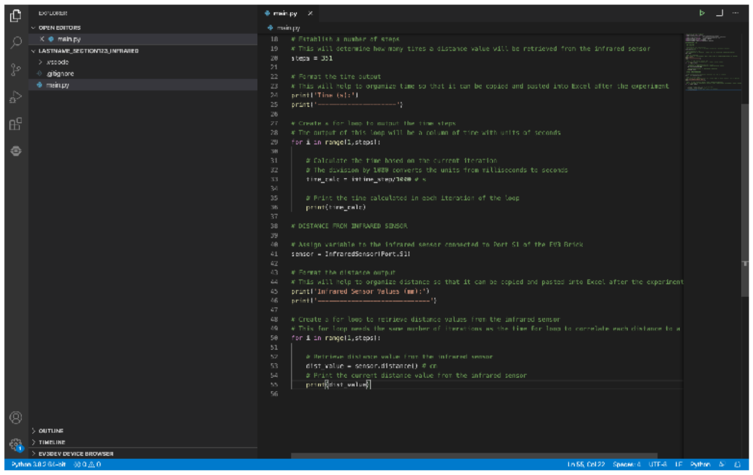
```python
# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the infrared sensor
steps = 351

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('--------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s

    # Print the time calculated in each iteration of the loop
    print(time_calc)

# DISTANCE FROM INFRARED SENSOR

# Assign variable to the infrared sensor connected to Port S1 of the EV3 Brick
sensor = InfraredSensor(Port.S1)

# Format the distance output
# This will help to organize distance so that it can be copied and pasted into Excel after the experiment
print('Infrared Sensor Values (mm):')
print('------------------------------')

# Create a for loop to retrieve distance values from the infrared sensor
# This for loop needs the same number of iterations as the time for loop to correlate each distance to a

```

Code Development



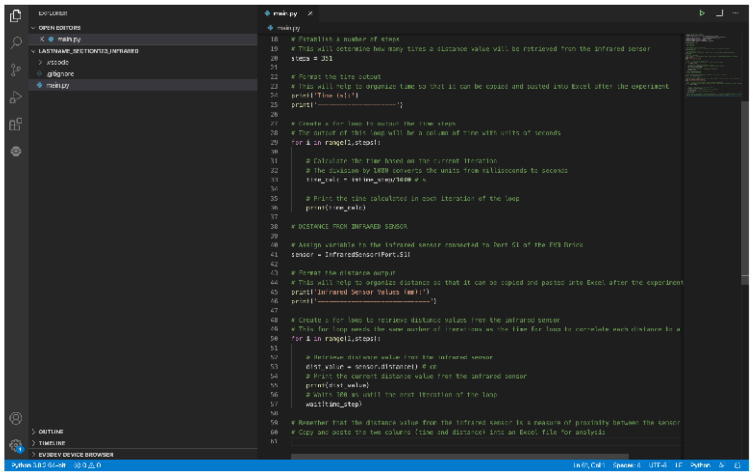
```python
# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the infrared sensor
steps = 351

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('-------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s

    # Print the time calculated in each iteration of the loop
    print(time_calc)

# DISTANCE FROM INFRARED SENSOR

# Assign variable to the infrared sensor connected to Port S1 of the EV3 Brick
sensor = InfraredSensor(Port.S1)

# Format the distance output
# This will help to organize distance so that it can be copied and pasted into Excel after the experiment
print('Infrared Sensor Values (mm):')
print('---------------------------')

# Create a for loop to retrieve distance values from the infrared sensor
# This for loop needs the same number of iterations as the time for loop to correlate each distance to a
for i in range(1,steps):

    # Retrieve distance value from the infrared sensor
    dist_value = sensor.distance() # cm
```

Code Development



```python
# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the infrared sensor
steps = 351

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('-------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s

    # Print the time calculated in each iteration of the loop
    print(time_calc)

# DISTANCE FROM INFRARED SENSOR

# Assign variable to the infrared sensor connected to Port S1 of the EV3 Brick
sensor = InfraredSensor(Port.S1)

# Format the distance output
# This will help to organize distance so that it can be copied and pasted into Excel after the experiment
print('Infrared Sensor Values (mm):')
print('---------------------------')

# Create a for loop to retrieve distance values from the infrared sensor
# This for loop needs the same number of iterations as the time for loop to correlate each distance to a
for i in range(1,steps):

    # Retrieve distance value from the infrared sensor
    dist_value = sensor.distance() # cm
    # Print the current distance value from the infrared sensor
    print(dist_value)
```

Code Development



```python
# Establish a number of steps
# This will determine how many times a distance value will be retrieved from the infrared sensor
steps = 351

# Format the time output
# This will help to organize time so that it can be copied and pasted into Excel after the experiment
print('Time (s):')
print('------------------')

# Create a for loop to output the time steps
# The output of this loop will be a column of time with units of seconds
for i in range(1,steps):

    # Calculate the time based on the current iteration
    # The division by 1000 converts the units from milliseconds to seconds
    time_calc = i*time_step/1000 # s

    # Print the time calculated in each iteration of the loop
    print(time_calc)

# DISTANCE FROM INFRARED SENSOR

# Assign variable to the infrared sensor connected to Port S1 of the EV3 Brick
sensor = InfraredSensor(Port.S1)

# Format the distance output
# This will help to organize distance so that it can be copied and pasted into Excel after the experiment
print('Infrared Sensor Values (mm):')
print('--------------------------')

# Create a for loop to retrieve distance values from the infrared sensor
# This for loop needs the same number of iterations as the time for loop to correlate each distance to a
for i in range(1,steps):

    # Retrieve distance value from the infrared sensor
    dist_value = sensor.distance() # cm
    # Print the current distance value from the infrared sensor
    print(dist_value)
    # Waits 500 ms until the next iteration of the loop
    wait(time_step)

# Remember that the distance value from the infrared sensor is a measure of proximity between the sensor
# Copy and paste the two columns (time and distance) into an Excel file for analysis
```

Texas A&M University Engineering | Texas A&M Engineering Experiment Station

Remote Laboratory for Physical Security – Infrared Sensor Experiment 21

Texas A&M University Engineering

Texas A&M Engineering Experiment Station

Questions?

Contact: anthgalindo@tamu.edu

Infrared Sensor Video Transcript

INTRODUCTION

SLIDE 1
This video will walk through the setup of the code needed to complete the infrared sensor experiment.

SLIDE 2
In this experiment, you will measure a moving object's dimensions using a LEGO MINDSTORMS EV3 infrared sensor. The infrared sensor sends an infrared signal towards the object and detects the reflection of the signal to output a measurement of proximity in units of millimeters. This signal should be viewed as a light because it is measured similarly to that of the light sensor used in the previous experiment.

SLIDE 3
The sensor will be positioned close to the moving object, and the object will pass directly in front of the sensor face. When the sensor outputs a distance value of 0 mm, this is indicative of an object being in front of the sensor. For all values greater than 0 mm, we will assume that an object is not in front of the sensor. We can then assign a "1" to the values equal to 0 mm and a "0" to values greater than 0 mm in an Excel spreadsheet to produce a plot that shows the overall shape of the object. Then, using the velocity you calculated in Part A of the ultrasonic sensor experiment and the time duration of each constituent of the object you can calculate the horizontal dimensions of the overall object.

CODE WALK THROUGH

SLIDE 4
In main.py, the first line sets up the environment that the code will run in. This environment is pybricks-micropython, an optimized version of Python that LEGO MINDSTORMS uses for its EV3 functions. The next seven lines import functions from their respective libraries to be used in the code. These seven lines of imports cover all of our bases for the functions we will use to control the infrared sensor in this experiment.

SLIDE 5
First, let's delete the remaining code in this file since it's not needed for this experiment. We'll start by commenting some sections of the code.

SLIDE 6
Let's start with "## INFRARED SENSOR LAB" to label what this code is for.

SLIDE 7
Next, "# TIME CALCULATION" to focus on the time iterations that will determine how often to retrieve a value from the infrared sensor.

SLIDE 8
Additionally, a "# DISTANCE FROM INFRARED SENSOR" section to focus on retrieving the distance value from the infrared sensor. Commenting your code helps to organize your thoughts and explain the purpose of each line.

SLIDE 9
In the ## TIME CALCULATION section of the code, let's establish a time step in milliseconds. This will determine how often to retrieve a distance value from the infrared sensor. The time step can be adjusted to whatever value you think is appropriate for the experiment. For now, let's set it at 100 milliseconds; we will need to retrieve distance values very often to characterize a moving object's dimensions.

SLIDE 10
Next, we need to establish a number of steps which will be the number of times that data will be collected in the experiment. The number of steps can be changed to however many data points are needed, but for now let's set it at 351 steps. I will explain the reason for 351 steps rather than an even 350 steps soon.

SLIDE 11
Next, we will want to write code that outputs a column of these time steps so that it's easy to copy and paste the values into an Excel spreadsheet. We will do this by printing two lines: one line with a label for Time and its units in seconds, and another line that prints out a dashed line for formatting purposes.

SLIDE 12
Lastly, we need to implement a structure that will calculate and output the time steps. There are for loops and there are while loops. For loops will repeat for a specified number of iterations whereas while loops will repeat a set of actions until a condition is met. Since we have a set number of time steps already defined, we will use a for loop.

SLIDE 13
In the line setting up the for loop, the i is used to represent the iteration passing through the loop. On the first iteration, the first value in the defined range of "1 through steps", (where steps = 351), will pass through the loop. Thus, the time_calc value will yield 1*time_step/1000 which equals 0.1 seconds. This will then be printed in the next line. The next iteration will pass i = 2 through the loop for a time_calc value = 1 second. This will continue until the last iteration. The last iteration will be i = 350 because the range function ignores the very last value in the range. In our case, the range is from 1 through 351, so the loop's last iteration will be for i = 350. This is the reason why the steps variable was assigned 351, to avoid having the loop end on i = 349 if we set steps = 350.

SLIDE 14
The infrared sensor is connected to the EV3 brick through Port S1. Let's go ahead and comment this into the code.

SLIDE 15

To implement this in the code, we have to assign a variable to the infrared sensor that indicates the port being used. We will do this by creating a variable named “sensor” and setting it equal to LEGO’s infrared sensor function. It is called “InfraredSensor()”. Within the “InfraredSensor()” function, we must indicate the port that the sensor is using, so we will put “Port.S1” in the function.

SLIDE 16
Next, we will want to write code that outputs a column of the infrared sensor values so that it’s easy to copy and paste the values into an Excel spreadsheet. We will do this by printing two lines: one line with a label for Infrared Sensor Values in units of millimeters, and another line that prints out a dashed line for formatting purposes. Notice that the units are millimeters for the infrared sensor but they were centimeters for the ultrasonic sensor. This shows that the infrared sensor can make finer measurements than the ultrasonic sensor, however it must operate in close proximity to the moving object. The ultrasonic sensor can operate much further away from a moving object but with not as fine of measurements.

SLIDE 17
Next, we want to retrieve the distance value from the infrared sensor, so let’s start by commenting that into the code. The distance value is a measure of how far away the object is from the infrared sensor in units of millimeters. We need to implement another for loop to retrieve the same number of distance values as there are time steps.

SLIDE 18
The for loop will be very similar to the for loop to calculate the time steps. To get the distance value, we need to create another variable, so let’s call it “dist_value.” To get this value, we need to perform an action on our previously-defined variable, “sensor.” The “sensor” variable is our gateway to the infrared sensor. To retrieve the distance value, we will set the “dist_value” variable equal to “sensor.distance(),” where the distance function acts on the “sensor” variable to retrieve the distance value. The parentheses will remain empty because no further input is needed by the function.

SLIDE 19
It’s also important to print the distance value. In each iteration, the print function will output the distance value for that time and it will appear in the infrared sensor value column we set up. Lastly, the for loop needs to wait for the amount of time equal to the time step before iterating again to retrieve another distance value. Luckily, the wait function will do this for us.

SLIDE 20
Details on running the code are included in the experiment procedures. Please contact a teaching assistant with any questions and concerns.

APPENDIX B

RADIATION DETECTION EXPERIMENT PROCEDURES

Gas-Filled Detectors, Counting Statistics, and Dead Time

Remote Laboratory Procedures

1. Purpose

Geiger-Muller (GM) detectors are some of the most simple detector systems to operate. They are relatively inexpensive compared to other detector system and extremely rugged. In this week's lab, you will use a GM detector to count beta particles emitted from a ^{204}Tl source. Using the counts collected from the detector you will explore counting statistics and determine the GM's dead time.

Lab Prep/Theory: Explain what dead time (τ) is. Through a little research find out and provide the typical dead time for a GM counter. Provide any formulas that will be used in this laboratory and how they will be applied.

2. Materials used for this lab

The instructor will provide the following:

1. a Ludlum 44-9 GM probe pancake detector

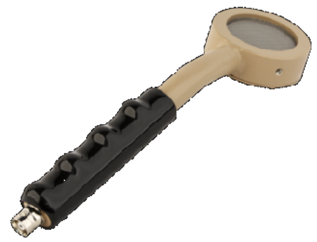


Figure 1. This is called a pancake probe because of its cylinder-like head.

2. a Ludlum 2200 Scaler Rate Meter

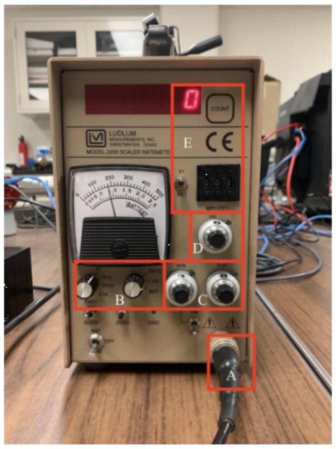


Figure 2. 2200 Scaler ratemeter

3. a ^{137}Cs source (known and unknown)
4. a ^{204}Tl split source

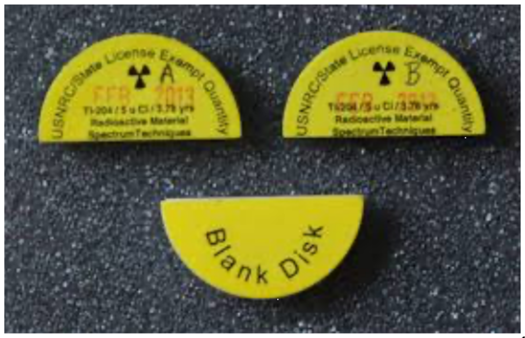

USNRC/State License Exempt Quantity
A
Tl-204 / 5 u Ci / 3.78 yrs
Radioactive Material
Spectrum Techniques
USNRC/State License Exempt Quantity
B
Tl-204 / 5 u Ci / 3.78 yrs
Radioactive Material
Spectrum Techniques
Blank Disk

Figure 3. The "split" source has two hemispheres each containing a 5 ∝Ci ^{204}Tl when it is new. These sources have a relatively short half-life so that is not the activity today. The "Blank disk" issued to help with the placement of half-sources below a detector face. It helps maintain source-detector geometry between source change-outs.

1. Software: Ludlum Model2200/2000 Counter Version 1.2.1

The student should bring the following:

1. Their lab notebook.
2. a copy of these procedures
3. a pen
4. a calculator

The Student **<u>should know</u>** the following:

1. How to calculate activity
2. Count rate statistics (how to calculate and propagate error)
3. How to calculate dead time

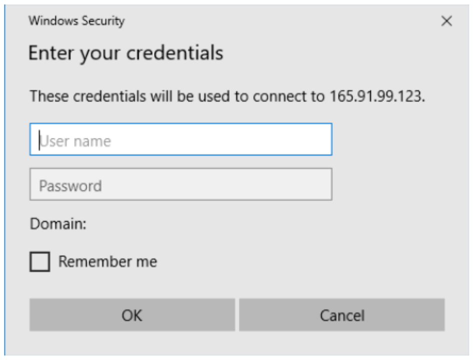
1. Remote Login and Join Zoom Meeting

1. Join the remote lab in Zoom at least 5 minutes before your lab time starts.
2. The instructor/TA will assign a station leader. Lab leaders, remotely log into the remote lab desktop computer
    a. If off campus the station leader needs to sign into TAMU's VPN.
    b. The station leader will use their Remote Desktop Application to login into the NE013 computer.

    Computer IP Address

        Remote Lab: 165.91.99.158

    c. **IMPORTANT:** WHEN LOGGING IN BE CARFUL OF THE DOMAIN YOU ARE IN. THE DEFAULT DOMAIN IS THE DOMAIN YOU ARE IN ON YOUR COMPUTER


Windows Security

Enter your credentials

These credentials will be used to connect to 165.91.99.123.

User name

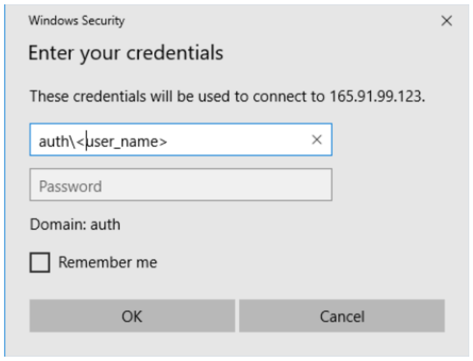
Password

Domain:

Remember me

OK Cancel

- You must login using you TAMU netID. To do this you must be in the “auth” domain. In the username window type “auth\<your_user_name> and then user your netID password. See the picture below:


Windows Security

Enter your credentials

These credentials will be used to connect to 165.91.99.123.

auth\<user_name>

Password

Domain: auth

Remember me

OK Cancel

- Leaders will be the first to operate the system. Leaders can allow their lab partners to operate the desktop through the zoom meeting.
- Leaders, share your screen by clicking the **Share Screen** button in the Zoom menu bar (usually at the bottom of the screen).
- Another participant in the breakout room can click on the **View Options** next to the Zoom’s green bar (usually at the top of the screen), and then select **Request Remote Control**. Hit **Request**.
- The person who is sharing the screen, please allow your lab partner to control your screen by hitting **OK** when Zoom asks you to allow remote control of your screen. **NOTE:** In some cases, you would have to change the Privacy Settings of your computer to allow Zoom to control your computer.

1. Once you are in the system and your partners are sharing your screen, start the lab station’s camera. This can be done by opening the “Camera” application and changing the camera in the upper right hand corner if necessary.

1. Experimental Procedure

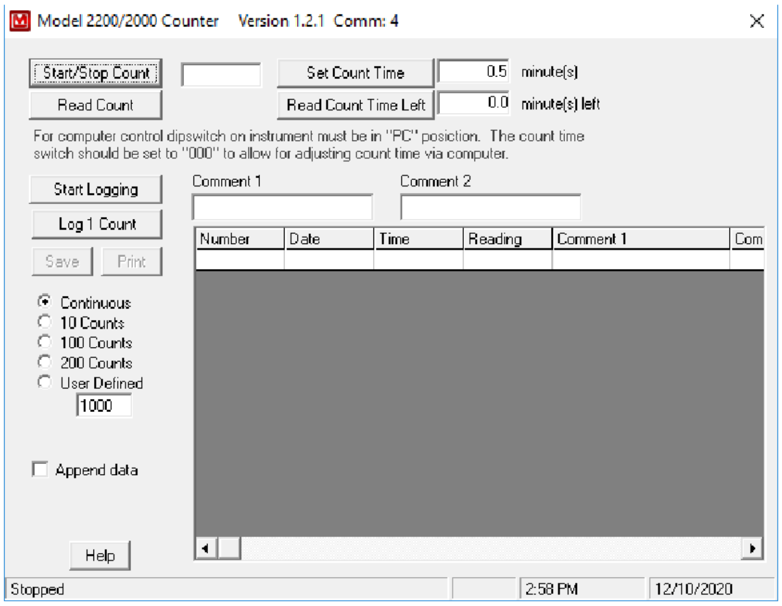
1.1 Starting software and taking background

For an in-person lab you would need to apply voltage and verify detector inputs on the instrument were correctly set. The meter is connected to the 44-9 through a cable.
For your information (These setting will be taken care of by the TA):

- The HV was set at 0.9 KV. This would be indicated by the gauge above ‘B’ in Figure 2.

- The LLD (THRESHOLD) is set to 50/1000 (right knob in C). This means the meter will only register counts when pulses coming from the probe are above 50 mV in height.
- The Window switch (below the left knob in C) is set to "off". This allows for all pulses coming from the GM, above the threshold, is counted.
- The rate meter connects to the lab station desktop computer through USB. The output from the meter is through a RS232 port that is connected to the computer using a RS232/USB converter cable.
- The counter must be specially set to communicate to the computer. These settings can be found in the 2200's manual

1. Start the Ludlum 2200/2000 Counter software (Figure 4)
2. Set the count time for 2 minutes
   - Notice that the pancake GM probe is secured to the source holder/counting stand
   1. Download and open the Dead Time LabVIEW VI to the remote lab desktop computer. You should have received this from the professor/TA. **<u>DO NOT EDIT ANY OF THE CODE. If you edit the code, please return to an original version of the program.</u>**
3. In the LabVIEW VI, click on the right-pointing arrow. This will run the program.
4. The rotary table and the linear slide each have a half of a ^{204}Tl source on their respective source platforms. The rotary table also has unknown and known ^{137}Cs sources on separate platforms. Set the rotary table to "Hide Half Source" and set the linear slide to "Hide Half Source" so that there are not any sources under the GM probe.
   a. In the Dead Time VI menu you will see 2 control sets: one for the **Rotary Table** and one for the **Linear Actuator**". Both of these will control the movement of the specified laboratory apparatus. Click "Start Move" to move a motor (Figure 6). Even if not looking at the camera feed, you will know if a motor is in operation if the "In Motion" light is red.
5. Press the *Start/Stop Count* button in the Ludlum software.
6. Record your 2 minute background. Calculate the background count rate and its uncertainty.



Set Count Time: 0.5 minute(s)

Read Count Time Left: 0.0 minute(s) left

For computer control dipswitch on instrument must be in "PC" posiction. The count time switch should be set to "000" to allow for adjusting count time via computer.

(●) Continuous
( ) 10 Counts
( ) 100 Counts
( ) 200 Counts
( ) User Defined 1000

Stopped | 2:58 PM | 12/10/2020

Figure 4. GUI for the Ludlum software.

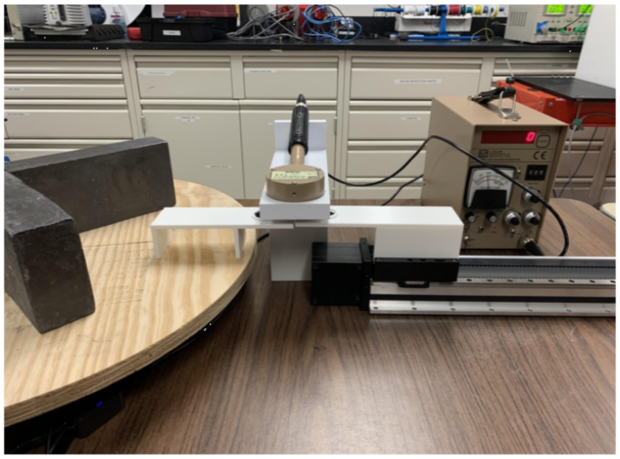


Figure 5. This picture shows how the GM probe should be placed onto the stand.

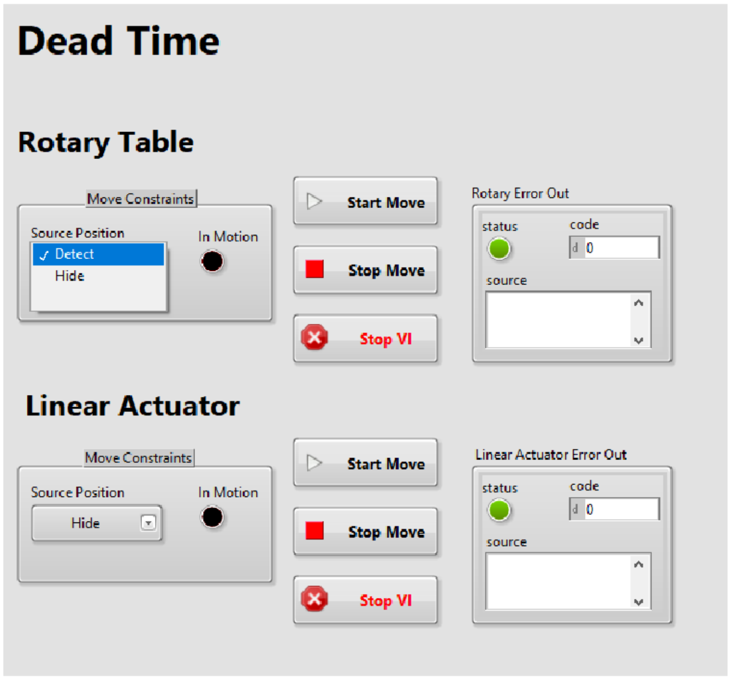


Figure 6. This picture shows the LabVIEW VI used to control the rotary table and linear slide motors.

1.1 Counting Statistics

1. After your background count, place a known ^{137}Cs source under the GM probe by selecting "Known Gamma Source" for the rotary table. Select "Hide Half Source" for the linear slide.
    a. Ask the TA if there is a calibration date and activity for the source. Assume 5% uncertainty in this value.
2. Take 40 sequential 6-second counts and record these raw counts N.
    a. Set your count time to 0.1 minutes
    b. On the left side of the Ludlum GUI, click on the *User Defined* button.
    c. Enter "40" in the window below the button

   a. Click on *Start Logging*
2. You can do the following calculations during lab, but you will eventually be required to do them as a Worksheet and/or Results and Discussion
   a. Calculate the mean of these 40 counts N_{avg}. Tabulate N-N_{avg}. Don't forget to take background into account. Note the number (N-N_{avg}) is called the residual and can be positive or negative. If you add up all of the (N-N_{avg}) values in the table, the answer should be zero. If it is not, a mistake has been made. Calculate the standard deviation (σ) using Eq 1. Sixty-eight percent of the observed data should lay within the range $N_{avg}+\sigma$ to $N_{avg}-\sigma$. Does this appear to be true?

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \bar{x}_e\right)^2} \qquad \text{(Eq. 1)}$$

1.1 Detector Dead Time Measurements

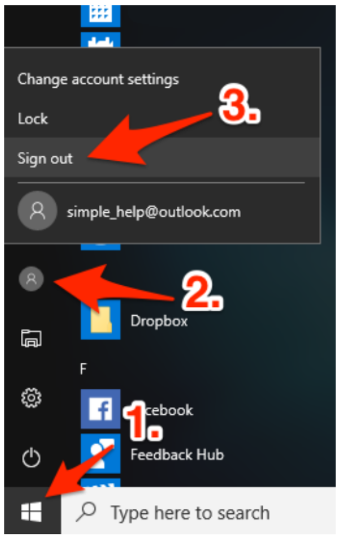
1. Select "Hide Half Source" for the linear slide and "Detect Half Source" for the rotary table so that only the rotary table's half source is under the GM probe.
   a. Ask the TA for the calibration date and activity for the source. Assume 5% uncertainty in this value.
   b. Ask the TA for the source to detector distance.
2. Count the first half of the source for 1 minute. Record the count time, counts, and count rate (s^{-1}). Be sure to propagate error. The count rate here will be m_1 in your deadtime calculation.
3. Select "Hide Half Source" for the rotary table and select "Detect Half Source" for the linear slide. Count the second half of the source for 1 minute (or the same time length used in step 2). Record the count time, counts, and count rate. Be sure to propagate error. This count rate is m_2.
4. Now select "Detect Half Source" for the rotary table so that both half sources are under the GM probe. Count both halves of the source at the same time for 1 minute. on the same shelf. Calculate the count rate and call this M_{12}. Record the count time, counts, and count rate. Be sure to propagate error.
5. Remove both sources by selecting "Hide Half Source" on both the rotary table and linear slide and count background for 1 minute. Calculate the background count rate and call this M_b. Record the count time, counts, and count rate. Be sure to propagate error.
6. From these values, calculate the dead time using Eq. 2. (assume non-paralyzable). Be sure to propagate error. Can you neglect background? Talk to the TA or instructor after you have calculated the dead time.

$$\tau = \frac{m_1 + m_2 - m_{12}}{2m_1m_2} \qquad \text{(Eq. 2)}$$

1.2 Detector Efficiency Measurements and Activity Measurements for "Unknown"

1. Place the known ^{137}Cs source under the GM probe by selecting "Known Gamma Source" from the LabVIEW VI drop-down menu. Count the source for 300 s. Record counts and count rate. Propagate error.
   a. Be sure to decay correct the activity to the month/year of the experiment.
2. Place the unknown ^{137}Cs source under the GM probe by selecting "Unknown Gamma Source". Count the source for 300 s. Record counts and count rate. Propagate error.
3. Calculate the activity of your unknown source in two ways:
   a. Taking into account dead time losses
   b. Not considering dead time losses
   c. Tell the TA the activity you calculated. They will then tell you the actual activity. Assume +/- 5% uncertainty in this activity. Be sure to decay correct this activity too.

1. Compare the difference in the activity results through two approaches.
    a. Is there a **<u>significant</u>** difference in the answers using the 2 approaches?
2. Compare your calculated activities to the actual (decay corrected) activity of the "unknown".
    a. Are your activities statistically the same?? If not, why???.
3. Have the TA ramp down the voltage of the ratemeter
4. Return the rotary table and linear slide back to "Detect Half Source" for both.
5. Close out of all programs.
6. If you are dismissed, lab leader please officially sign out of the remote desktop. Don't just close the remote window.
    a. This is accomplished by clicking on the start menu. See Figure 6 below.



This week you will have 2 separate assignments: Worksheet Lab 2 and Results and Discussion for Lab 2.

R&D Grading Cycle <u>(these are due the day of your lab!!)</u>
Draft Due (go to reviewers): Week 2/15-19
Edited Drafts due(returned to authors): Week of 2/22-26
Finals R&D due: Week of 3/1-5

1. **Worksheet for Lab 2** on Counting Statistics **(Due next Week!!)**.
    a. Create your own worksheet
        i. Create a table and put all 40 data entries in the table. Table headings should be: Reading Number, Count, Count Uncertainty, Count Rate (s^{-1}), Count Rate Uncertainty (s^{-1}), $N\text{-}N_{avg}$
    b. What is the mean?
    c. What is the standard deviation (σ)?
    d. What is the sum of the residuals?
    e. Does 68% of the values fall between ±1σ?
    f. Make another table of $(N\text{-}N_{avg})/\sigma$ to the nearest 0.5. For example, if $(N\text{-}N_{avg})/\sigma = +1.11$, then the rounded off value would be +1.0.
    g. Produce a histogram on your worksheet of the rounded off events

1. **Results and Discussion.**
   a. Your Results and Discussion will focus on 2 aspects of your lab
      i. Dead time results.
         1. Be sure to make a table of the data
         2. Show the formula you used to calculate the dead time
         3. Be sure to include uncertainty in the value.
         4. Be sure to compare your dead time to an "accepted" GM dead time. Be sure to cite the source you used.
   b. The unknown activity determination.
      i. Include the formula you used
      ii. Propagate uncertainty

Exploring the r^{-2} Rule

Remote Laboratory Procedures

1. Purpose

The purpose in this laboratory is to give students some experience with scintillation detectors and to explore the "$1/r^2$" rule. This rule is a fundamental behavior that we have all been taught in our most rudimentary radiation classes. To explore this rule, students will use a NaI detector, gamma spectroscopy software, and a linear slide to move a source away from the detector.

Lab Prep: Give a diagram and in words describe what the "$1/r^2$" rule is. Give an equation to calculate the expected photon intensity at a given distance.

Hint: http://hyperphysics.phy-astr.gsu.edu/hbase/Forces/isq.html#c4

2. Materials Needed

The instructor will provide the following:

1. a Canberra NaI detector
2. a Canberra Osprey MCA
3. access to GENIE-2000 Gamma Acquisition & Analysis
4. a linear slide
5. a ^{137}Cs source

The student should bring the following:

1. Their lab notebook
2. a copy of these procedures
3. a pen
4. a calculator

The student should know the following:

1. How to calculate activity and convert between becquerel and curie

3. Experimental Procedure

3.1 Remote Desktop Connection

1. VPN into the TAMU network using the Cisco AnyConnect Secure Mobility Client. Use these instructions if you have not done this before:
   https://agrilifelaserfiche.tamu.edu/documents/tamu-connect-vpn.pdf/
   https://it.tamu.edu/services/network-and-internet-access/virtual-private-networks/virtual-private-network-vpn/
2. Open the Remote Desktop client on your computer. For MAC users, use "Microsoft Remote Desktop" in the app store. For Windows users, you should already have the remote desktop program installed on your computer.
3. Enter the following IP address: 165.91.99.158
4. Sign in with your NET ID login and be prepared to use Duo.

1. Open the "camera" application and set the camera to the Logitech webcam in the upper right-hand corner. You should see the NaI detector and the linear slide as shown in Figure 1.
2. Download and open the Inverse Square Law LabVIEW program given to you by the professor/TA. **DO NOT EDIT ANY OF THE CODE.**
3. Launch the program by selecting the arrow in the upper left-hand corner of the LabVIEW window.
4. Organize these windows to ensure you can see everything on your screen.

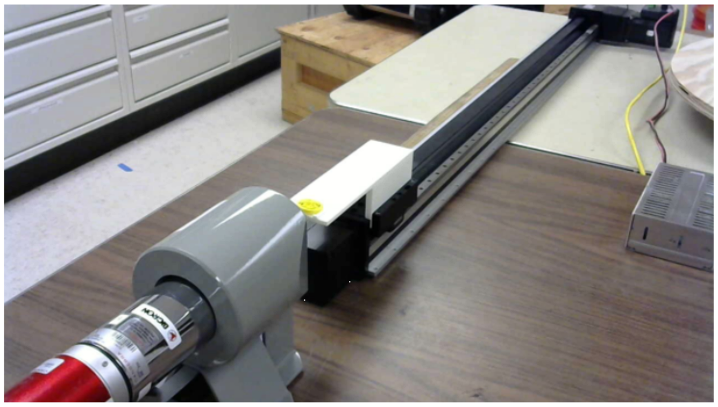


Figure 1. Camera view of the experiment apparatus.

1.1 Energy Calibration for NaI Detector

1. Notice that the NaI detector is placed in a lead collimator pointed towards the linear slide. The pre-amp tube base is connected to a Canberra OSPREY multi-channel analyzer.
2. Open Genie Gamma Acquisition and Analysis.
3. Click on File ➔ "Open datasource".
4. Select Detector from "Source" (About half way down in the Open File Window).
5. Select "NAI_TEST".
6. Adjust Setting
   a. Go to **MCA>Adjust>HVPS** and set the HV to 850 V. Click the "OK" button right next to the HV value. Then turn **ON** the HV. (Wait for the HV to fully ramp up before using the detector).
   b. Assure Filter settings are correct. Select the **Filter** button. Set rise time to 0.800 ∝s and flat top 0.2.
   c. Go to Gain. Hit the **NEXT** button in the "Adjust" window and make sure the LLD is at 1.5 % to eliminate low energy noise.
   d. Go to (**MCA>Acquire Set-up…**). Adjust your acquisition time (Time Preset) to 200 seconds.
7. Make sure the ^{137}Cs source is set at the "10 cm" position using the LabVIEW VI. If the VI outputs an error, do not worry; it simply means that the source is already in that position. Note that this source has a tabulated activity of 10 µCi and a calibration date of August 2010.
8. You are "setting the scale" of the spectrum. This is the maximum energy displayed in the spectrum.
   a. **Start** an acquisition (Under Acquire to the left of the spectrum).

Before adjusting the Gain settings, calculate at what channel the 662-keV photopeak from ^{137}Cs should be in a 1.5 MeV spectrum. The channel limit on your

a. spectrum is 2048. **Hint**: If channel 2048 corresponds to 1.5 MeV, assuming linear relationship between channel number and energy, what channel would correspond to 662 keV?

$$\frac{1.5MeV}{0.662MeV} = \frac{2048}{x}$$

b. Place your cursor in this channel.
c. Go back to the **MCA>Adjust** window and select the **Gain**.
d. Adjust the Coarse and Fine Gain settings so that the peak coincides with the cursor (the channel that you just calculated ± 20 channels). **This may require you hitting the "Clear" button occasionally to clear the spectrum. There is no need for you to hit Stop… then Start… repeatedly**

2. Once the ^{137}Cs peak is where you want it, accumulate a spectrum for 200 s
   a. Once the acquisition is complete, make sure that your cursor is at the apex of the peak. Go to **Calibrate – Energy Only Calibration.** Click on the Cursor button (The channel # should appear in the Channel box. Clicking this button makes Genie put the channel number where your cursor currently is).
   b. Enter 662 in the **Energy** box and click **Accept.**
   c. Click **OK.**
3. Record peak features
   a. In your "INFO" box below the spectrum, click Next until you get to MARKER INFO (Figure 1).
   b. Put your "Markers" on either side of the Peak.
      i. Put the cursor on the left side of the peak. Press **Ctrl L.**
      ii. Put your Cursor on the right side of the peak. Press **Ctrl R.**
      iii. Record what channels the markers are on.
      iv. Record the channel number for the left and right markers, centroid, FWHM and Area.

1.1 Record the Peak Area (Counts) for each Distance Measurement

1. In your notebook, prepare a table that will allow you to record distance, counts, count uncertainty, count rate, and count rate uncertainty. Refer to the LabVIEW VI for the number of rows you will need in the table.
2. With the source positioned on the linear slide 10 cm away from the detector, start an acquisition for 30 s (adjust acquisition time like in 3.2.6.d).
3. Record peak features
   a. In your "INFO" box below the spectrum, click Next until you get to MARKER INFO (Figure 1).
   b. Put your "Markers" on either side of the Peak.
      i. Put the cursor on the left side of the peak. Press **Ctrl L.**
      ii. Put your Cursor on the right side of the peak. Press **Ctrl R.**
      iii. Record what channels the markers are on.
      iv. Record the channel number for the left and right markers, centroid, FWHM and Area.
4. Repeat steps 3.3.1-3.3.3 for the remaining distances in the LabVIEW VI.
5. Calculate the count uncertainty, count rate, and count rate uncertainty using the following equations:

$$\sigma_{Count} = \sqrt{Counts}$$

$$\text{Count Rate} = \frac{\text{Counts}}{\text{Acquisition Time}}$$

$$\sigma_{\text{Count Rate}} = \frac{\sigma_{\text{Count}}}{\text{Acquisition Time}}$$

Post-Lab Assignment

Make a table of your data and plot the count rate curves in Excel. Analyze and discuss the shape of the curve, and compare it to the expected r^{-2} rule curve.

Alpha Attenuation

Remote Laboratory Procedures

1. Purpose

This experiment is designed to show the student how to use Passivated Implanted Planar Silicon (PIPS) detectors to study the properties of alpha-emitting isotopes. The student will use these detectors to identify and quantify the nuclear material in a small sample and determine the efficiency of the detector as it relates to the energy of the emitted alphas. The student will gain an understanding of the stopping power of a material by calculating that value for Mylar film.

Lab Prep: In this lab you will use the peak ratio method to determine the activity of the unknown sample. Describe the measurements and data you will need to take in order to determine the unknown activity. In your discussion, include the appropriate formula and how it will be used.

2. Materials Needed

The instructor will provide the following:

1. an Alpha Analyst alpha spectroscopy system
2. an MCA
3. assorted alpha sources

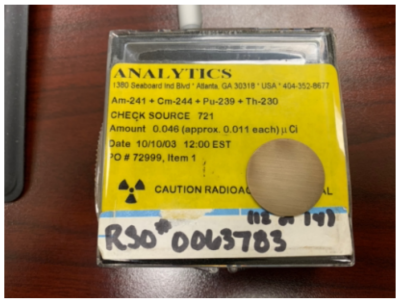

ANALYTICS
1380 Seaboard Ind Blvd * Atlanta, GA 30318 * USA * 404-352-8677
Am-241 + Cm-244 + Pu-239 + Th-230
CHECK SOURCE 721
Amount 0.046 (approx. 0.011 each) µCi
Date 10/10/03 12:00 EST
PO # 72999, Item 1
CAUTION RADIOA
AL
RSO# 0063783

Figure 1. This is a mixed alpha source to be placed in the alpha spectrometer.

4. Mylar film

Figure 2. This is a roll of Mylar film with a portion of it pulled out.

The student should bring the following:

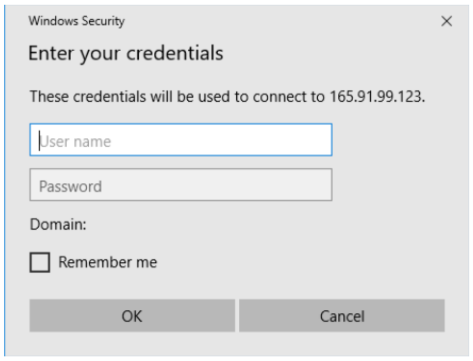
1. lab notebook.
2. a copy of these procedures
3. a pen
4. a straight edge for drawing lines
5. a calculator

1. Remote Login and Join Zoom Meeting

1. Join the remote lab in Zoom at least 5 minutes before your lab time starts.
2. The instructor/TA will assign a station leader. Lab leaders, remotely log into the remote lab desktop computer
   a. If off campus the station leader needs to sign into TAMU's VPN.
   b. The station leader will use their Remote Desktop Application to login into the NE013 computer.

   Computer IP Address
      Remote Lab: 165.91.99.158

   c. **IMPORTANT:** WHEN LOGGING IN BE CARFUL OF THE DOMAIN YOU ARE IN. THE DEFAULT DOMAIN IS THE DOMAIN YOU ARE IN ON YOUR COMPUTER


Windows Security

Enter your credentials

These credentials will be used to connect to 165.91.99.123.

User name

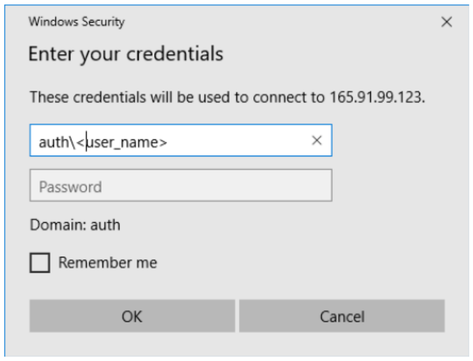
Password

Domain:

Remember me

OK Cancel

- You must login using you TAMU netID. To do this you must be in the "auth" domain. In the username window type "auth\<your_user_name> and then user your netID password. See the picture below:


Windows Security

Enter your credentials

These credentials will be used to connect to 165.91.99.123.

auth\<user_name>

Password

Domain: auth

Remember me

OK Cancel

- Leaders will be the first to operate the system. Leaders can allow their lab partners to operate the desktop through the zoom meeting.
- Leaders, share your screen by clicking the **Share Screen** button in the Zoom menu bar (usually at the bottom of the screen).

- Another participant in the breakout room can click on the **View Options** next to the Zoom's green bar (usually at the top of the screen), and then select **Request Remote Control**. Hit **Request**.
- The person who is sharing the screen, please allow your lab partner to control your screen by hitting **OK** when Zoom asks you to allow remote control of your screen. **NOTE:** In some cases, you would have to change the Privacy Settings of your computer to allow Zoom to control your computer.

1. Once you are in the system and your partners are sharing your screen, start the lab station's camera. This can be done by opening the "Camera" application and changing the camera in the upper right hand corner if necessary.

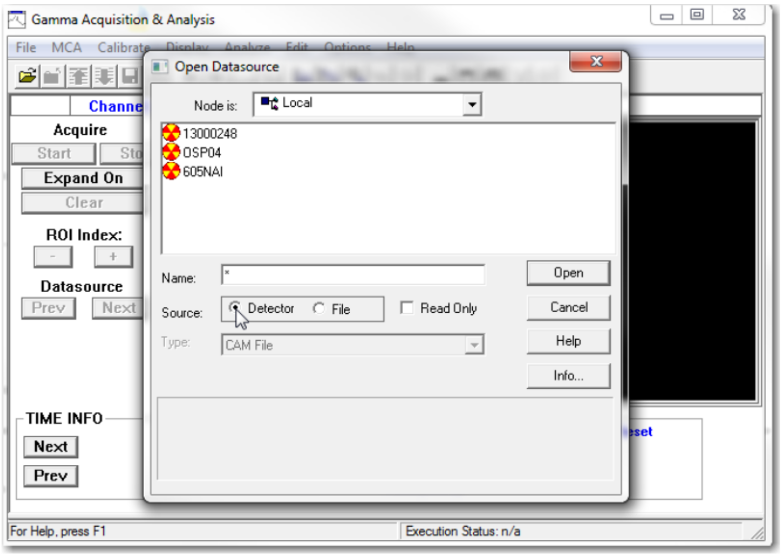
1. Experimental Procedure

Table 1. A list of the alpha sources, the energies they emit, and the yield of each energy.

Isotope	Alpha Energy (MeV)	Yield (%)
^{230}Th	4.621	23.4
	4.687	76.3
^{239}Pu	5.106	11.9
	5.144	17.1
	5.157	70.8
^{241}Am	5.443	13.1
	5.486	84.8
^{244}Cm	5.763	23.1
	5.805	76.9

1.1 Determination of the Detector Efficiency as a Function of Energy

1. Open the *Genie Alpha Acquisition and Analysis* software on the laboratory computer. Go to **File – Open Datasource**. A window will open. Select the **Detector** button. There should be six detector files named "A_1_1A", "A_1_1B", "A_1_2A", "A_1_2B", "A_1_3A", and "A_1_3B". These represent the six PIPS detectors that you will operate for this lab. To begin, double click on "A_1_1A" to open that detector (Figure 3).
   a. If when attempting to open the file GENIE returns an error with the message "Hardware Unavailable," then the Canberra SNAP driver needs to be restarted before the detector can be operated. Contact the professor/TA for assistance.



Node is: Local

13000248
OSP04
605NAI

Name: *

Source: [✓ Detector] File

Type: CAM File

Figure 3. Open Datasource window. The **Detector** button is circled.

1. Now go to **MCA** – **Adjust**. A small window will open near the bottom of the screen.
   a. Select **Gain**. Verify that all the gains are set at their minimum.
   b. Select the **Filter** button. Make sure the rise time is 0.5 ∝s and the flat top is 0.0 ∝s.
2. This first detector chamber will have a mixed alpha source already placed as close to the detector as possible.
3. Evacuate the chamber by first going to **MCA – Adjust**
   a. Select **Vacuum** tab.
   b. Set vacuum status to **Pump**. Wait 5 seconds.
4. Set the acquisition time to 180 s. Go to **MCA – Acquire Setup** and set the time (Figure 4). Select Live Time and enter 180.
5. The bias voltage will need to be set to +40V. Go to the **Voltage** tab. The current voltage setting should show **0** V. Slowly drag the bar until GENIE shows a voltage of **40 V**. This system has no on/off setting for bias voltage, so the voltage that the bar shows is the voltage the system will attempt to apply to the detector.
   a. ***Note***: *whenever you pump down or let the chamber up to air, it is imperative that the bias supply be turned off (set to 0 V). It should be turned on only when the system is under vacuum. If these procedures are not followed, the detector and the preamp can be damaged.*
6. Click the Start button under Acquire to begin an acquisition.
   a. ***Note***: *It may take a minute or more after clicking the Start button before GENIE begins displaying data. GENIE and the Alpha Analyst have internal checks to prevent the system from collecting data when it is in a state that may potentially damage the detector (i.e. not enough vacuum or incorrect voltage). If you do not see data for up to five minutes after clicking Start, check to make sure your settings are correct and try again, and if the problem persists contact the professor/TA.*

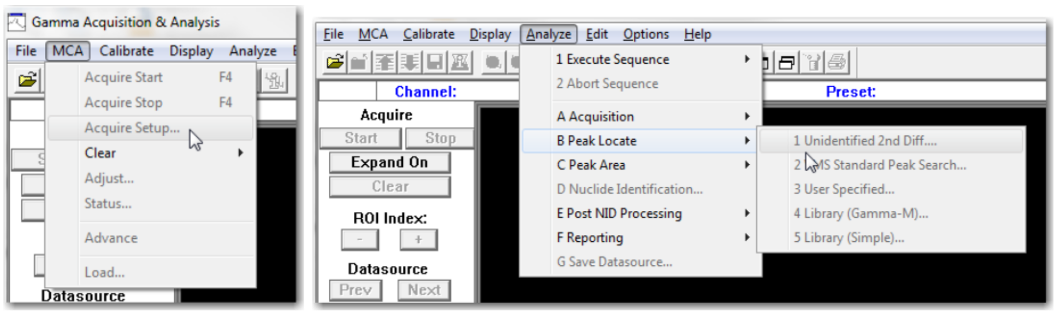


Figure 4. Acquire setup and Peak Locate functions.

1. Using the GENIE software, locate and record the centroid energy and area of each peak. From the menu bar select **Analyze** - **Peak Locate** – **Unidentified 2nd Diff**. (Figure 4) A window will pop up. Select **Generate Report** and then select **Execute.**
2. To review the report select the icon on the menu bar that looks like a white box with a thick black line on top.
    a. You can minimize the report window by selecting the icon that looks like a thick black line.
3. For each peak of interest record the channel number with the highest count.
4. Using the GENIE software, locate the peaks in the spectrum and manually determine the peak areas, centroid energies and FWHMs.
    a. On the upper left-hand part of the screen click on the **Expand On** button. The spectrum will be split in two and you will notice a small box outlined in white in the main window. Whatever is in this box will be magnified in the upper window. The box can be moved using the mouse. Put the mouse in the middle of the box and move it to the first peak.
    b. With the mouse, click on the left side of this peak where you think the peak begins. On the keyboard press **Ctrl L**. This will move a marker to the left side of the peak at the cursor location.
    c. Now move the cursor and click on the right side of the peak. Now press **Ctrl R**. Similarly, a marker is now at the right side of the peak.
    d. Below the spectrum is an information window. Hit the **Next** button until you find **Marker Info**. You can find all of the information about the Region of Interest you have just created. **Record the Marker information too.** You may have to use it again.
5. Calculate the current activity for each source using the date of this experiment and the calibration date of the source.
6. Using the source activities determine the number of alpha particles emitted per second from each source. Since the peaks are overlap in your spectra it may be easiest to assume 100% alpha yield for each source. Table 1 above may be helpful in doing this. This value is sometimes called the source strength.
7. Using the source strength and the peak areas, calculate the detector efficiency at each alpha particle energy. **<u>Discuss your results with your instructor or TA.</u>**

1.1 Determination of the Stopping Power of Mylar Film

1. Use the results from the energy calibration as your zero-thickness data.
2. **Turn off the bias**, in **MCA** – **Adjust** – **Vacuum** select **Vent** (?) to return the chamber to normal pressure, and then close the detector file. Open the detector file for the next chamber: “A_1_1B.”

1. This chamber has the same energy calibration as what you just created in the first chamber, and it has the same alpha source as the first chamber with one layer of Mylar placed between it and the detector.
2. In **MCA – Adjust – Vacuum** select **Pump** and then go to **MCA – Adjust – Voltage** and set the bias to + 40 V.
3. Accumulate counts for at least 300 s. Does the spectrum look as you expected?
4. Using the GENIE software, manually locate the 'peaks' in the spectrum and determine the energy at the 'peak' (energy at the channel with most counts). Record this value.
5. Repeat steps 2-6 for the next two chambers, "A_1_2A" and "A_1_2B", which have the same source and energy calibrations as the previous detectors with two and three layers of Mylar, respectively.
   a. You may need to extend the acquisition time for detector "A_1_2B" to 600 seconds or longer until the peaks are easily distinguishable from each other.
6. If each layer of film is 10 μm thick, plot the Mylar Thickness vs. Alpha Energy.
   a. What is the relationship between the two?
   b. **The slope of the line has units of MeV cm^{-1} and is known as the stopping power of the material.** What is your calculated stopping power? You will need to propagate error of the slope using the excel LINEST function.

1.1 Determination of the Activity of an Unknown Alpha Source by the Ratio Method

1. **Turn off the bias**, in **MCA – Adjust – Vacuum** select **Vent** (?) to return the chamber to normal pressure, and then close the detector file. Ask the TA for the source number and record it for later identification. Open the detector file corresponding to the next chamber: "A_1_3A"
2. This chamber has the same energy calibration as what you created in the first chamber and used in the previous three chambers.
3. In **MCA – Adjust – Vacuum** select **Pump** and then go to **MCA – Adjust – Voltage** and set the bias to + 40 V.
4. Accumulate counts for at least 180 s or the same amount of time used in 4.1.
5. The software can automatically find the peak area(s). On the menu bar go to **Analyze - Peak Locate – VMS Standard Peak Search**. Make sure **Generate Report** is marked. Select **Execute**.
6. Using whatever resources are available to you, identify the constituent(s) of the unknown from the reported peak energies. Given the resolution of these detectors, you should be able to identify the isotopes in the unknown sample.
7. Using the peak ratio method determine the activity, with uncertainty, of the unknown. Record this activity.

$$\frac{Area_{known}}{Activity_{known}} = \frac{Area_{unk}}{Activity_{unk}}$$

8. **<u>Discuss your results with your instructor or TA.</u>** Be sure to get the true activity, calibration date, and nuclide(s) of your unknown source.
9. Before logging out, **turn off the bias**, in **MCA – Adjust – Vacuum** select **Vent** (?) to return the chamber to normal pressure, and then close the detector file.
10. Close out of all programs.
11. If you are dismissed, lab leader please officially sign out of the remote desktop. Don't just close the remote window.
    a. This is accomplished by clicking on the start menu. See Figure 5 below.

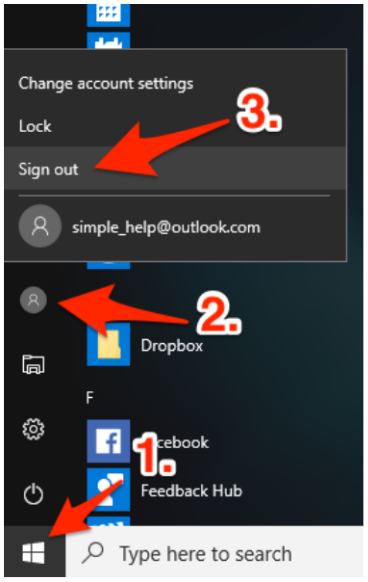


Figure 5. This shows the order of steps to officially sign out of the remote desktop.

Results and Discussion

Rough Draft due in lab week of 2/22:

Write a Results and Discussion on your Stopping Power plot. Include the plot and your calculation of Stopping Power. How does it compare to known value (assume a 5% error)? **DO NOT report the percentage difference or percentage error!** Use the statistics to compare the calculated and known values, i.e. do the values agree within 1σ or 2σ? If no, why?

Exploring Photon Scattering

Remote Laboratory Procedures

1. Purpose

The purpose of this laboratory is to explore the utility of the Compton scattering formula. The students will use a scattering table, NaI detector system and multi-channel analyzer (MCA) to explore Compton scattering. They will continue learning the operation of gamma spectrometers and how the Compton scattering formula can be used to predict the scattering energy of photons.

2. Lab Prep

For your theory section, describe the Compton scattering effect and how it is different from Incoherent scattering. Sketch the process and provide the equation for the interaction. Make a table for the information you will collect in 4.4 below. Prepopulate the table with the expected scattered photon energies.

3. Materials Needed

The following will be provided by the instructor:

1. A set of industrial sources
2. Compton scattering table
3. Inspector 2000 or DSA 1000 MCA
4. NaI detector
5. 14-pin 2" Pre-Amplifier tube base
6. A collimated 14.8 MBq (400 ∝Ci) ^{137}Cs source
7. Collimated detector holder

The student should bring the following:

1. a calculator
2. your notebook
3. a copy of these procedures
4. a pen

4. Experimental Procedure

The table below will help with the energy calibration.

Source	Half-life (yr)	Energy (keV)	Branching Ratio
^{137}Cs	30.1	662	0.85

4.0 Set Up the Remote Desktop Connection to the Laboratory Computer

1. VPN into the TAMU network using the Cisco AnyConnect Secure Mobility Client. Use these instructions if you have not done this before:
   https://agrilifelaserfiche.tamu.edu/documents/tamu-connect-vpn.pdf/
   https://it.tamu.edu/services/network-and-internet-access/virtual-private-networks/virtual-private-network-vpn/

1. Open the Remote Desktop client on your computer. For Windows users, you should already have the remote desktop program installed on your computer. For Mac users, use "Microsoft Remote Desktop" in the app store.
2. Your instructor should have given you an IP address of the laboratory computer. Use this IP address for connection.
3. Sign in with your institution's login information.
4. Open the "camera" application and change the camera to the Logitech webcam in the upper right-hand corner.
5. Download and open the Compton scattering LabVIEW program given to you by the professor/TA. **<u>DO NOT EDIT ANY OF THE CODE. If you edit the code, please return to an original version of the program.</u>**
6. Organize these windows to ensure you can see everything on your screen.

1.1 NaI System Setup with a Single Source Energy Calibration

1. Open the GENIE gamma acquisition software. Open the detector MID file "NAIDET" and set the HV to **<u>850 V</u>**. Turn on HV, **MCA>Adjust>HV.** Select the **Filter** button. Set rise time to 0.800 ∝s and flat top 0.2. Hit the **NEXT** button in the Adjust Window and adjust the LLD to eliminate to 3%.
2. **Setting the optimum voltage for your detector.**
    a. You will set a "1.5 MeV spectrum".
        i. The channel limit on your spectrum is 2048. In a 1.5 MeV spectrum 1.5 MeV = 2048. At what channel should the 662 keV photopeak from ^{137}Cs fall on? **(Hint: make a ratio)** Place your cursor in this channel.
        ii. Place the ^{137}Cs source in front of the detector at an angle of 0° as in Figure 1.
        iii. Adjust your course and fine gains until the photopeak centroid is near your cursor. This can be +/- 10 to 15 channels. This may require to continuously erasing your spectrum. **THERE IS NO NEED FOR YOU TO "STOP" ACQUIRING** during this process. **<u>Just erase.</u>**
3. Once the peak is near the specified channel, acquire a ^{137}Cs spectrum for a live time of 120 s. To change acquisition time go to **MCA>Acquire Setup**.
4. Using the Canberra GENIE-2000 software, energy-calibrate the spectrum using the 662 keV photopeak. On the menu bar select **Calibrate – Energy Only Calibration.** The calibration window will open. Enter the energy and centroid channel number for the peak and click the **Accept** button. Record the channel and energy information you used in the calibration.

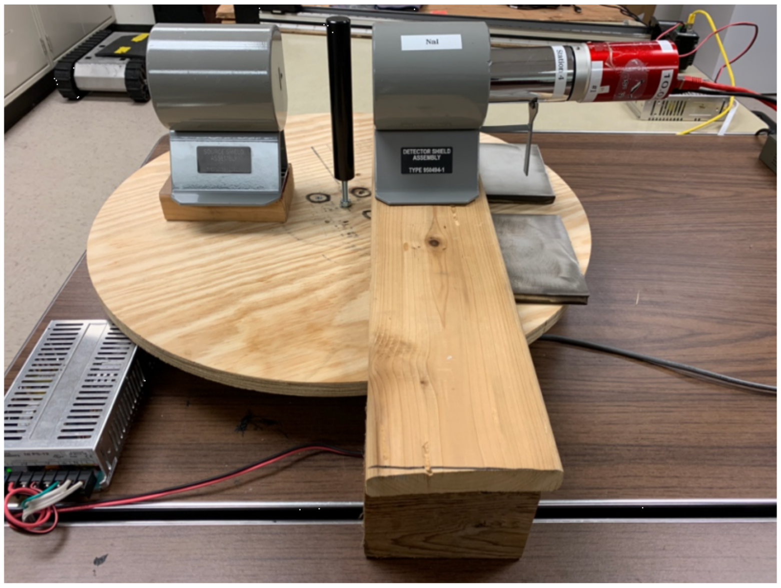


Figure 1. ^{137}Cs source placed at an angle of 0° with respect to the NaI detector.

1.1 Scattering Angle

1. In GENIE-2000, change to linear scale for the y-axis.
2. Change your acquisition time to 120 s (**MCA>Acquire Setup**). Acquire a spectrum.
3. Make 2 regions of interest (ROIs).
    a. Put your markers on either side of the cesium photopeak and hit **Insert**.
    b. There is a lower energy peak around 72 keV. Make an ROI for this peak using the procedure above.
    c. For each photopeak record the centroid energy (**the uncertainty for the energy will be FWHM/2.33**)., FWHM, channel number and the counts in that centroid channel
    d. Save the spectrum
4. Move the source to 30° using the Compton scattering LabVIEW VI. Assume an uncertainty in the angle of 1°. **Be sure to propagate this error!!**
    a. Change the acquisition time to 600 s.
    b. Acquire a spectrum
    c. Make new ROIs for any "new" photopeaks that appear
    d. Record the energy (**the uncertainty for the energy will be FWHM/2.33**), centroid channel number and counts in each centroid channel for all major peaks
    e. How do these energies compare to the scatterings energies predicted by the Compton formula?
5. Repeat step 6 for angels 40° through 70° in increments of 10°. Increased count times may be needed to generate defined scattered photon peaks.

1.2 Leaving the Remote Laboratory

1. Return the source back to an angle of 0° as a courtesy to the next student.
2. In the Genie Gamma Acquisition & Analysis software, go to **MCA>Adjust>HVPS** and turn the HV **OFF**.

1. If all of your data and results are saved on the laboratory computer, be sure to send it all to your personal computer via email, Google Drive, etc.
2. Close out of all programs and log off of the computer.

Due Next Week: Formal lab report

1. Write a full lab report on the Compton scattering portion of this lab
   a. Include how you energy calibrated.

Photoelectric Effect and Scintillation Detectors

Remote Procedures

1. Purpose

The purpose of this laboratory is to study the use of the NaI and LaBr detectors and their efficiencies. The student will use two types of scintillator detectors to help them understand how to perform energy and efficiency calibration for the system.

2. Lab Prep

Explain why do we have the following spectral features: Photo peak, Compton Edge, Compton continuum and backscatter peak. Draw a diagram showing these features and the expected energies of these peaks for a ^{137}Cs source.

Also make a table of **only** the sources you will be calibrating with. Prepopulate the table with the calibration isotope and their photon energies. The column headings will be Channel, Energy, Left Marker, Right Marker, Area and FWHM (Full Width Half Maximum).

3. Materials

Table 1. Sources of interest for this lab

Source	Half-life (yr)	Energy (keV)	Branching Ratio
^{137}Cs	30.1	662	0.85
^{60}Co	5.27	1173	1
		1332	1
^{109}Cd	1.24	88	0.36
^{54}Mn	0.85	835	1
^{22}Na	2.6	511	1.8
		1275	1
^{133}Ba	10.7	81	0.33
		302	0.19
		356	0.62

The following will be provided by the instructor:

1. A set of industrial gamma sources
2. 14-pin 2" pre-amplifier tube base (Canberra OSPREY)
3. NaI detector
4. LaBr detector

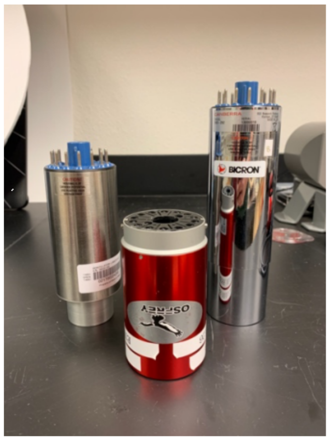


Figure 1. Canberra Osprey (center) with LaBr (left) and NaI (right) detectors.

Students should bring the following:

1. A calculator
2. Laboratory notebook
3. A copy of the procedure
4. A ruler
5. A pen

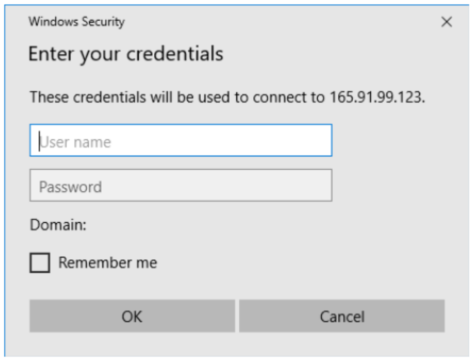
1. Remote Login and Join Zoom Meeting

1. Join the remote lab in Zoom at least 5 minutes before your lab time starts.
2. The instructor/TA will assign a station leader. Lab leaders, remotely log into the remote lab desktop computer
   a. If off campus the station leader needs to sign into TAMU's VPN.
   b. The station leader will use their Remote Desktop Application to login into the NE013 computer.

   Computer IP Address

   Remote Lab: 165.91.99.158

   c. **IMPORTANT:** WHEN LOGGING IN BE CARFUL OF THE DOMAIN YOU ARE IN. THE DEFAULT DOMAIN IS THE DOMAIN YOU ARE IN ON YOUR COMPUTER


Windows Security

Enter your credentials

These credentials will be used to connect to 165.91.99.123.

User name

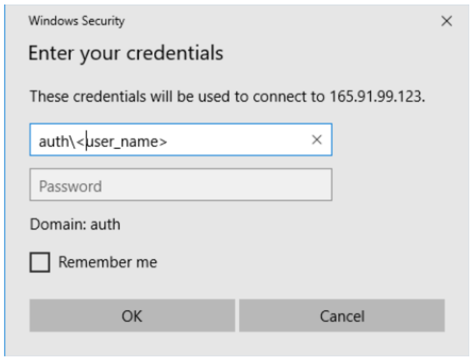
Password

Domain:

☐ Remember me

OK Cancel

- You must login using you TAMU netID. To do this you must be in the "auth" domain. In the username window type "auth\<your_user_name> and then user your netID password. See the picture below:


Windows Security

Enter your credentials

These credentials will be used to connect to 165.91.99.123.

auth\<user_name>

Password

Domain: auth

- Leaders will be the first to operate the system. Leaders can allow their lab partners to operate the desktop through the zoom meeting.
- Leaders, share your screen by clicking the **Share Screen** button in the Zoom menu bar (usually at the bottom of the screen).
- Another participant in the breakout room can click on the **View Options** next to the Zoom's green bar (usually at the top of the screen), and then select **Request Remote Control**. Hit **Request**.
- The person who is sharing the screen, please allow your lab partner to control your screen by hitting **OK** when Zoom asks you to allow remote control of your screen. **NOTE:** In some cases, you would have to change the Privacy Settings of your computer to allow Zoom to control your computer.

1. Once you are in the system and your partners are sharing your screen, start the lab station's camera. This can be done by opening the "Camera" application and changing the camera in the upper right hand corner if necessary.

1. Experimental Procedure

1.1 Energy Calibration for NaI Detector

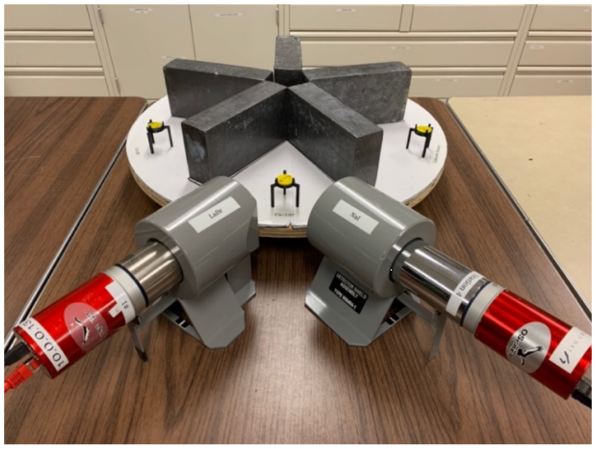


Figure 2. Experiment apparatus with LaBr on the left and NaI on the right.

Notice that the NaI and LaBr detectors are in lead collimators each with a 30 mm aperture (Figure 2). They are pointed towards the same spot on the rotary table so that a source can be measured by either detector when needed. Each source is on its own holder to place it at the same vertical height

1. as the collimator aperture. The detectors are connected to their own Canberra OSPREY digital MCA tube base. The OSPREY contains a high-voltage power supply and preamplifier. It connects to the computer through ethernet.
2. Open Genie Gamma Acquisition and Analysis.
3. Click on the "Open datasource".
4. Select Detector from "Source" (About half way down in the Open File Window).
5. Select "303NAI".
6. Adjust Setting
   a. Go to **MCA>Adjust>HVPS** and set the HV to 850 V. Click the "OK" button right next to the HV value. Then turn **ON** the HV. (Wait for the HV to fully ramp up before using the detector).
   b. Assure Filter settings are correct. Select the **Filter** button. Set rise time to 0.800 ∝s and flat top 0.2.
   c. Go to Gain. Hit the **NEXT** button in the "Adjust" window and make sure the LLD is at 1.5 % to eliminate low energy noise.
   d. Go to **(MCA>Acquire Set-up…).** Adjust your acquisition time (Time Preset) to 200 seconds.
7. Open the "Scintillation Detectors Lab" LabVIEW VI given to you by the professor/TA. Be sure that opens in LabVIEW 2018. Press the right-pointing arrow in the taskbar (upper left corner of the window). This will begin the program.
8. Place a ^{137}Cs source in front of the detector face by selecting "Cs-137" in the Source Position drop-down menu. Begin the move by clicking on "Start Move" (Figure 3). Ask the TA for the source activity and calibration date.

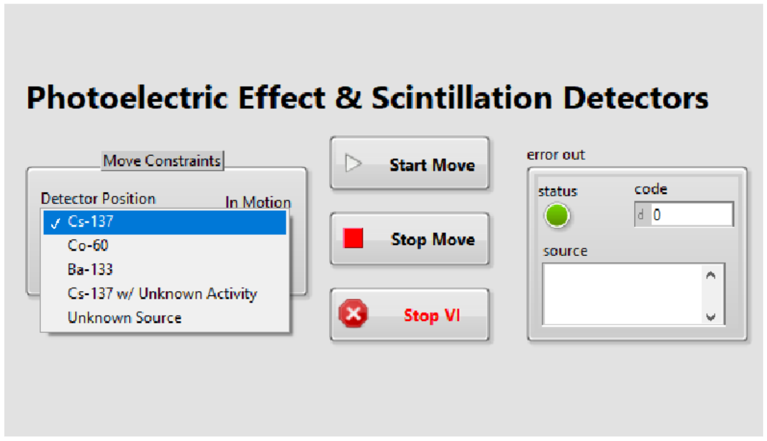


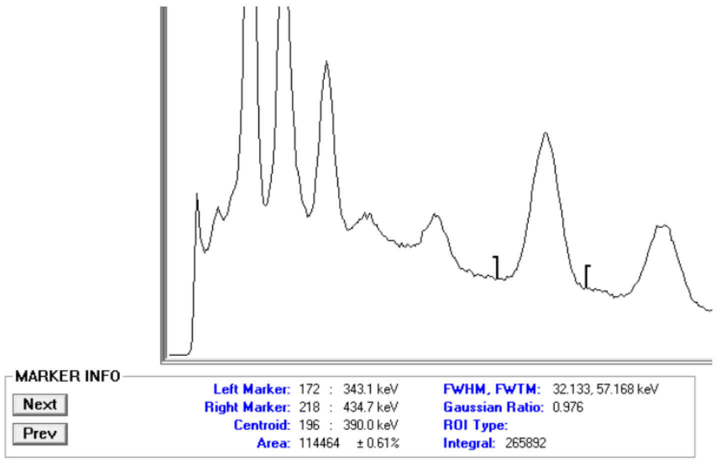
Figure 3. Scintillation detectors LabVIEW VI used to control the rotary table.

9. In this step you are "setting the scale" of the spectrum. This is the maximum energy displayed in the spectrum.
   a. **Start** an acquisition (Under Acquire to the left of the spectrum).
   b. Before adjusting the Gain settings, calculate at what channel the 662-keV photopeak from ^{137}Cs should be in a 1.5 MeV spectrum. The channel limit on your spectrum is 2048. **Hint**: If channel 2048 corresponds to 1.5 MeV, assuming linear relationship between channel number and energy, what channel would correspond to 662 keV?

$$\frac{1.5MeV}{0.662MeV} = \frac{2048}{x}$$

   c. Place your cursor in this channel.

   a. Go back to the **MCA>Adjust** window and select the **Gain**.
   b. Adjust the Coarse and Fine Gain settings so that the peak coincides with the cursor (the channel that you just calculated ± 20 channels). **This may require you hitting the "Clear" button occasionally to clear the spectrum. There is no need for you to hit Stop… then Start… repeatedly**
2. Once the ^{137}Cs peak is where you want it, accumulate a spectrum for 200 s.
   a. Save the spectrum in its native .CNF format
   b. Once the acquisition is complete, make sure that your cursor is at the apex of the peak. Go to **Calibrate – Energy Only Calibration.** Click on the Cursor button (The channel # should appear in the Channel box. Clicking this button makes Genie put the channel number where your cursor currently is).
   c. Enter 662 in the **Energy** box and click **Accept.**
   d. Click **OK.**
3. Record peak features
   a. In your "INFO" box below the spectrum, click Next until you get to MARKER INFO (Figure 4).
   b. Put your "Markers" on either side of the Peak.
      i. Put the cursor on the left side of the peak. Press **Ctrl L.**
      ii. Put your Cursor on the right side of the peak. Press **Ctrl R.**
      iii. Record what channels the markers are on.
      iv. Record the channel number for the left and right markers, centroid, FWHM and Area.
      v. Record the photopeak, backscatter peak, and Compton edge. Be sure to include the channel number and energy for each.
4. Press the insert button on your keyboard.
   a. This will set a region of interest (ROI) around the cesium photopeak. This ROI will be used later.
5. Replace your Cs source with a ^{60}Co source by selecting "Co-60" from the Source Position drop-down menu in the LabVIEW VI (Figure 3).
6. Clear the spectrum and acquire a ^{60}Co spectrum for 200 s.
7. Save the spectrum
8. Energy-calibrate the detector system using the two photopeaks from ^{60}Co. You can do this by going to **Calibrate>Energy Full>By Nuclide List**. Be sure to check the **"<u>append to existing calibration</u>".** Select Co-60 from the list. Another window will appear listing the photopeaks. In the window click on the "1173.00" entry. Now in the spectrum put your cursor on the centroid of the lower energy peak. Click the "**Cursor**" button. This will enter the channel number of the peak. Repeat this for 1332 keV.
   a. Put your "Markers" on either side of the Peak.
      i. Put the cursor on the left side of the peak. Press **Ctrl L.**
      ii. Put your Cursor on the right side of the peak. Press **Ctrl R.**
      iii. Record what channels the markers are on.
      iv. Record the channel number for the left and right markers, centroid, FWHM and Area.



```
Left Marker:  172  :  343.1 keV     FWHM, FWTM:  32.133, 57.168 keV
Right Marker: 218  :  434.7 keV     Gaussian Ratio: 0.976
Centroid:     196  :  390.0 keV     ROI Type:
Area:         114464   ± 0.61%      Integral:  265892
```

Figure 4. Spectrum with markers around a reference peak. Below the spectrum is the MARKER INFO window.

1. Clear the spectrum. Replace the ^{60}Co with ^{133}Ba by selecting "Ba-133" from the Source Position drop-down menu in the LabVIEW VI.
   a. Save the spectrum
   b. Repeat the calibration process as in step 14, but for the ^{133}Ba peak.
2. Set your markers as you did before and record the information. Record the energy and FWHM for each peak (You will use this to compare to the LaBr spectrum) listed in Table 1.
   a. **BEFORE YOU HIT "OK", CLICK THE "SHOW" BUTTON. SHOW THIS TO YOUR TA TO MAKE SURE YOUR CALIBRATION IS GOOD.**
3. Using the information in your notebook table, create a table in Microsoft Excel or another spreadsheet program, plot energy (y-axis) versus channel number (x-axis). Be sure to include the equation of the line and R^2 value. This is your energy calibration equation.

3.2 Unknown Activity and Unknown Isotope

1. Place the Cs-137 source with unknown activity in front of the NaI detector by selecting "Cs-137 w/ Unknown Activity" from the Source Position drop-down menu in the LabVIEW VI.
   a. Count the source for 200 s.
   b. Save the spectrum
   c. Use the ratio method to determine the activity of the unknown source. Remember that for the ratio method to work the known and unknown source must be counted for the same amount of time.

$$\frac{Area_{known}}{Activity_{known}} = \frac{Area_{unk}}{Activity_{unk}}$$

      i. The uncertainty in your count is the square root of the area
      ii. Assume the uncertainty of your known source activity is 5%.
      iii. Do not forget to ask the TA for the information of the unknown Cs source.
2. **Identify an unknown isotope.** Place the unknown source in front of the NaI by selecting "Unknown Source" from the Source Position drop-down menu in the LabVIEW VI. Collect a spectrum for 200 s.
   a. Save the spectrum

    a. Record the peak energies and areas.
    b. Identify the source(s) based on the energy information from Table 1.
2. Turn off the HV. **(MCA>Adjust>HV>Off) (Wait for the HV to fully ramp down).**

3.3 Resolution Comparison: NaI/ LaBr

1. Close GENIE and open it again.
2. Click on the “Open datasource”.
3. Select Detector from “Source” (About half way down in the Open File Window).
4. Select “303LABR”.
5. Adjust Settings
    a. Go to **MCA>Adjust>HVPS** and set the HV to 650 V. Click the “OK” button right next to the HV value. Then turn **ON** the HV. (Wait for the HV to fully ramp up before using the detector).
    b. Assure Filter settings are correct. Select the **Filter** button. Set rise time to **0.800** ∝s and flat top 0.2.
    c. Go to Gain. Hit the **NEXT** button in the “Adjust” window and make sure the LLD is at 1.5 % to eliminate low energy noise.
    d. Go to **(MCA>Acquire Set-up…).** Adjust your acquisition time (Time Preset) to 200 seconds.
6. Place a ^{137}Cs source in front of the detector face by selecting “Cs-137” in the Source Position drop-down menu. Begin the move by clicking on “Start Move” (Figure 3).
7. In this step you are “setting the scale” of the spectrum. This is the maximum energy displayed in the spectrum.
    a. **Start** an acquisition (Under Acquire to the left of the spectrum).
    b. Before adjusting the Gain settings, calculate at what channel the 662-keV photopeak from ^{137}Cs should be in a 1.5 MeV spectrum. The channel limit on your spectrum is 2048. **Hint:** If channel 2048 corresponds to 1.5 MeV, assuming linear relationship between channel number and energy, what channel would correspond to 662 keV?
    $$\frac{1.5MeV}{0.662MeV} = \frac{2048}{x}$$
    c. Place your cursor in this channel.
    d. Go back to the **MCA>Adjust** window and select the **Gain.**
    e. Adjust the Coarse and Fine Gain settings so that the peak coincides with the cursor (the channel that you just calculated ± 20 channels). **This may require you hitting the “Clear” button occasionally to clear the spectrum. There is no need for you to hit Stop… then Start… repeatedly**
8. Once the ^{137}Cs peak is where you want it, accumulate a spectrum for 200 s.
    a. Save the spectrum in its native .CNF format
    b. Once the acquisition is complete, make sure that your cursor is at the apex of the peak. Go to **Calibrate – Energy Only Calibration.** Click on the Cursor button (The channel # should appear in the Channel box. Clicking this button makes Genie put the channel number where your cursor currently is).
    c. Enter 662 in the **Energy** box and click **Accept.**
    d. Click **OK.**
9. Record peak features
    a. In your “INFO” box below the spectrum, click Next until you get to MARKER INFO (Figure 4).
    b. Put your “Markers” on either side of the Peak.
        i. Put the cursor on the left side of the peak. Press **Ctrl L.**
        ii. Put your Cursor on the right side of the peak. Press **Ctrl R.**

        i. Record what channels the markers are on.
        ii. Record the channel number for the left and right markers, centroid, FWHM and Area.
        iii. Record the photopeak, backscatter peak, and Compton edge. Be sure to include the channel number and energy for each.

2. Press the insert button on your keyboard.
    a. This will set a region of interest (ROI) around the cesium photopeak. This ROI will be used later.
3. Replace your Cs source with a ^{60}Co source by selecting "Co-60" from the Source Position drop-down menu in the LabVIEW VI (Figure 3).
4. Clear the spectrum and acquire a ^{60}Co spectrum for 200 s.
5. Save the spectrum
6. Energy-calibrate the detector system using the two photopeaks from ^{60}Co. You can do this by going to **Calibrate>Energy Full>By Nuclide List**. Be sure to check the **"<u>append to existing calibration</u>"**. Select Co-60 from the list. Another window will appear listing the photopeaks. In the window click on the "1173.00" entry. Now in the spectrum put your cursor on the centroid of the lower energy peak. Click the "**Cursor**" button. This will enter the channel number of the peak. Repeat this for 1332 keV.
    a. Put your "Markers" on either side of the Peak.
        i. Put the cursor on the left side of the peak. Press **Ctrl L.**
        ii. Put your Cursor on the right side of the peak. Press **Ctrl R.**
        iii. Record what channels the markers are on.
        iv. Record the channel number for the left and right markers, centroid, FWHM and Area.
7. Replace the ^{60}Co with ^{133}Ba, clear the spectrum and after taking a 200 s spectrum
    a. Save the spectrum
    b. Repeat the calibration process as in step 13, but for the ^{133}Ba peak.
    c. Set your markers as you did before and record the information.
    d. **<u>BEFORE YOU HIT "OK", CLICK THE "SHOW" BUTTON. SHOW THIS TO YOUR TA TO MAKE SURE YOUR CALIBRATION IS GOOD.</u>**

3.4 Ba-133 Resolution Comparison

1. You acquired ^{133}Ba spectra using NaI and LaBr. In your notebook compare the peaks you identified and the resolution of each peak. How were the spectra different?
2. Close out of all programs.
3. If you are dismissed, lab leader please officially sign out of the remote desktop. Don't just close the remote window.
    a. This is accomplished by clicking on the start menu. See Figure 5 below.

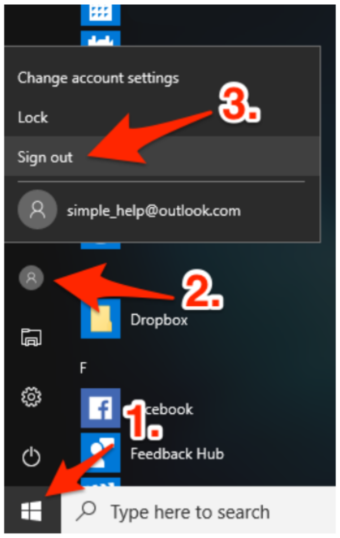


Figure 5. This shows the order of steps to officially sign out of the remote desktop.

Due Next Week in Lab: Worksheet Lab 5

1. For the NaI
   a. Make a table of Energy and FWHM as a function of channel number (put channel number in the first column).
   b. Make a plot of Energy as a function of Channel number. Include a curve fit and R^2 equation in the plot.
2. For the LaBr
   a. Make a table of Energy and FWHM as a function of channel number (put channel number in the first column).
   b. Make a plot of Energy as a function of Channel number. Include a curve fit and R^2 equation in the plot.
3. For 4.2.7, calculate the activity of your "unknown" source. **Show your work and include uncertainty. How does it compare to the actual activity?**
4. For your unknown ID in 4.2.8, make a table of the photopeak energies you identified. Based on these energies, state what you thought the unknown sources were.

Attenuation Coefficients

Remote Laboratory Procedures

1. Purpose

One of the disadvantages to gamma detection is that they can be attenuated by all materials. Typically a simple equation is used to predict the level of attenuation a given material will have (Eq. 1).

$$I = I_0 e^{-\propto t} \qquad \text{Eq. 1}$$

Where,
I = intensity of detected photons with attenuation
I_o = Intensity from photon source without an attenuator (counts, count rate, exposure rate.....)
$\propto$= linear attenuation coefficient (cm^{-1} or mm^{-1})
t = thickness of material (cm or mm)

The linear attenuation can be determined by taking a series of measurements while adding increasing the thickness of a material. The purpose of this laboratory is to show the students a technique to determine linear attenuation coefficient of various shielding materials.

Lab Prep

- Briefly describe the meaning of the attenuation coefficient in words.

- In this lab you are going to plot the natural log of the cesium photopeak area (counts) versus absorber thickness. Area will be the "I" from Eq 1. Put Eq 1 in the form of a linear equation and show the attenuation coefficient is the slope. **(hint: take the natural log of both sides of Eq. 1)**

- Create tables in your notebook for each absorber you will use in this experiment. The tables will have 4 columns: Material Thickness (mm), Thickness Uncertainty (assume 0.01 mm), Photopeak Count, Count Uncertainty (square root of the area). The table will have at least 4 rows in addition to the row for column titles.

2. Materials

The following will be provided by the instructor:

1. A ^{137}Cs source
2. Pre-amplifier tubebase
3. NaI detector Attenuator set

Students should bring the following:

1. A calculator
2. Laboratory notebook
3. A copy of the procedure
4. A pen

1. Experimental Procedure

Remote Desktop Connection

1. VPN into the TAMU network using the Cisco AnyConnect Secure Mobility Client. Use these instructions if you have not done this before:
   https://agrilifelaserfiche.tamu.edu/documents/tamu-connect-vpn.pdf/
   https://it.tamu.edu/services/network-and-internet-access/virtual-private-networks/virtual-private-network-vpn/
2. Open the Remote Desktop client on your computer. For MAC users, use "Microsoft Remote Desktop" in the app store. For Windows users, you should already have the remote desktop program installed on your computer.
3. Enter the following IP address: 165.91.99.158
4. Sign in with your NET ID login and be prepared to use Duo.
5. Open the "camera" application and set the camera to the Logitech webcam in the upper right-hand corner. You should see the NaI detector and the linear slide as shown in Figure 1.
6. Download and open the Gamma Attenuation LabVIEW program given to you by the professor/TA. **DO NOT EDIT ANY OF THE CODE.**
7. Launch the program by selecting the arrow in the upper left-hand corner of the LabVIEW window.
8. Organize these windows to ensure you can see everything on your screen.

3.1 Setting the channel for the cesium peak

1. Open Genie Gamma Acquisition and Analysis.
2. Click on the "Open datasource"
3. Select Detector from "Source". (About half way down in the Open File Window)
4. Select "NAIDET"
5. Adjust Setting
   a. Be sure the HV is at 850 V **(MCA>Adjust>HVPS)**. Then turn **ON** the HV. (Wait for the HV to fully ramp up before using the detector)
   b. Assure Filter settings are correct. Select the **Filter** button. Set rise time to **0.800** μs and flat top 0.2.
   c. Go to **(MCA>Acquire Set-up…).** Adjust your acquisition time (Time Preset) to 120 seconds.
6. Rotate the rotary table so that there are not any slabs of material between the source and the detector (No Attenuator in the drop-down menu) (Figure 1-2).
7. **Start** an acquisition
8. Select the Gain tab in **MCA>Adjust** window. Adjust the Coarse and Fine Gain settings so the peak at near channel 1400 (+/- 20 channels). This may require you hitting the "Clear" button occasionally to clear the spectrum. There is no need for you to hit Stop… then Start… repeatedly.
9. Once the peak is where you want it, acquire a 120 s spectrum.
   a. Record the spectrum
10. After acquisition record the peak features
    a. In your "INFO" box below the spectrum click Next until you get to MARKER INFO
    b. Put your "Markers" on either side of the Peak.
       i. Put the cursor on the left side of the peak. Press **Ctrl L.**
       ii. Put your Cursor on the right side of the peak. Press **Ctrl R.**

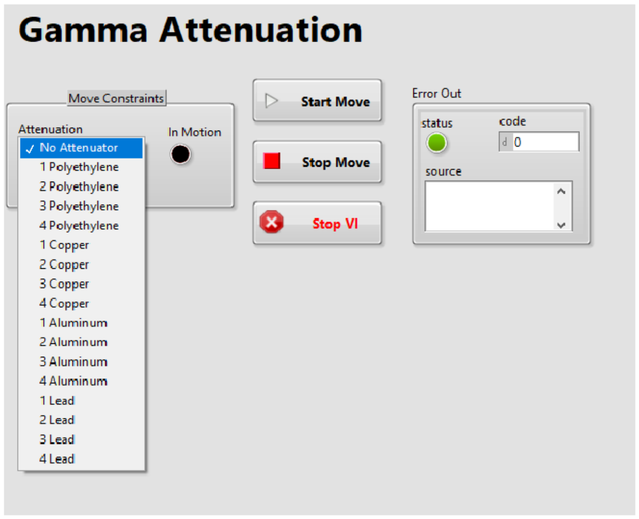
i. Press the "insert" button on your keyboard. This will set an ROI.

2. The area of this peak will be your *Io* for all the attenuation coefficients below.

NOTE: This week won't require a detailed energy calibration since all we care about is the Cs-137 peak. The only thing that will change as you add attenuation are the counts in the 662 keV peak.

3.2 Determination of linear attenuation coefficient for polyethylene.

1. Rotate the table with the LabVIEW VI so that 1 slab of polyethylene is between the source and the detector. The thickness of 1 slab of polyethylene is 12.7 mm. Assume 0.01 mm uncertainty **per slab.**



Gamma Attenuation

Move Constraints

Attenuation: ✓ No Attenuator

In Motion: ●

Error Out: status ●, code 0, source

Figure 2. LabVIEW VI showing selection of No Attenuator.

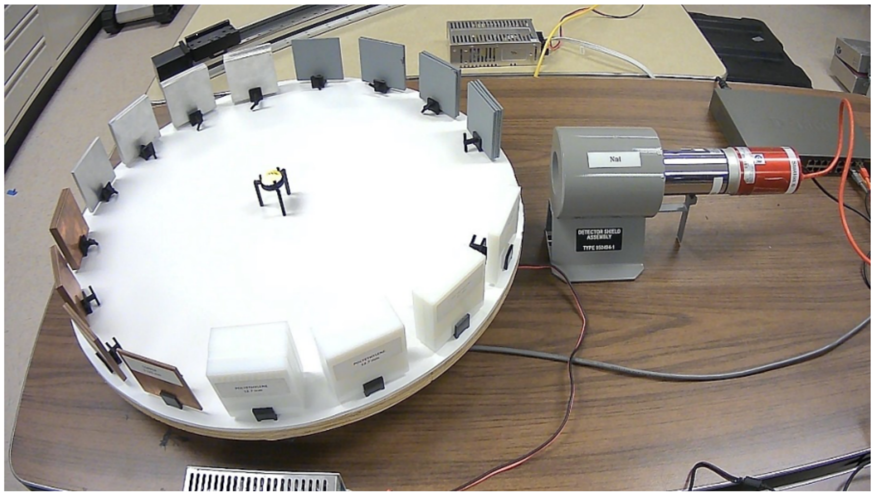


Figure 2. Apparatus of the experiment.

1. Acquire a 120 s spectrum
2. Record the Area for the peak.
3. Cycle the table for 2 slabs of polyethylene and record the total thickness of polyethylene.
4. Acquire a 120 s spectrum
5. Record the Area for the peak.
6. Repeat steps 4-6 until 4 slabs of polyethylene are used.

3.3 Determination of copper, aluminum, and polyethylene linear attenuation coefficient

1. Repeat steps 2-7 from 3.2 for copper, aluminum and lead. Use the count data from 3.1 in each table for your "I_0" thickness.
   a. The thickness of 1 slab of copper is 3.175 mm
   b. The thickness of 1 slab of aluminum is 6.35 mm
   c. The thickness of 1 slab of lead is 3.175 mm
   d. Assume 0.01 mm uncertainty **per slab** for all materials

3.4 Leaving the Remote Laboratory

1. In the Genie Gamma Acquisition & Analysis software, go to **MCA>Adjust>HVPS** and turn the HV **OFF**.
2. Return the rotary table back to without any slabs of material as a courtesy to the next student/group.
3. If all of your data and results are saved on the laboratory computer, be sure to send it all to your personal computer via email, Google Drive, etc.
4. Close out of all programs and log off of the computer.

Due next week in lab

1. Rough draft of a formal report:
   a. There will be a Theory section for this report. In this section discuss Compton scattering and how it applies to this lab. Include Eq 1 (above) in your theory and how you will use it to determine the linear attenuation coefficients for each material.
   b. In your results sections include Tables and plots of your data
   c. Be sure to compare your experimental results (with uncertainty) to the accepted values

Gamma-ray Spectroscopy with HPGe's and Uranium Enrichment

Remote Laboratory Procedures

1. Purpose

The purpose of this laboratory is to study the basic operation of HPGe detectors for quantitative assessment of radioactive samples. The student will setup an HPGe detector system using a Canberra MCA and spectrum analysis software then learn how to energy and efficiency calibrate the system. The will also learn to use the comparator method to determine the enrichment of uranium.

Lab Prep: Quiz on this procedure

2. Materials Needed

The following will be provided by the instructor:

1. set of sources
2. a Falcon 5000 HPGe detector
3. a digital MCA system with spectrum analysis software

The student should bring the following:

1. a copy of these procedures
2. a pen
3. your notebook
4. a calculator
5. a ruler

Source	Half life (yr)	Energy (keV)	Branching Ratio
^{137}Cs	30.1	662	0.85
^{60}Co	5.27	1173	1
		1332	1
^{133}Ba	10.7	384	0.09
		356	0.62
		303	0.18
		276	0.073
		81	0.33

Table 1. List of radionuclides used in this experiment.

3. Experimental Procedure

3.0 Set Up the Remote Desktop Connection to the Laboratory Computer

1. VPN into the TAMU network using the Cisco AnyConnect Secure Mobility Client. Use these instructions if you have not done this before:

https://agrilifelaserfiche.tamu.edu/documents/tamu-connect-vpn.pdf/
https://it.tamu.edu/services/network-and-internet-access/virtual-private-networks/virtual-private-network-vpn/

2. Open the Remote Desktop client on your computer. For Windows users, you should already have the remote desktop program installed on your computer. For Mac users, use "Microsoft Remote Desktop" in the app store.

3. Your instructor should have given you an IP address of the laboratory computer. Use this IP address for connection.

4. Sign in with your institution's login information.

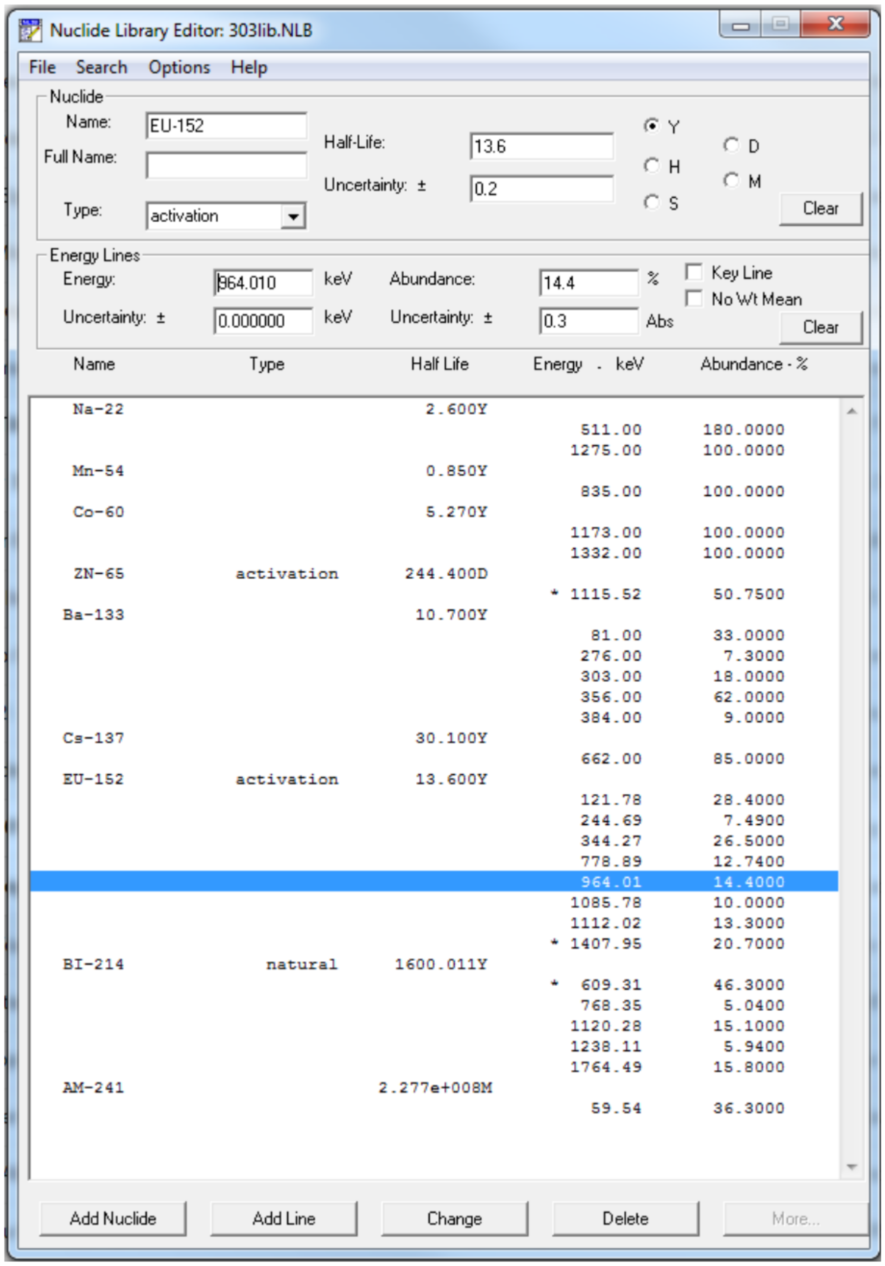
5. Open the "camera" application and change the camera to the Logitech webcam in the upper right-hand corner.

6. Download and open the Uranium Enrichment LabVIEW program given to you by the professor/TA. **DO NOT EDIT ANY OF THE CODE. If you edit the code, please return to an original version of the program.**

7. Organize these windows to ensure you can see everything on your screen.

1.1 Create a library in GENIE

1. On your computer's desktop start menu: **Type: "Nuclide Library Editor"** the program should appear hit return to start it.



Nuclide

Name: EU-152

Full Name:

Type: activation

Half-Life: 13.6

Uncertainty: ± 0.2

[✓ Y] H S D M

Energy Lines

Energy: 964.010 keV

Uncertainty: ± 0.000000 keV

Abundance: 14.4 %

Uncertainty: ± 0.3 Abs

☐ Key Line

☐ No Wt Mean

```
Name       Type            Half Life     Energy  -  keV    Abundance - %

Na-22                         2.600Y
                                                511.00        180.0000
                                               1275.00        100.0000
Mn-54                         0.850Y
                                                835.00        100.0000
Co-60                         5.270Y
                                               1173.00        100.0000
                                               1332.00        100.0000
ZN-65      activation       244.400D
                                             * 1115.52         50.7500
Ba-133                       10.700Y
                                                 81.00         33.0000
                                                276.00          7.3000
                                                303.00         18.0000
                                                356.00         62.0000
                                                384.00          9.0000
Cs-137                       30.100Y
                                                662.00         85.0000
EU-152     activation        13.600Y
                                                121.78         28.4000
                                                244.69          7.4900
                                                344.27         26.5000
                                                778.89         12.7400
                                                964.01         14.4000
                                               1085.78         10.0000
                                               1112.02         13.3000
                                             * 1407.95         20.7000
BI-214        natural      1600.011Y
                                             *  609.31         46.3000
                                                768.35          5.0400
                                               1120.28         15.1000
                                               1238.11          5.9400
                                               1764.49         15.8000
AM-241                     2.277e+008M
                                                 59.54         36.3000
```

Figure 1. Radionuclides in Table 1 and additional sources added to the library (saved library).

1. Using this tool create a library from the sources listed in Table 1. To accomplish this, first fill out the top most portion of the window which contains the name of the isotope you are adding (Ex: ^{137}Cs, and its half-life, as shown in Table 1). Hit **Add Nuclide**. You can leave uncertainty blank. Now fill out the energy line information and its abundance percentage based on Table 1, and hit **Add Line**. You can leave the uncertainty at 0. Do this for each nuclide in Table 1.

NOTE: Remember, click **Add Nuclide** to add a nuclide in your library, and click **Add Line** to insert the energy peak information. For ^{60}Co and ^{133}Ba, you need to do the **Add Line** process for each gamma the isotope emits. Figure 1 shows the entered details in the library.

1. Add additional sources. Go to **Options > Extract**. Open **STDLIB.NLB**. Click on Zn-65, ^{152}Eu, ^{241}Am and ^{214}Bi. Check that these sources are selected by ensuring that these sources are highlighted in blue as you scroll back up the source list. Hit **OK**. Several peaks will show for each of these sources. Delete lines that are less than 5% abundant.
2. Now add ^{235}U and ^{238}U. Go to **Options > Extract.** Open ANSI_GammaGuru.nlb. Select ^{235}U and "U-238+dau". Do not delete any peaks.
3. Save your library as 605lib_(DAY OF YOUR LAB).

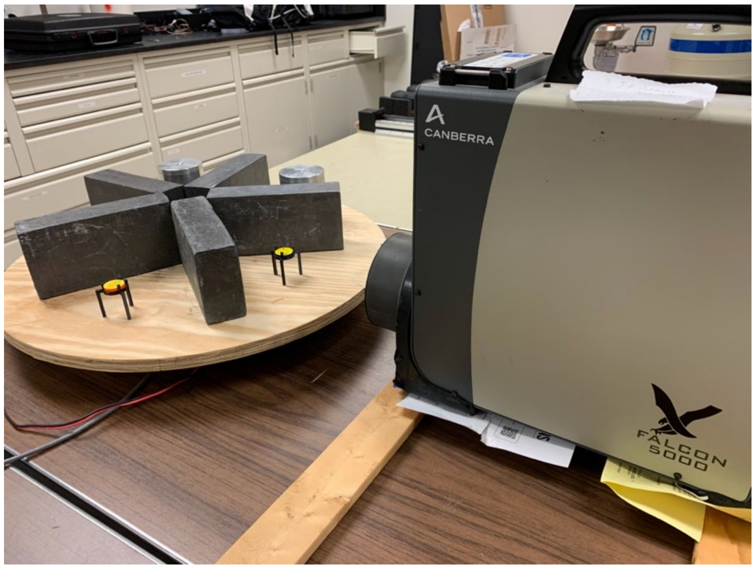


Figure 2. Apparatus of the experiment.

1.1 Energy Calibration.

1. Open the GENIE software. Go to **File > Open**. Select the **Detector** button and choose the HPGe mid file. Go to **MCA > Adjust**. Change the LLD to 0.5%. Verify that the HV potential and polarity are properly set. Turn on your HV. Set the acquisition time to 120 s. (**MCA > Acquire Set up…)**
2. The rotary table should already have the ^{137}Cs source in front of the detector.
   a. Set your MCA to a "**3 MeV scale**" using the equation provided. Your maximum channel number is 8192. Write the expected channel number in your notebook.

$$\text{Expected channel number} = \frac{8192}{3\ MeV} \times 661.7\ keV$$

   b. Place your cursor on the channel number you just calculated. The channel number can be seen in Figure 5 in the red box.
3. Once your scale is set, acquire a spectrum for 120 s.
4. Once 120 s has elapsed calibrate the spectrum. **Calibrate > Energy Only.** Put your cursor on the peak centroid**.** Click the **Cursor** button. Enter the peak energy. Hit **Accept**. Record the channel number and energy of the photopeak, backscatter peak and Compton edge.

1. Save the spectrum
2. **^{152}Eu Measurement.** Rotate the rotary table to replace the ^{137}Cs source with a ^{152}Eu source. The location of this source and all other sources used in this section are kept the same in order to acquire accurate efficiency data.
3. **Count the ^{152}Eu** source for 120 s.
4. Using the Canberra GENIE-2000 software, add the ^{152}Eu photopeaks to the energy calibration for the detector system. You can do this by using the **Energy Full > By Nuclide List** command. Select your 605lib. Remember to check "Append to Existing Calibration" so that it adds this calibration point to the existing calibration. Put the cursor on the spectrum over the peak centroid. Click on the **Cursor** button. Record the channel and energy information. Delete any peaks that do not have any relevance in your calibration.
5. **<u>Before exiting out of the calibration window, get the TA or instructor and "Show" them your calibration curve</u>**
6. "Save" your settings by clicking on the save icon at the top of the GENIE window.
7. Also, save the spectrum

1.1 Efficiency Calibration

1. Go to **Calibrate>Efficiency>By Nuclide list.** Select ^{152}Eu. Do **NOT** check "Append to existing calibration."
2. Click on the **Additional Information** button.
3. Enter the assay date on the source. For time enter 12:00:00.
4. Enter the activity of the source and add an uncertainty of 10%. Click the **Change** button and the information should update in the window.
5. Hit **OK** then hit **OK** in the remaining window. The efficiency calibration window should appear. Hit **Auto** on the bottom right of the window. Your peak efficiencies should automatically fill in. If they don't, you may not have recorded the spectrum for long enough or your energy calibration is off.
6. Once all of the efficiencies are entered hit **OK.**
7. Go to **Calibrate>Efficiency Show.** Show the instructor your curve.
8. After showing the instructor your curve hit accept the calibration
9. Save the spectrum
10. Replace the ^{152}Eu source with ^{133}Ba.
    a. Take a 120 s
    b. Go to **Calibrate>Efficiency>By Nuclide list.** Select ^{133}Ba. This time, check "Append to existing calibration."
    c. Repeat steps 2-8 above for ^{133}Ba
11. "Save" the settings by clicking on the save icon at the top of the GENIE window.
12. Save the spectrum

1.2 U Measurements

1. Rotate the rotary table so that the first uranium sample is in front of the detector. This sample will serve as your standard.
2. Place this sample next to the detector.

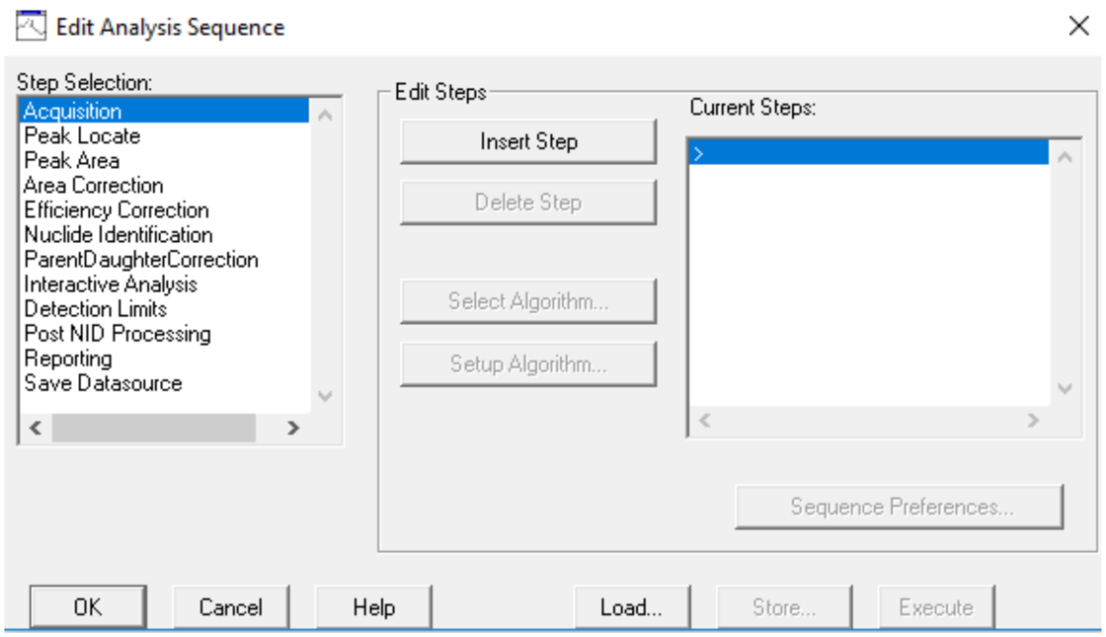


Figure 3. Edit Analysis Sequence window that can be used to adjust the library for identification and quantification.

1. Acquire a spectrum for 5 minutes. This measurement time must be long enough that the 1001 keV peak for U-238 has a sufficient number of counts.
2. Go to **Analysis>Executive sequence>NID Analysis with Report.**
3. Record the peak energies of what was found, what the nuclides identified were along with the calculated activity. Record the net area (i.e. counts) of the 186 keV and 1001 keV peaks.
4. Save the spectrum
5. Repeat steps 4-7 for a second uranium sample.
6. Using the simple comparator method discussed in class, estimate the enrichment of the second uranium sample.

Results and Discussion

1. Write a results and Discussion for 3.5. Make a table for the data to include the area for each peak and the calculated enrichment. Propagate error by assuming the error in the known enrichment is ± 5% the recorded enrichment. How does the calculated enrichment compare with the known enrichment?