Teaching the Sonification of Snake Robot Movement

Using Project-Based Learning Activities

by

Lynn E. Connelly

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master's in Robotics Engineering
CECS
in the University of Michigan-Dearborn
2025

Master's Thesis Committee:

Associate Professor Alireza Mohammadi, Chair
Professor Brahim Medjahed
Dr. Furat Al-Obaidy

Dedication

To my husband, Pat, and my children, Piper, Parker, Price and Pepper without whom I would have never been able to complete this degree while also working full-time, raising a family, and making sure that I am there for you every day. Thank you for your endless patience and understanding. All of you are role models for me and your love, encouragement, and support have given me the strength to conquer any challenge and never give up. I am lucky to have all of you in my life and am forever grateful for all your love.

Acknowledgements

I would like to express my deep appreciation to my dissertation advisor, Dr. Alireza Mohammadi from the University of Michigan – Dearborn. His guidance, support, encouragement, and technical knowledge have been essential throughout this research process and in the success and completion of this paper. His kindness, camaraderie, and professionalism has made this experience as smooth as possible, especially given that we work about 700 miles apart from each other. I am forever grateful.

I would also like to thank Dr. Furat Al-Obaidy and Dr. Brahim Medjahed for serving on my dissertation defense committee. Your insight and guidance with teaching college-level research using PBL activities in the high school classroom is invaluable and will be used as I continue to develop this program at my own high school.

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations

4C's …..…….…… Critical Thinking, Creativity, Communication, and Collaboration

CAD ……..…….Computer-Aided Design

COM ….…….……...….…Center of Mass

DC …….…….Direct Current

DOF ……..…….……...……..Degrees of Freedom

HRI ……..……..……Human-Robot Interaction

PBL …….……...……...…..Project-Based Learning

POSE ……..Position and Orientation

RL …….……...……..Reinforcement Learning

STEM …….……..…Science, Technology, Engineering, Mathematics

Abstract

Sonification is a method to represent data and convey information using sound. Just like the Geiger counter, humans can use sound to better understand complex sets of data that are either unable to be seen or visualized or that are too complex to understand with visual displays. Sonification research and learning have been predominantly conducted at the higher education level. However, as STEM-related programs and activities continue to be increasingly important in secondary school education, it is possible to expose high school students to university-level research through project-based learning (PBL) activities in the classroom. Using a physical snake robot prototype that was built and programmed with low-cost materials, high school students are introduced to the field of sonification and its applications to snake robots. This dissertation demonstrates the feasibility of using project-based learning to teach university level research in secondary school education. Using the sonification of snake robot movement, students learned advanced topics in robotics with the goal of realizing that university level research is accessible and understandable through PBL. This paper will begin by discussing the concept of human-robot interaction, introduce sonification, and give a brief overview of project-based learning. A detailed discussion of how the snake robot prototype was constructed and programmed, an in-depth explanation of the sonification algorithm that was used, and how sonification was taught in a high school classroom using PBL is presented along with student feedback and suggestions for future work.

Chapter 1: Introduction

The "21st Century Skills" movement has been active for the last two or three decades [1]. At the core of this movement is the idea that students leaving education need to be prepared to succeed in the global society that is rapidly changing and developing and one that is increasingly complex and uncertain [2]. Herianto et al. studied this movement using a method they called the "STEM engineering design learning cycle model" at the secondary school level and evaluated the twenty-first century skills known as the 4C's (critical thinking, creativity, communication, and collaboration) [2]. Wagner's book titled "The Global Achievement Gap", suggests that no matter the quality of education students receive, they are still not prepared to enter the global world and are not equipped with essential skills to be successful [3]. No longer are rote memorization skills the key to success. Instead, Wagner offers seven skills which he calls "survival skills", which are critical thinking and problem solving, collaboration and leading by influence, agility and adaptability, initiative and entrepreneurialism, effective oral and written communication, accessing and analyzing information, and curiosity and imagination [3]. Industry leaders that Wagner talked with stated that they were not concerned with a newly hired engineer's technical skills; rather, they most value the ability of someone who can make observations and ask good questions.

How students can be prepared for their lives after education is the task of educators at all levels whether the 4C's model, seven survival skills model, or another model is followed. Focusing on the secondary school level, this paper proposes one methodology for preparation by using project-based learning (PBL) to introduce complex university-level research problems to high

school students which will give them an opportunity to learn how to think critically and problem solve an open-ended question that has no set pre-determined answer. This paper aims to provide an example of this methodology by using a research topic in the field of robotics. Specifically, human-robot interaction is currently a highly researched field in robotics since it is expected that robots will become more social and part of society's everyday life and should be able to adapt to a human's sensory impairments, if they exist. The example of robotics this paper employs is the sonification of snake-robot locomotion which has various uses such as search and rescue operations in hard-to-reach spaces. The remaining part of this section will introduce the three main parts of this dissertation namely, Human-Robot Interaction, Sonification, and Project-Based Learning.

1.1 Human-Robot Interaction

Human-robot interaction (HRI) is a growing and expanding field of study about how to coexist and collaborate with robots that could potentially become a part of our everyday life. Bartneck et al. [4] stress the importance of viewing HRI as a multidisciplinary approach with a team of engineers, psychologists, computer scientists, and sociologists all working together to determine the overall purpose of the robot being designed. It is important to know if the robot's primary design is for functionality, socializing, or both [4]. Robots that are designed to socialize (social robots) have additional psychological design elements that functional robots do not necessarily need. Social robots are expected to interact with humans on a level that is deemed acceptable to societal norms. The robot's design plays an important role when it is expected to interact with humans. Robots that mimic human anatomy can be more inviting to collaborate with as there is a sense of comfort and familiarity during robot and human interaction.

However, there are many applications where a humanoid robot is not the ideal design. Many robots that mimic non-human forms such animals, reptiles, and fish, for example, have important applications that are currently performed by the live versions of these non-human forms [5]. Robotic dogs, for example, have the potential to be used in search and rescue missions where the environment is not safe for a live dog to navigate. Robotics snakes can be used to reach enclosures that are much too small for humans or dogs [6]. Many of these types of non-human animals, evoke certain emotions in humans. For example, a live dog usually confers a sense of comfort while a snake can evoke a sense of fear. For humans to feel safe and secure around robotic animals, a design element that encourages socialization is needed. For example, if a robotic snake is sent on a search and rescue mission, there are psychological design considerations that are needed for the robotic snake to be deemed safe and helpful by the rescued party such as the snake emoting a sense of security.

In addition to evaluating the human response to a robot's physical appearance, humans and robots interact with each other using sensory queues. Many types of interactions are visual such as blinking lights or the position and orientation (POSE) of a robot. For example, when a robot arm picks up an object in an assembly line, a light may illuminate, indicating the arm's direct contact with the object and in a certain configuration. Both indicators provide visual feedback for a robot's status to the human operator [7]. However, many times, the human operator is not physically near the robot, or the human is visually impaired. When these cases occur, there is a need for the robot to continue to successfully communicate its status back to the human so that a productive interaction can be established. For example, robots that are used in search-and-rescue operations can enter spaces that are not accessible by humans or other types of robot configurations. Due to a snake robot's redundant degrees of freedom (DOF) structure, they can

access spaces and adapt to quickly changing kinematic pathways that other types of robots cannot navigate as successfully. To maintain communication with the snake robot, it is possible that non-visual feedback is necessary to inform the human operator about the current state of the robot. Situations where cameras mounted on the robot do not give adequate visual display feedback due to extreme dark lighting conditions or a dense environment, would require other types of sensory feedback such as auditory displays. These could be in the form of tones or pieces of music that can indicate the location and POSE of a snake robot that is out-of-sight.

1.2 Sonification

One technology that employs auditory displays and can be used to understand snake robot movement is a methodology called sonification. Specifically, sonification is a method to represent data sets and convey information using non-speech sound [8]. Humans can use sound to better understand complex sets of data that are either unable to be seen or visualized or that are too complex to understand with visual displays. Processing and analyzing large datasets are skills that are mastered by the researcher that is accumulating the data usually through experimentation or observation. How processed data is presented to experts and novices is a skill that can manifest in various forms. Most commonly, the output of the processed data is presented using visual displays such as graphs and tables. These mediums have been incredibly useful at presenting data and are fully supported with printed text. A more uncommon and underexplored [8] method is to use auditory displays to interpret and communicate processed data. According to Kacem et. al, sonification allows users to "experience data by listening" [9].

According to Herman, data is sonified if it meets the necessary and sufficient conditions of data-dependency, objectivity, systematicness, and reproducibility [8]. Sonification techniques

include audification, auditory icons, earcons, parameter-mapping, model-based, and wave-space [7] [9].

Robotic movements such as locomotion and POSE can generate a significant amount of sensor data that can either require post processing after the data is acquired or on-demand (or real-time) processing while the movement is occurring. Visual displays, such as graphs, can aid in summarizing data as a post-processing method that allows the user to spend time analyzing the results. However, there are instances when the user needs to understand and analyze the data in real-time while watching the robotic movement or in the absence of having the robotic movement available to view. In these cases, it may not be possible to effectively view data using visual displays while concurrently watching robotic movement or there may be an absence of a human's ability to view visual displays. Sonification can provide a solution to these visual display issues and can even enhance the ability to understand and detect certain movements in real-time.

Current work in sonification of snake-robot movement, while limited, is being researched by Kacen et. al. using wave-space sonification to model the folding and unfolding of protein molecules using snake robot movement with hyper-redundant degrees of freedom [29]. In their paper, protein linkages are modeled as snake-robot links with their movement tracked using a sonification methodology that causes a unique sound file to play when a certain configuration of the protein linkages is achieved. Multiple, sequential configurations will chain unique sounds together with the intent to play a resulting sound file that will reveal the protein's folding/unfolding pattern. The snake robot is used as a model having known equations of motion that can be applied to the protein movement.

Using sonification as a means for human-robot social interaction is explored by Schwenk et al which uses real time robot configurations to generate a non-speech audio sound which is used

as a social cue for communication [28]. According to Schwenk, audio is an acceptable social cue and is a type of sensory stimulation that is welcomed by humans, thus suggesting that sonification can be used as a means for humans and robots to effectively communicate with each other.

1.3 Project-Based Learning Implementation

"Learning by doing" [10] is an idea from educational reform pioneer, John Dewey, in his 1897 essay, "My Pedagogical Creed". In his essay, he expressed his view that "the teacher is not in the school to impose certain ideas or to form certain habits in the child but is there as a member of the community to select the influences which shall affect the child and to assist him in properly responding to these" [11]. Over the years, education has refined his beliefs into a methodology known as project-based learning (PBL). Essentially, PBL is an educational model that challenges students to identify real-world issues and problems to create solutions that can have meaningful and lasting impact on society [10]. Instead of students sitting in class and learning by rote memorization, the goal of PBL is for students to have agency and to learn how to become critical thinkers, resilient problem solvers, and advocates for change to make the world a more sustainable place to live by identifying problems within a chosen community and by working in teams to create solutions to those problems. Students develop an ownership and commitment to their work and a sense of responsibility to see their work succeed [12]. The Buck Institute for Education is one example of a consortium of educators promoting the education of teachers on how to engage with and conduct effective PBL activities either as the focus of the whole class or as a short-term class activity [13].

Project-based learning (PBL) in a classroom can take on two forms. One form is the entire duration of the class is project-based which typically means that the class does not have traditional assessments such as quizzes or tests which are used to evaluate students' mastery of the material

that was taught. Instead, mastery is evaluated based on a method known as standards-based grading that evaluates a student's success in defined areas of the project. These types of classes give students the agency to determine their own project, usually in a small, team-based environment. Some daily or weekly homework may be given to strengthen student background knowledge or skill level to support the project. Another form of PBL is a short activity within a traditional class. These activities can happen once or multiple times during a semester or school year and give students an opportunity to work in small teams on a focused project that supports the topic of the class and can span a few days to a few weeks in duration. According to the Buck Institute for Education, there are seven essential design elements that should be followed in order to create a "Gold Standard PBL"[27]. Below, Figure 4-1 illustrates these design elements.

Figure 1-1: Gold Standard PBL – Seven Essential Project Design Elements

Project-based learning has been the subject of classrooms both in higher education and secondary school education across a variety of disciplines for many years. Using robotics for

student learning in a PBL environment was first pioneered by Seymour Papert who is credited for using robots to create novel, experiential learning environments in a project-based setting [12]. According to Papert, this environment "helped students in ordered logical thinking, self-discipline, responsibility, and creativity" [12]. He is credited as being the "Father of Education Robots" [14] and invented the first education robot called the Turtle along with a programming language called Logo. This invention was the result from his philosophy which Catlin coined "Papert's Paradigm" which promotes forming constructionist cultures in schools that encourage students to both fail and succeed with equal importance and which also promote students to be creative, think, explore, and love to learn all the while developing their socialization and community-building skills [14]. He saw educational robots as a vehicle to realize his philosophy where students are supported by teachers as they develop their own knowledge from their surrounding culture and environment. The Turtle robot was a piece of technology that students could interact with as it moved with their body as they moved. Over the years, the education system has increasingly used the combination of project-based learning and robotics as a model to help students understand complex, unanswered issues in a more deterministic, predictable way.

For example, Mohammadi and Heilman conducted a joint-university, cross-disciplinary PBL experience for college students to learn about protein folding and unfolding through the lens of robotics and how snake robots articulate [15]. By bringing two seemingly different subject matters together, namely biochemistry and robotics, students were able to experience, first-hand, the importance of how one discipline can help aid the understanding of a problem in another discipline. Designing this cross-disciplinary idea as a project-based learning experience provided the students with an active learning educational model that created problem solving opportunities and appreciation for collaboration among peers at both universities [15].

While the above examples illustrate the use of PBL in higher education, project-based learning is also a teaching method that is used in secondary schools as well. While there are many examples of PBL activities, this paper focuses on the use of PBL to expose university-level research to high school students in a manner that is manageable, accessible, and engaging. By doing so, students are introduced to STEM topics that are typically reserved for higher education; however, exposing students to these topics in high school can generate increased interest in pursuing a STEM-related field in college.

Specifically, this paper focuses on a project-based learning activity about the sonification of snake robot movement that was delivered to and executed by a group of high school students at Hopkins School in New Haven, Connecticut. Sonification research and learning has been predominantly conducted at the college level. However, as STEM-related programs and activities become increasingly important in high school, this paper demonstrates the possibility to expose high school students to college level research through project-based learning (PBL) activities in the classroom using methods that are accessible to high school students but still maintain the rigorous academic detail that is needed for college-level material. From this, the student can gain an appreciation of how college-level research be used in real world engineering applications.

The objective of this thesis is to deliver a suggested solution for teaching higher education robotics subjects – specifically sonification of snake robot movement - at the secondary school level using Project-Based Learning. This thesis will cover the specifics of the snake robot in Chapter 2, explain the software of the simulation in Chapter 3, the execution and results of the PBL activity in a high school classroom in Chapter 4, and conclude with results and future work in Chapter 5.

Chapter 2: Snake Robot Construction, Programming, and Data Collection

Snake robots have many applications due to their ability to navigate confined spaces such as in search and rescue operations [6]. Many of these applications require snake robots to enter areas that are not visible to humans. Understanding the movement and locomotion of snake robots, without visual displays, with sonification provides valuable assistance to humans. To develop a model that could be used to further explore and teach the importance of sonification and its applications in robotics, a physical snake robot prototype was built before the PBL activity conducted in the classroom. Additionally, building the robot outside of the classroom allowed the teacher to assess the feasibility of each student or team of students building their own robot and collecting POSE data in class given the amount of time that was dedicated to the PBL activity. This chapter will explain how the prototype was constructed over the course of two iterations, how the robot was programmed, and how movement data was collected.

2.1 Snake Robot Prototype

A prototype of a snake robot was constructed using simple materials that could realistically model an actual snake's movement and interaction with its environment. Construction was based on having an affordable prototype model that could be implemented in all high schools nationwide and even internationally. A bill of materials is provided in the last section of this chapter. Tagboard (32 oz or 907 g) was used for each piece of the snake body part (or linkage). Figure 2-1 shows the design of a linkage which is 7.5 cm wide by 10 cm long with extra space left at each end to attach additional linkages. A template is provided in Appendix A. The snake linkages were modeled after

the snake robot in this video link [16]. As can be seen in Figure 2-1, the linkages have a curved shape so that the oscillatory movement is not impeded by any square edges on the linkages. To reduce friction between the tagboard and support surface, two roller beads were inserted with a pin in the middle of each linkage as can be seen in Figure 2-1. The pin was attached to the tag board with glue. The green dot indicates that the linkage is directly behind the head and is used for data tracking which will be explained in a later section in this chapter.

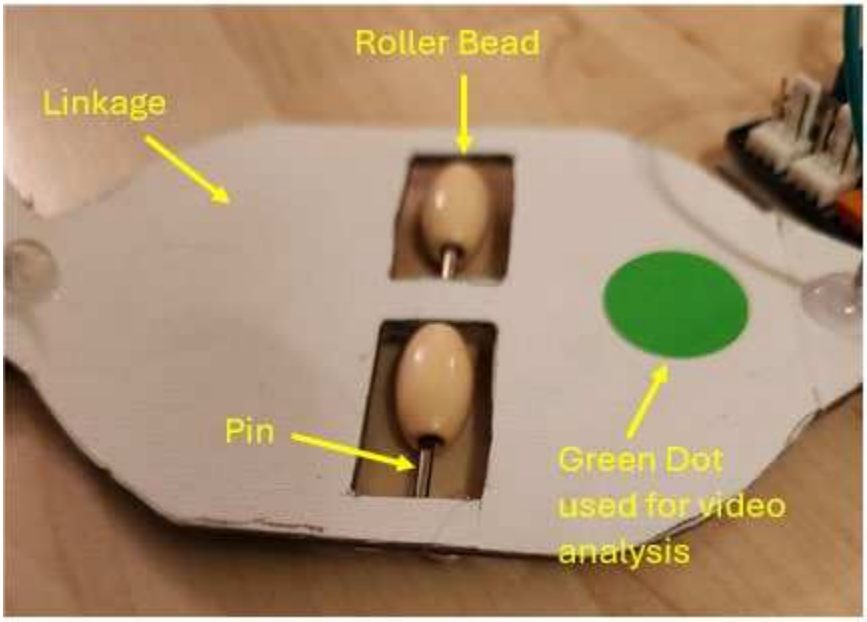


Figure 2-1: One Snake Robot Linkage

Five linkages were assembled and connected with small stick pins and fastened with hot glue such that the robot could freely move side-to-side and locomote and oscillate at the same time. Figure 2-2 shows the entire robot with five linkages. The entire length of the robot, including the head, was 38 cm. The design of this robot is expandable as more linkages can be added if needed; however, five linkages were chosen as the minimum number to achieve an oscillatory movement that is sinusoidal. The colored, circle dots (green, orange, pink) are used when sensor movement data is created.

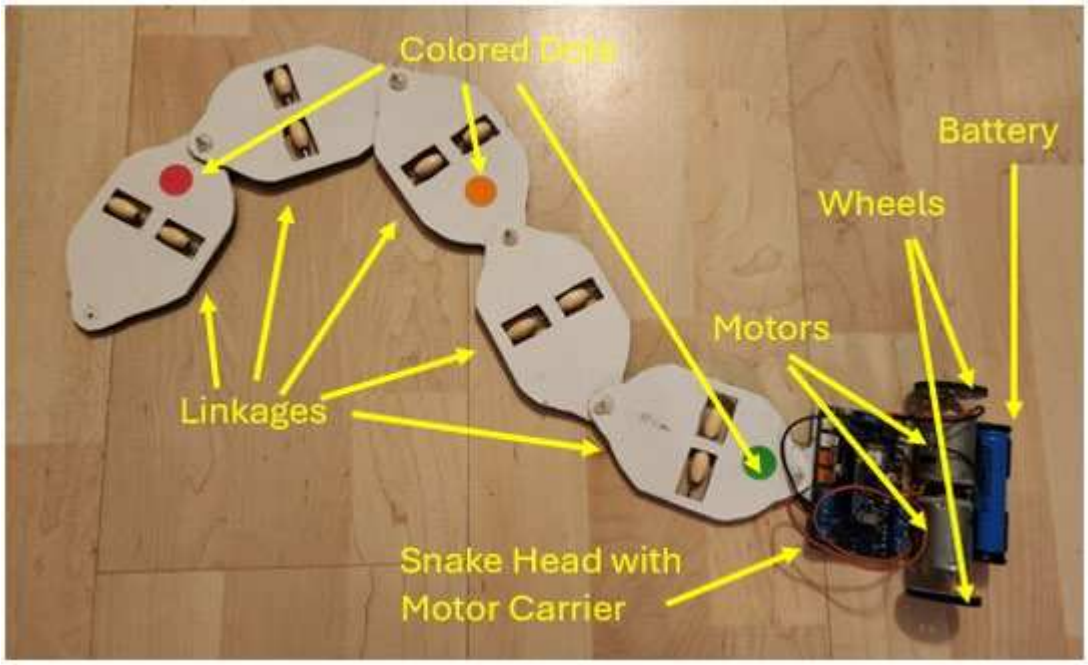


Figure 2-2: Snake Robot with Five Body Linkages

The leading linkage, also known as the snake head, was also 7.5 cm by 10 cm but was a rectangular cutout for all the electronics, motors and battery to fit in one location which resulted in greater inertia in the front so the head could more easily locomote its entire body. The template for the head can also be found in Appendix A. The size of the head was chosen so that it is close in size to the rest of the snake.

The electronics were powered by a 4 V rechargeable lithium-ion battery that was seated with a holder on an Arduino Nano Motor Carrier as can be seen as a closeup in Figure 2-3. The battery can either be charged externally with a charger or it can be charged with the embedded code in the Arduino software which can be found in Appendix B. Overall, the snake robot was controlled with an Arduino Nano 33 IoT microcontroller with 256 Mb of memory storage [17] as can be seen in Figure 2-4 with the pinout diagram shown in Figure 2-5.

Figure 2-3: Closeup View of Lithium-Ion Battery

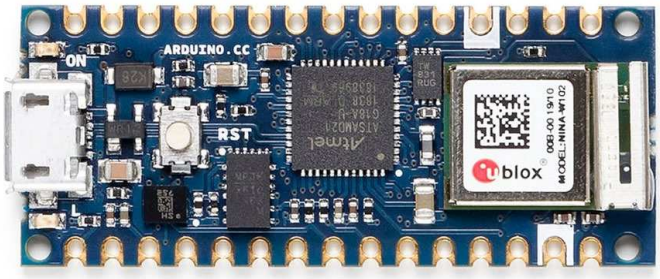


Figure 2-4: Arduino Nano33 IoT [17]

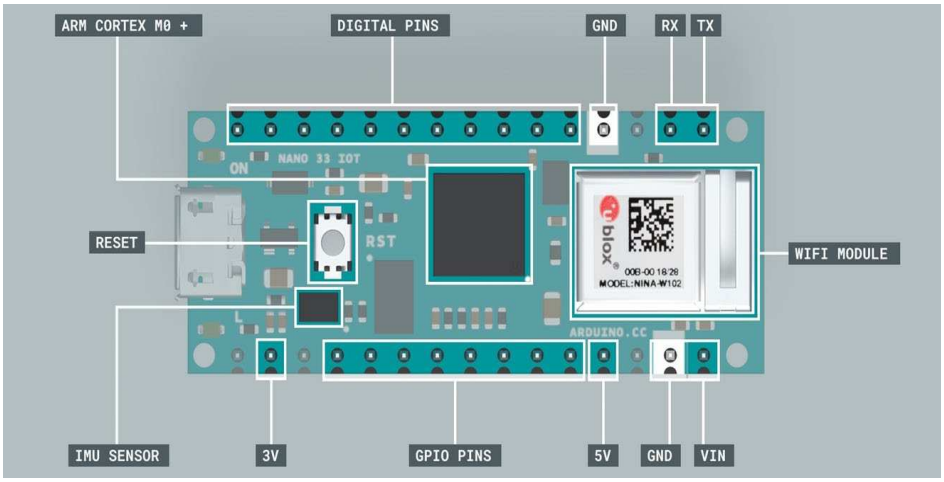


Figure 2-5: Pinout Architecture for Nano IoT 33 [17]

The Nano was inserted into an Arduino Nano Motor Carrier [18] which was secured onto the top of the head tagboard using hot glue; however, small screws could also be used. A diagram for the Motor Carrier without the Nano installed can be seen in Figure 2- 6, while Figure 2-7 shows the Nano 33 IoT installed in the middle socket of the Motor Carrier. The Motor Carrier was chosen for its compact form factor, ability to accommodate a small microcontroller, as well as having motor controllers integrated into the board, which manage the current draw between the Nano IoT and the motors. The pinout diagram for the Motor Carrier can be found in Figure 2-8.

Other microcontrollers that are available, such as the Raspberry Pi Pico and the ESP32 are also small form factors but require a breadboard and separate motor controllers which would require additional space on the snake head than is available. A microcomputer such as a Raspberry Pi is not feasible since it is a larger circuit board without motor controllers which would result in not having enough space on the snake head to mount all the electronic parts.

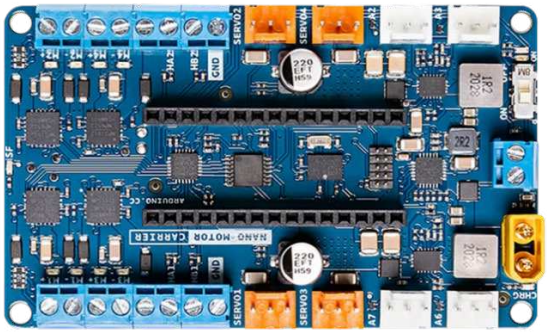


Figure 2-6: Top View of Arduino Nano Motor Carrier [18]

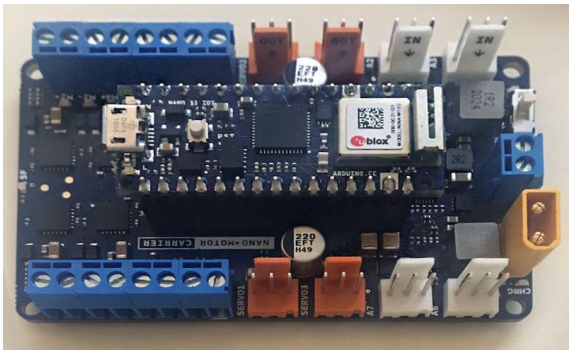


Figure 2-7: Top View of Arduino Nano Motor Carrier with Nano IoT Installed

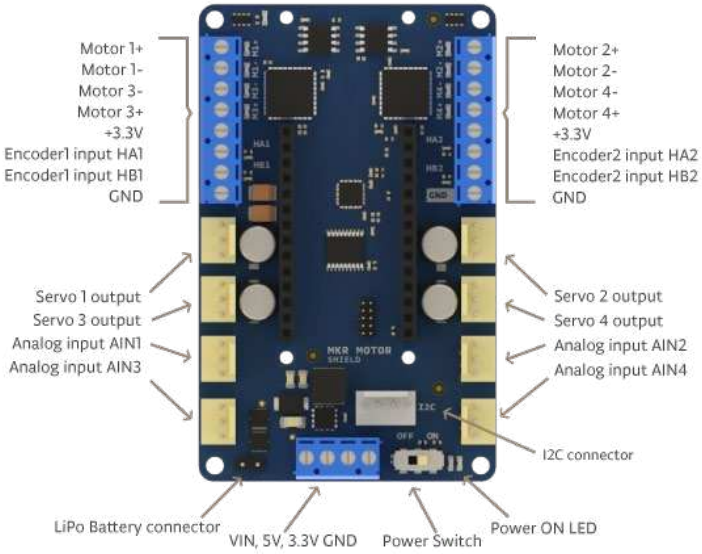


Figure 2-8: Pinout Diagram for Arduino Motor Carrier [19]

Motor selection was restricted to a size that could fit on the snake head yet still be powerful enough to move the prototype. To determine an effective snake head design, two different types of DC (direct current) motors were selected for experimentation resulting in two design iterations. The first design used a Type 130 Miniature DC motor (Motor 1) and the second design used a Type 280 Micro DC motor (Motor 2). The same tag board template for the snake head was used for each design. A comparison of the specifications of each motor can be found below in Table 2-1.

Table 2-1: Comparison of Motors Used for Two Iterations of the Snake Head

	Motor 1	Motor 2
Type	130 Miniature DC	280 Micro DC
Size	15mm x 20mm	24mm x 31mm
Voltage	1V – 6V	3V – 12V
Reference Current	0.35A – 0.40 A	0.25A – 0.50A
Speed	16000 RPM at 3V	13500 RPM at 3V
Shaft Diameter	2.0 mm	2.0 mm
Shaft Length	9.0 mm	9.5 mm
Mass	17.5 g	208 g
Power Consumption	1.4 W	1 W

Figures 2-9 and 2-10 show the diagram of Motor 1 and Motor 2, respectively.

Figure 2-9: Type 130 Miniature DC Motor (Motor 1) Used for First Iteration [23]

Figure 2-10: Type 280 Micro DC Motor (Motor 2) Used for Second Iteration [24]

Figure 11 shows the circuit diagram for the snake head. The circuit consisted of the Nano 33 IoT, the Arduino Motor Carrier, two motors, and a rechargeable battery. Sensors and encoders were not needed for this PBL activity. The intent was to keep the circuit simple for secondary school students, especially if students do not have any STEM background. For each iteration, the pair of motors were connected to two of the motor controller ports of the Motor Carrier using wire

as can be seen in Figure 2-11. Specifically, motor port +/- M3 and +/- M4 were used but any combination of +/- M1-4 pairs could be used. In general, motors require a larger amount of current to operate than the Nano IoT can supply. Since the Nano board does not have the necessary circuitry to directly power and control motors, the motor controllers provided the necessary current management between the motors and the Nano. Since the Nano can only supply a maximum of 0.06 Amps, the motor controllers stepped up the current received from the Nano to the required operational level for the motors.

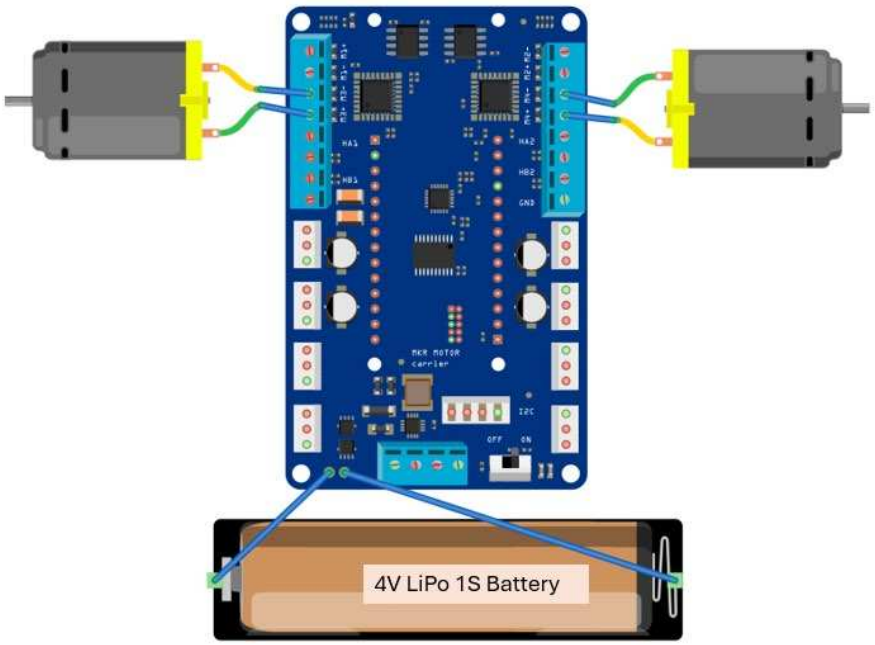


Figure 2-11: Snake Robot Circuit Layout using Fritzing (without the Nano installed)

For the first iteration, two Motor 1 motors were mounted on the sides of the head linkage at an angle such that the ends of the axles of each motor contacted the supporting surface, as can be seen in Figure 2-12. Small eraser heads, as can be seen in Figure 13, were pressed on to each axle to increase static friction between the two surfaces so that the robot could move along the smooth surface.

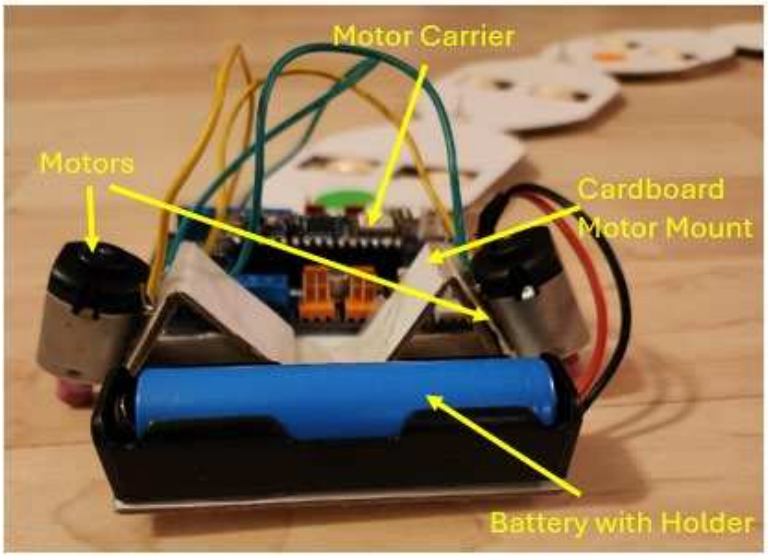


Figure 2-12: First Iteration of Snake Robot Head.

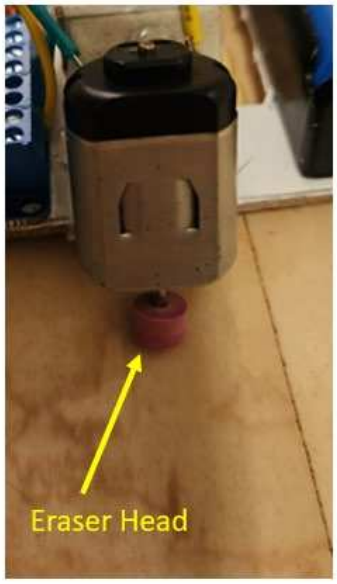


Figure 2-13: Close Up of Eraser Head on Axle of Motor 1

The second iteration used the larger Motor 2 motors, as shown in Figure 2-14, which were again mounted on either side of the head linkage but instead were mounted parallel to the support surface. Plastic wheels having a diameter of 3 cm were 3D printed and pressed onto the axles along

with black bands installed on the rims to increase static friction between the wheels and the floor, as can be seen in Figure 2-15.

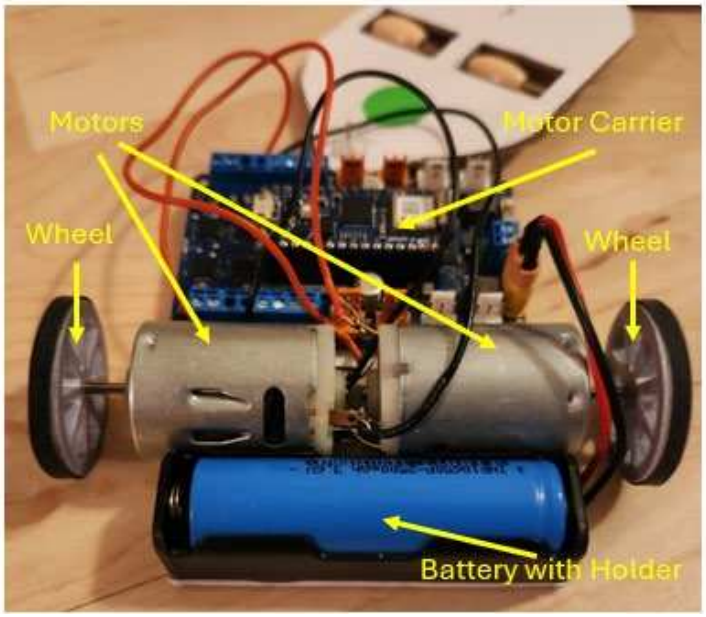


Figure 2-14: Second Iteration of Snake Robot Head

Figure 2-15: 3D Printed Wheel – Left diagram shows wheel face with 3 cm diameter and the diagram on the right shows the black band on the wheel's rim.

A comparison summarizing the similarities and differences between the construction of each iteration for the snake robot head can be found below in Table 2-2.

Table 2-2: Comparison Between First and Second Iteration of Snake Robot Head Construction

	First Iteration	Second Iteration
Motors	Motor 1	Motor 2
Motor Mounts	Angled using thick cardboard	Flat with axle parallel to floor
Type of End Attachment to Axle	Eraser from wood pencil	3d printed wheel with rubber band
Power Consumption	2.8 W (for both motors)	2 W (for both motors)
Snake Head Base	Same thick cardboard and same shape.	
Battery (Mount + Location)	Same mount, battery, and frontal location.	
Embedded System	Same Arduino Motor Carrier	

Both iterations were programmed with the same code, which is described in the next section, and the same surface was used to test the robot movement.

2.2 Snake Robot Programming

The snake robot was programmed in C using the Arduino IDE. The code, which is listed in Appendix B, was written to mimic snake motion. More specifically, the goal was to develop code so that the robot would oscillate while locomoting forward at the same time while taking advantages of ground friction forces both sideways and longitudinally, according to prior research in the area of snake locomotion by Hamidreza et al [25].

Since each motor was mounted opposite each other, the motors were wired in opposite directions for the snake to move forward. Three functions were written to assess which motor movement more closely mimicked snake robot locomotion and each function was tested in the loop() section of the program which continuously ran each function. Below, Table 2-3 illustrates how each potential function operated the motors.

Table 2-3: Three Functions Written to Assess Proper Snake Motion

Function	Operation
oldMove()	One motor turns on at 25% speed for 350 msec and then turns off. The other motors turn on at ¼ speed for 350 msec and then turns off.
simpleMove()	One motor turns on at 50% speed for 1 sec then turns off. Both motors turn on at ½ speed on for 50 msec then off for 500 msec. The other motor turns on at ½ speed for 1 sec then turns off.
snakeMove()	Both motors ramp up to 40% motor speed. After 15 msec, both motors ramp down to 0% motor speed.

It was found that the snakeMove() function best represented both the locomotion and oscillation of a snake. The other two functions either oscillated only (oldMove()) or locomoted too much without the snake's linkages oscillating (simpleMove()).

Once movement function was decided on, each iteration of the snake robot was programmed, and movement was video recorded using an Android phone at 30 msec frame rate. Each video was taken from a top view with the phone parallel to the path of the snake's movement, otherwise known as a “bird’s eye view”. The video using the snakeMove() function using Motor 1 can be found with this link [20] and with Motor 2 can be found with this link [21].

2.3 Data Collection

Two-dimensional position data of the snake's movement was captured by processing the video file with PASCO Capstone software [22]. Using the software's Video Analysis feature, the green dot on the snake's first body linkage, which represented the snake's center-of-mass (COM), was tracked by mouse clicking on the same location of the green dot as each frame of the video advanced. The first snake robot design used in Figure 2-16 shows the Capstone screen after clicking was completed. The blue X's denotes one click for each frame of data. It was important to click on the same location of the green dot COM for each frame to accurately track the snake's movement. The indicator in the diagram marked "60 cm Ruler" denotes a 60 cm frame of reference length for the video analysis software to use as it created the sensor position data. Since the data was captured using mouse clicks instead of an actual electrical sensor, noise that would normally be introduced into the data was minimized; therefore, post-processing of the data to eliminate noise was not needed.

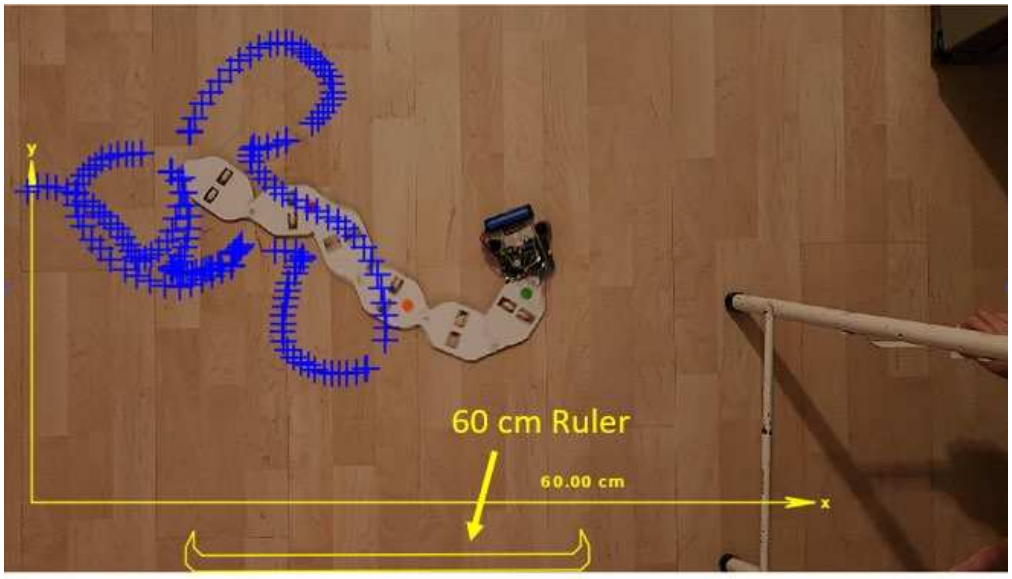


Figure 2-16: Results for Clicking on the Green Dot for Each Frame

The blue X's generated two-dimensional position data which was graphed in Pasco Capstone to create a visual display of the snake's movement which is illustrated in Figure 2-17. The graph is created using the y-position data on the y axis and the x-position data on the x-axis.

Figure 2-17: Graph of Vertical (yPosition1) and Horizontal (xPosition1) POSE Using Motor 1

The second snake robot design iteration was video recorded and processed through PASCO Capstone's Video Analysis feature by again tracking the green dot of the first body linkage. Figure 2-18 shows the results for clicking on the green dot and Figure 2-19 illustrates the horizontal and vertical position data. The indicator in the Figure 2-18 diagram marked "90 cm Ruler" denotes a 90 cm frame of reference length for the video analysis software to use as it created the sensor position data.

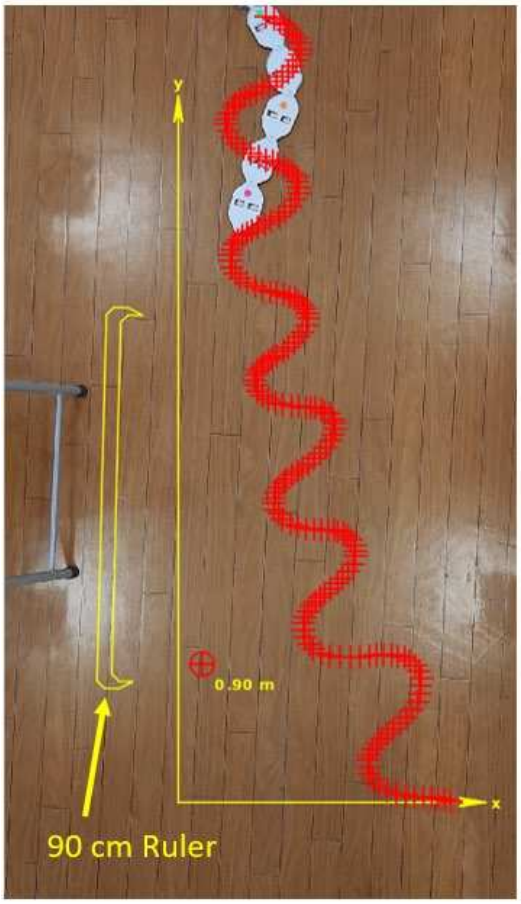


Figure 2-18: Results for Clicking on the Green Dot for Each Frame

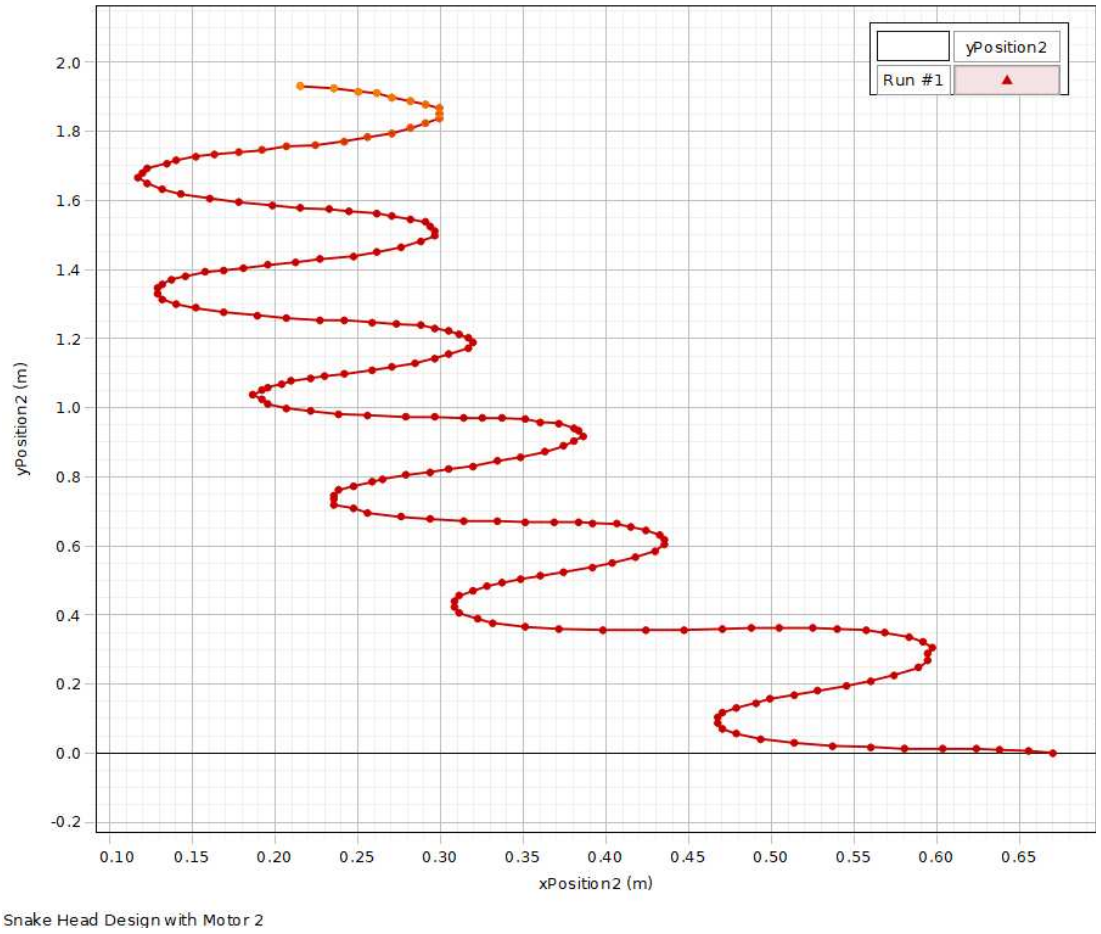


Figure 2-19: Graph of Vertical (yPosition2) and Horizontal (xPosition2) POSE Using Motor 2

While position data was required for the sonification algorithm, it should be noted that PASCO Capstone can also generate velocity and acceleration data in each dimension as well for the time that the snake moved. This could be an additional module to develop in a PBL activity.

2.4 Results

Both snake robot designs produced movement results; however, the different motors used for each iteration resulted in different locomotion and oscillation movements even when the same code was used. Table 2-5, below, summarizes the differences between the first and second iterations.

Table 2-4: Summarizing the Differences Between Both Iterations

	First Iteration	Second Iteration
Motor Used	Motor 1	Motor 2
Axle Attachment	Rubber eraser	Wheel plus band on rim
Observations	Rubber erasers tended to act as a pivot which prevented the robot to easily locomote forward. Even though code was turning the motors on and off for oscillation, the erasers used that motion to only move in a circular pattern of "S"like movement.	The black band on the rim provided just enough static friction between the wheel and floor to allow forward motion while still allowing the robot to oscillate because of the on/off pattern of the motors.
Type of Movement	Mostly oscillations and very little locomotion, keeping the snake essentially in one x-y location.	More sinusoidal movement such that the snake would have a forward displacement but still oscillate.
Quality of Axle Attachment	The rubber erasers would fall off periodically	The wheels stayed secure to the axle and the black bands remained on the rim.
Ability to Video Movement	Easy since the snake robot oscillated back and forth in one place.	More challenging since the snake robot move quickly out of the frame of the phone
Mimicking Snake Movement	Very little locomotion so not the best model for snake movement	Better model for snake movement since motion was more sinusoidal.

The floor that was used was a faux wood floor type material. The wheels from iteration two responded well to this type of floor material. The snake robot was also run on different types of flooring, and it was found that floors that are waxed have an adverse effect on the POSE of the robot as it is running. Some floors would not allow the robot to locomote even with the wheels as the axle attachments. The differences in the robot performance based on the flooring could be

taken care of in the Nano code by potentially changing the motor speed or the oscillation pattern; otherwise, a stronger motor may be needed, or the axle attachment (such as the wheel) could be modified by replacing the black band with a band that has a lower static friction component.

The most challenging part of the construction was mounting the motors next to the electronics such as the motor carrier. The size of the head was chosen to be the same size as the other linkages to model a real snake's body proportions. However, once the electronics were mounted, there was little space to mount the motors. The first iteration tried to save space by mounting the motors at an angle, but the eraser head axle attachment did not allow the robot to easily locomote. With the angled motors, the wheel attachments could not be mounted. Fortunately, there was just enough space to mount the slightly larger motors in the horizontal position which allowed the wheels to be used.

2.5 Bill of Materials

The snake robot prototype was designed to be affordable so that schools in the United States and internationally can afford to incorporate this technology and research into their classrooms. Table 2-4 lists all the parts that were used and their corresponding cost.

Table 2-5: Bill of Materials for Snake Robot Prototype

Part	Qty	Cost ($ US Dollars)
Tagboard	5	2.00
Roller Beads	8	0.50
Pins	8	0.25
Motor Carrier + Battery	1	80.00
Nano IoT 33	1	28.00
Wire	4	2.00
Motors	2	5.00
Wheels/Bands or Rubber Eraser Tip	2	1.00
Total		118.75

Since the Motor Carrier + Battery and the Nano IoT 33 are the most expensive parts in the total cost ($108.00), it is possible for a different type of embedded system to be used such as an ESP32 microcontroller, mounted on a small breadboard, which has a similar form factor as the Nano IoT 33. Instead of the Motor Carrier, a separate motor controller could be used such as the L298N Motor Driver Controller board. Together, their total cost is approximately $20.00 which would save $88.00 towards the total cost of the prototype. Both replacements are separate parts and would need to be mounted alongside each other on the snake head and then wired accordingly.

Chapter 3: Simulation and Sonification

After the snake movement was recorded and tracked in Pasco, the x and y movement data was saved in a .CSV file. This data was captured before the PBL activity was presented to the class to assess the feasibility of building a snake robot and acquiring and POSE data with the entire class; therefore, it was saved for later use in the next revision of this PBL activity. To provide an introductory lesson about sonification and its application to snake robots, a simulation [26] that used sonification of snake robot movement was used to motivate the class to develop an appreciation and understanding of the mechanics of sonification. This simulation did not require the captured data from Chapter 2 to be imported into the software, instead the simulation used data that was created specifically for the three types of snake gaits that were used by the students. This chapter explains the code for simulation in detail if part of the PBL classroom activity requires students to create their own code or learn how to improve the documentation of the existing code.

3.1 Parameter-Mapping Sonification

There are several different types of sonification methods. The method used for this paper was Parameter-Mapping Sonification (PMSon) which involves the association of data with auditory parameters so that data is understood through hearing sound rather than seeing a visual display via the use of a mapping transfer function. For this paper, non-speech sound was used to understand the POSE of the snake-robot's movement. Figure 3-1 shows the general guidance provided by Herman for creating a PMSon model as there are many potential factors to consider to correctly sonify data [8].

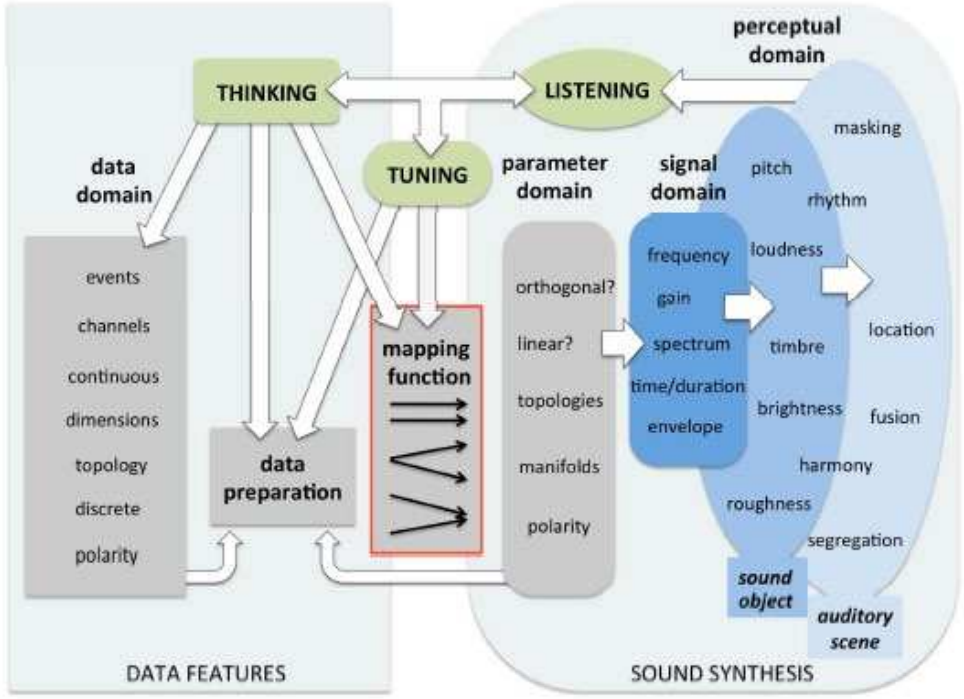


Figure 3-1: General Parameter-Mapping Workflow [22]

The three major parts to this model are Thinking, Tuning, Listening. Thinking involves collecting and preparing the data. Tuning requires the use of a transfer function to map data to sound elements. Listening involves interpretation and understanding of the data through an auditory display.

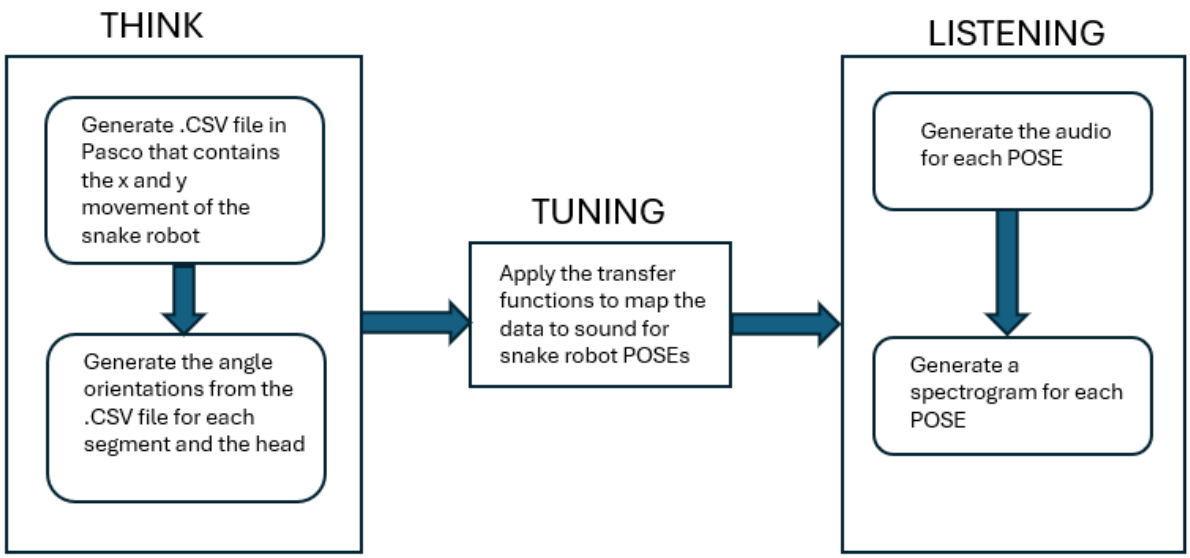


Figure 3-2: Parameter-Mapping Workflow for Snake Robot Sonification

Using Herman's guidance from the general model, Figure 3-2 illustrates a simplified version of used to create the snake robot sonification. The elements of the workflow were executed with MATLAB files that are presented and described in the remaining parts of this chapter. The sonification algorithm that was used in the simulation was written by Dr. Alireza Mohammadi from the Robot Motion Intelligence Lab at the University of Michigan-Dearborn [26]. The algorithm consists of MATLAB files: DAFXsnakeGaitGen2.m and DAFXsoundSweep.m. Both files are listed in their entirety in Appendix C and D, respectively. To gain a comprehensive understanding of the mechanics of each file, the major sections of each file are described step-by-step in the following sections of this chapter.

3.2 Snake Gait Generation

The first file that will be explained is DAFXSnakeGaitGen2.m. This file allows the user to select from one of three gait patterns, namely, Concertina (inch-worm-like movement), Lateral

Undulation (wave-like slithering) or Side Winding (diagonal movement). Figure 3-3 shows the first part of the code that prepares the simulation environment and, as an example, sets the gait pattern to 3, Side Winding, with a simulation time, Tfinal, equal to 1.5 seconds.

```
21  close all; clc; clear;
22
23  % Set the gait pattern: 1 for Concertina, 2 for Lateral Undulation,
24  % 3 for Side Winding
25  gaitPattern = 3; % <-- Set this to the desired pattern
26
27  % gaitPattern = ...
28  % input("Choose the snake robot gait pattern (1 for Concertina, 2
29  % for Lateral Undulation, 3 for Side Winding): \n");
30
31  % Simulation time step
32  delT = 0.02;
33  % Simulation final time
34  Tfinal = 1.5;
35
36  switch gaitPattern
37      case 1
38          filename = 'snakeConcertina.gif';
39          fileNameSound = 'snakeConcertina.wav';
40      case 2
41          filename = 'snakeLatUndul.gif';
42          fileNameSound = 'snakeLatUndul.wav';
43      case 3
44          filename = 'snakeSideWind.gif';
45          fileNameSound = 'snakeSideWind.wav';
46  end
47  %
48  if isempty(filename)
49      filename = 'snake_gait_animation.gif'; % Default filename if none is provided
50  end
```

Figure 3-3: Select Gait Pattern and Simulation Time [26]

The next piece of code, shown in Figure 3-4, initializes the pose of the head of the robot (x0, y0) and the number snake robot segments, n, each having a length of l units. Line 56 creates a 2 x (n+1) zero matrix, r, which initials the segments' position vector. Lines 59-69 set up the graphical display environment.

```
51  % Initial position and configuration of the snake robot
52  x0 = -100; y0 = 0;
53  n = 45;    l = 2;
54
55  % Initialize the position vectors for the snake's segments
56  r = zeros(2, n+1);
57  r(:,1) = [x0; y0];
58
59  % Create the figure for displaying the simulation with specified axis limits
60  figure('Color', 'w'); % Set the background color to white during figure creation
61  set(gcf, 'WindowState', 'maximized'); % Maximize the figure window
62  hold on;
63  grid on;
64  axis equal;
65
66  % Use axis limits that will contain the snake for all gait patterns
67  % You may need to adjust these limits based on your specific gaits
68  axisLimits = [-200, 200, -150, 150];
69  axis(axisLimits);
70
71
72
73  % Initialize the segment angles array
74  th = zeros(1, n);
75  a = 30; % Counter used in the update step of the gait equations
76
77  % Initialization
78  for i=1:length(th)
79      th(i) = gait(i, gaitPattern, a);
80  end
```

Figure 3-4: Initialize Snake Robots POSE and Setup Visual Display [26]

Starting with Line 74, the segment angles are initialized by creating a row vector, th, having n elements. The for-loop uses a function called “gait” which generates the initial angles of each segment based on the type of gait pattern chosen. The “gait” function will be described later in this chapter and is a part of a set of helper functions that are called throughout the code.

3.3 Snake Gait Animation

The next piece of code in Figure 3-5 is the main loop and simulates the movement of the robot based on the gait pattern chosen and generates an animated visualization of its motion as a GIF file. Starting with the highest level for-loop in Line 82, the robot’s current position and angles for the current time step, delT, is calculated until the simulation time, Tfinal is reached. The next two for-loops starting at Line 84, calculates the position of each segment i, starting with segment

2 (one segment after the head). The inner loop calculates the cumulative offset from the head using trigonometry to determine the horizontal offset ($\cos\theta$) and vertical offset ($\sin\theta$) with the segment length, l, scaling the offset. Line 89 shows that the position of segment i, in the array, r, is found by adding the cumulative offset to the head position. The head's POSE is found using the headShape helper function which calculates the coordinates of a polygon that represents the snake's head, oriented according to the head angle. This function will be explained later in this chapter. The elements in the headPos vector in Line 93 refer to the position of the snake's head and the cumulative angle, sum(th), of the entire snake which determines the orientation of the head.

Starting with line 97, the snake body segments start to be drawn, and the head is drawn based on the coordinates calculated by headShape. The snake image is captured as a .GIF in line 103, saved to a file and the image is deleted. To simulate snake movement, the segment positions are incremented by 1 in Line 120 and the helper functions, "updateCounter" and "updateGait" are run so that new head POSE and segment orientation can be calculated. When this entire process is run in a for-loop, the movement of the snake is successfully animated for the "Tfinal" time duration.

```
% Main simulation loop
for t=0:delT:Tfinal
    % Update coordinates
    for i=2:length(r)
        temp = [0;0];
        for j=1:i-1
            temp = temp + l*[cos(sum(th(1:j)));sin(sum(th(1:j)))];
        end
        r(:,i) = r(:,1) + temp;
    end

    % Compute the head's position and orientation
    headPos = [r(:,n+1); sum(th)];
    [headX, headY] = headShape(headPos);

    % Draw the snake's body
    snake = drawSnakeBody(r, n, l);

    % Draw the head
    head = fill(headX, headY, [0, 0, 0]);

   % Capture the frame
    drawnow; % Ensure the figure updates have been processed
    frame = getframe(gcf);
    im = frame2im(frame);
    [imind, cm] = rgb2ind(im, 256);

    % Write to the GIF file
    if t == 0
        imwrite(imind, cm, filename, 'gif', 'Loopcount', inf, 'DelayTime', delT);
    else
        imwrite(imind, cm, filename, 'gif', 'WriteMode', 'append', 'DelayTime', delT);
    end

    % Remove the snake for the next frame
    delete(snake);
    delete(head);

    % Update positions for movement
    r(:,1:n) = r(:,2:n+1);

    % Update the gait
    a = updateCounter(a, gaitPattern);
    th = updateGait(th, n, gaitPattern, a);
end
```

Figure 3-5: Main Simulation Loop to Create Animation [26]

The three .GIF files that are generated are shown in Figures 3-6 – 3-8 below. To use the desired gait, the “gaitPattern” variable was set to 1, 2, or 3 on Line25 in Figure 3-3. Figure 3-6 shows the concertina movement with a periodic oscillation and straight-line locomotion movement. Figure 3-7 shows the lateral undulation movement with a constant oscillation

movement and minimal locomotion. Figure 3-8 shows the side winding movement as the snake moves from side-to-side along with forward locomotion.

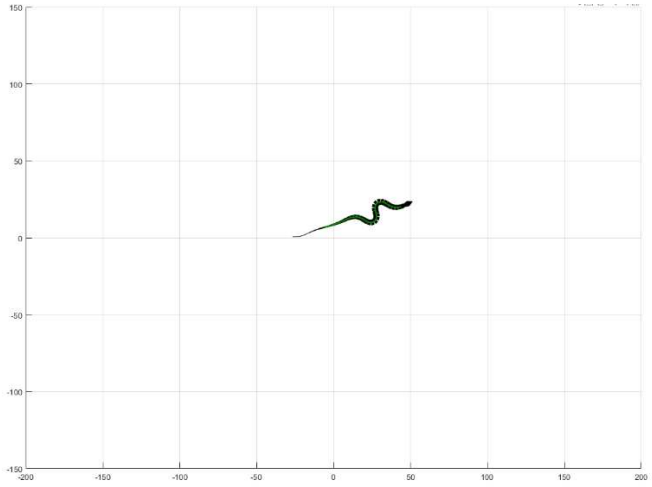


Figure 3-6: Snapshot of Concertina Gait

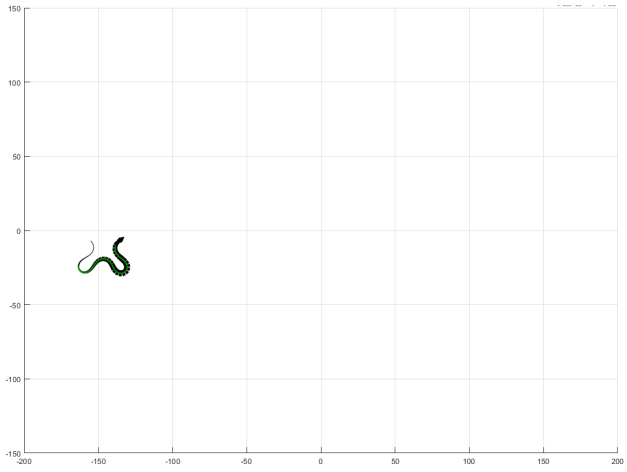


Figure 3-7: Snapshot of Lateral Undulation Gait

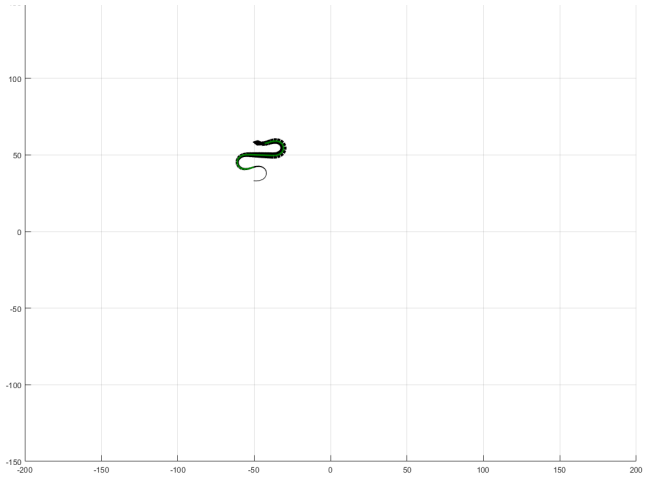


Figure 3-8: Snapshot of Side Wind Gait

All three of these gaits are uniquely mapped to non-speech audio sounds which allows the user to “hear” the gait if the visual animated display is not able to be seen. The mechanics of mapping the gait to the audio is explained below.

3.4 Snake Gait Audio Mapping

In Figure 3-9, starting at Line 127, sound generation is initialized by defining the sampling rate, period, and duration of the audio (6 seconds) to match the duration of the animation. Additionally, the C major scale is defined in terms of the frequency values for each note in the octave. The array “audioSignal” is initialized as a zero array but will later be populated with audio waveform values.

```
126 % Sound Generation
127 Fs = 44100; % Sample rate in Hz
128 T = 1/Fs; % Time step for audio
129 duration = 6; % Duration to match the snake animation
130 tAudio = 0:T:duration; % Time vector for audio
131
132 % Base notes for the major scale, starting from C
133 baseNotes = [261.63, 293.66, 329.63, 349.23, 392.00, 440.00, ...
134     493.88, 523.25];
135
136 % Initialize the audio signal array
137 audioSignal = zeros(1, length(tAudio));
138
```

Figure 3-9: Initialize Sound Generation [26]

Figure 3-10 shows the next piece of code which generates an audio signal based on the chosen gait pattern by processing the audio in time steps which is referred to as frames. Based on the gait pattern, the modulation amplitude and frequency are fed into the audioSignal function in Line 166 using a C major sixth chord to calculate the actual audio signal. The snake's gait is then updated in Lines 173 and 174 and the loop runs again until the tAudio length is reached.

```
% Loop through each frame in the audio
for frame = 1:length(tAudio)
  % Select the current base note - C (index 1)
  currentNoteIndex = 1;
  currentNote = baseNotes(currentNoteIndex);

  % Gait-specific modulations
  switch gaitPattern
    case 1 % Concertina
      % Calculate modulation factor based on active segments
      activeSegments = sum(abs(th) > 2.0); % Active segments thresholding
      % Simple amplitude modulation based on active segments
      modulationAmplitude = activeSegments / n;
      modulationFrequency = 0.5;
    case 2 % Lateral Undulation
      % Frequency modulation extent in Hz
      modulationFrequency = 1;
      % Keep constant amplitude
      modulationAmplitude = 1;
    case 3 % Side Winding
      % Tremolo effect based on time
      modulationAmplitude = abs(sin(2*pi*5*tAudio(frame)));
      modulationFrequency = 2;
  end

  % Calculate the actual audio signal for this frame with C major sixth chord
  % Add three notes of the C major sixth chord: C, E, and A
  audioSignal(frame) = modulationAmplitude * ( ...
      sin(2*pi*currentNote*tAudio(frame)*modulationFrequency) + ...  % C (root note)
      sin(2*pi*(currentNote + 4/12)*tAudio(frame)*modulationFrequency) + ... % E (major third)
      sin(2*pi*(currentNote + 9/12)*tAudio(frame)*modulationFrequency) ...  % A (major sixth)
  );

  % Update the gait for the next frame
  a = updateCounter(a, gaitPattern);
  th = updateGait(th, n, gaitPattern, a);
end
```

Figure 3-10: Calculate the Audio Signal Based on the Gait Pattern [26]

The next code section starting at Line 178, in Figure 3-11, processes the audio signal generated above and prepares it for playback and optionally saves it to a file. The audio file is normalized to a safe range for playback. The function “soundsc” plays the audio file through the computer’s sound system and the function “audiowrite” saves the audio to a .WAV file.

```
177  % Normalize the audio signal
178  audioSignal = audioSignal / max(abs(audioSignal));
179
180  % Play the generated sound
181  soundsc(audioSignal, Fs);
182
183  % Uncomment below to save the sound to a WAV file
184  audiowrite(fileNameSound, audioSignal, Fs);
```

Figure 3-11: Normalize, Play, and Save Sound [26]

The remaining parts of the code are labeled “helper functions” which are used throughout the code above. A brief description of each function is given below.

```
187  % Gait equation for each segment based on the selected pattern
188  function th = gait(i, gaitPattern, a)
189      th = 0;
190      switch gaitPattern
191          case 1 % Concertina
192              n = 40;
193              th = 8 * pi / n * sin(6 * pi / (n+1) * (i)) * exp(-0.01 * (i - (n+1)/2)^2);
194          case 2 % Lateral Undulation
195              n = 40;
196              phase = 0.07;
197              th = 5.5 * pi / n * sin(4 * pi / (n+1) * (i)) + phase;
198          case 3 % Side Winding
199              n = 30;
200              p = 4/3;
201              if (i < (n+1)/p)
202                  th = 4.5 * pi / n * sin(p * 2 * pi / (n+1) * (i));
203              else
204                  th = 0;
205              end
206      end
207  end
```

Figure 3-12: Gait Equation to Determine Angles for Each Segment [26]

The above function in Figure 3-12, line 188, called “gait”, uses an equation specific to the gait pattern to calculate the angle, *th*, for each segment of the snake robot to produce the associated movement during the animation. The gait equation [26] for the concertina movement is shown below:

$$th = \frac{8\pi}{n}\left(\sin\left(\frac{6\pi i}{n+1}\right)\right)e^{-0.01\left(i-\frac{n+1}{2}\right)^2} \quad (1)$$

This equation produces a stretch and compression type movement like the motion of an accordion [23], where $\frac{8\pi}{n}$ is the amplitude of the sinusoid, $\sin\left(\frac{6\pi i}{n+1}\right)$ produces the oscillations, and $e^{-0.01\left(i-\frac{n+1}{2}\right)^2}$ ensures that the segment's oscillation tapers off the farther the segment is from the head. Here, *n* is the number of segments and *th* is the angle of each segment with respect to the defined horizontal x-axis.

The gait equation [26] for the lateral undulation movement is shown below:

$$th = \frac{5.5\pi}{n}\left(\sin\left(\frac{4\pi i}{n+1}\right)\right) + phase \quad (2)$$

This equation produces smooth, lateral, damped sinusoidal waves that propel the snake forward, where $\frac{5.5\pi}{n}$ is the amplitude of the sinusoid, $\sin\left(\frac{4\pi i}{n+1}\right)$ produces the oscillations, and $phase$ produces a shift as the snake oscillates. Here, *n* is the number of segments and *th* is the angle of each segment with respect to the defined horizontal x-axis.

The gate equation [23] for the side-winding movement is shown below.

$$th = \frac{4.5\pi}{n}\left(\sin\left(\frac{2\pi pi}{n+1}\right)\right) \quad (3)$$

This equation produces back-and-forth sideways motion but only for a portion of the snake as can be seen by the if-else condition to determine whether to use the equation, where $\frac{4.5\pi}{n}$ is the amplitude of the sinusoid, $\sin\left(\frac{2\pi pi}{n+1}\right)$ produces the oscillations. In general, this function "gait" in

Figure 3-13 is modular, and the switch statement allows the gait pattern to change as the snake changes its motion.

```
208  % Update the counter based on the gait pattern
209  function a = updateCounter(a, gaitPattern)
210      switch gaitPattern
211          case 1
212              a = mod(a+1, 40);
213          case 2
214              a = mod(a+1, 40);
215          case 3
216              a = mod(a+1, 30);
217      end
218  end
```

Figure 3-13: Update the Counter for Smoothness [26]

The function in Figure 3-14 updates the counter "a" and is written to ensure that each gait pattern repeats periodically in a smooth fashion. The switch statement allows the counter to update according to the chosen gain pattern.

```
219  % Update the gait based on the selected pattern
220  function th = updateGait(th, n, gaitPattern, a)
221      th(1) = th(1) + th(2);
222      th(2:n-1) = th(3:n);
223      th(n) = gait(a, gaitPattern, a);
224  end
```

Figure 3-14: Update the Gait to Simulate Propagation [26]

The function in Figure 3-15 propagates each segment's motion to the next segment in order to simulate propagation and wave-like snake movement. Line 221 calculates the orientation angle of the head, th(1), and updates it based on the angle for the next segment allowing the head segment to move continuously with body segments.

```
225    % Generate the head shape based on the snake's head position
226    function [headX, headY] = headShape(headPos)
227        % Define the points for the head shape polygon
228         % Define the x-points of the head polygon
229        headX = [headPos(1) - 4*cos(headPos(3)),...
230                 headPos(1) - 4*cos(headPos(3)) + 0.2*2*sin(headPos(3)),...
231                 headPos(1) + 1*cos(headPos(3)) + 0.8*2*sin(headPos(3)),...
232                 headPos(1) + 4*cos(headPos(3)),...
233                 headPos(1) + 1*cos(headPos(3)) - 0.8*2*sin(headPos(3)),...
234                 headPos(1) - 4*cos(headPos(3)) - 0.2*2*sin(headPos(3)),...
235                 headPos(1) - 4*cos(headPos(3))];
236        % Define the y-points of the head polygon
237        headY = [headPos(2) - 4*sin(headPos(3)),...
238                 headPos(2) - 4*sin(headPos(3)) - 0.2*2*cos(headPos(3)),...
239                 headPos(2) + 1*sin(headPos(3)) - 0.8*2*cos(headPos(3)),...
240                 headPos(2) + 4*sin(headPos(3)),...
241                 headPos(2) + 1*sin(headPos(3)) + 0.8*2*cos(headPos(3)),...
242                 headPos(2) - 4*sin(headPos(3)) + 0.2*2*cos(headPos(3)),...
243                 headPos(2) - 4*sin(headPos(3))];
244    end
```

Figure 3-15: Snake Head Shape Polygon Calculation [26]

The function “headShape” in Figure 3-15, line 226, calculates the shape of the head as a polygon of seven connected x-y coordinates. The polygon is determined by calculating the x and y POSE values of each vertex of the polygon (headPos(1) and headPos(2)) and the overall orientation amount, in radians, (headPos(3) of the head. The function is dynamic so that it can be used as the snake’s POSE changes.

```
245    % Calculate the width of the snake's body segment
246    function w = width(i, l)
247        % Width function used for drawing each segment
248        n = 50;
249        a = 16 / l;
250        b = 0.003;
251        t = -1 * n * 2 / 3;
252        w = a * exp(-b * (i + t)^2);  % Tapered width equation
253    end
```

Figure 3-16: Calculate Segment Width [26]

The function “width” in Figure 3-16, line 246, calculates the width of each segment. The exponential function in Line 252 causes the segments to be tapered the farther they are from the head resulting in the smooth development of a tail which is a very tapered segment.

```
254  % Function to draw the snake's body by connecting the segments
255  function snake = drawSnakeBody(r, n, l)
256      % Code to draw each line segment of the snake's body
257       snake = zeros(n, 2);
258      for i = 1:n-1
259          segmentWidth = width(i, l); % Call the width function with the correct arguments
260          snake(i, 1) = line([r(1, i), r(1, i+1)], [r(2, i), r(2, i+1)], 'LineWidth', segmentWidth, 'color', '0,0,0,1.0');
261      end
262      for i = ceil(n/4):n
263          snake(i, 2) = line([r(1, i), r(1, i+1)], [r(2, i), r(2, i+1)], 'LineWidth', 2, 'color', '0,1,0,0.5');
264      end
265  end
```

Figure 3-17: Draw Snake Body [26]

The function in Figure 3-17 draws the body of the snake based on the matrix, r, which contains the information about each segment’s x and y position. The matrix “snake” is an n-segment by 2 matrix that contains the graphical data for each segment. The first column contains data for the main segments while the second column contains data for highlighted segments that enhance the visual display of the movement. The “width” helper function is called in Line 259 to calculate the taper of each segment the further it is from the head. The “line” function in Line 260 draws a line connecting two consecutive points such as r(1, i) and r(2, i).

3.5 Spectrogram Generation

To more easily understand and analyze the audio qualities of the generated sound, a spectrogram was created from the sound file. A spectrogram is a method that can visually represent sound as a function of time by displaying the frequencies that are present and their strengths over the course of a defined amount of time. Usually, these graphs present themselves as arrays of various colors where the different colors represent the intensity of the frequency and the concentration of the color represents the presence of a particular frequency.

The second MATLAB file that was used was DAFXsoundSweep.m which was also written by Dr. Alireza Mohammadi [26]. The purpose of this code is to seamlessly blend two of the gait patten audio files and generate a blended spectrogram. This spectrogram can then be compared with the two original audio file spectrograms to compare the frequency overlap and transitions within the individual files. The blended file can be used to analyze a sound produced by two overlapping gait patterns and determine which gait may have the more dominate presence if a snake robot's gait is a combination of two gait patterns. The next part of this Chapter will explain the major parts of this code. The complete code is listed in Appendix D.

```
25  FileNames = {'snakeConcertina.wav','snakeLatUndul.wav','snakeSideWind.wav'};
26  userChoice = [1, 3]; % The elements can be chosen from 1, 2, 3.
27
28  % Ask for input WAV filenames
29  firstFileName = FileNames{userChoice(1)};
30  secondFileName = FileNames{userChoice(2)};
31
32  % Read in the two sound files
33  [firstAudio, firstFs] = audioread(firstFileName);
34  [secondAudio, secondFs] = audioread(secondFileName);
35
36  % Check if both files have the same sampling frequency
37  if firstFs ~= secondFs
38      error('Sampling frequencies of the input files do not match.');
39  end
40
41  % Assuming both sound files have the same length
42  if length(firstAudio) ~= length(secondAudio)
43      error('Audio files do not have the same length.');
44  end
45
46  % Generate a sine sweep from 20 Hz to 20 kHz over the duration of the files
47  sweepDuration = length(firstAudio) / firstFs;
48  t = linspace(0, sweepDuration, length(firstAudio));
49  startFreq = 20;
50  endFreq = 20000;
51  sweepSignal = chirp(t, startFreq, sweepDuration, endFreq);
52
```

Figure 3-18: Importing Files and Performing Sine Sweep [26]

The first part of this code, as shown in Figure 3-18, imports two of the audio .WAV gait pattern files. The example shows in Line 26 that the Concertina and SideWind files were chosen.

The code then checks that both files were created with the same sampling frequency and have the same length. The “chirp” function in Line 51 generates a sine sweep from 20 Hz to 20 kHz for the duration of the files so that all frequencies are represented in the audio file which will make for a smooth transition when the two files are blended.

```
53  % Assuming sweepSignal is a column vector at this point
54  % Generate a smooth fade-in for the first audio file and a fade-out for the second
55  fade = 0.5 * (1 - cos(2 * pi * linspace(0, 1, length(sweepSignal))));
56
57  % Apply fading to the first audio signal and its inverse to the second
58  firstAudioFaded = firstAudio .* fade';
59  secondAudioFaded = secondAudio .* flipud(fade');
60
```

Figure 3-19: Fading First Audio File and Inverse Fading Second Audio File [26]

The next part of the code in Figure 3-19 fades and inverse fades the two files for smooth blending and seamless transition between the two audio files. A cosine function is used in Line 55 to fade the first audio file (snakeConcertain.wav) where the amplitude starts at 0 and gradually increases to a value of 1. The second audio file (snakeSideWind.wav) is inverse faded in Line 59 where the amplitude starts at 1 and gradually decreases to a value of 0. Each audio file is then multiplied by its fading process to create two new faded audio files.

```
61  % Blend the two faded audio clips
62  blendedAudio = firstAudioFaded + secondAudioFaded;
63
64  % Normalize the blended audio to prevent clipping
65  maxVal = max(abs(blendedAudio(:)));
66  if maxVal > 1
67      blendedAudio = blendedAudio / maxVal;
68  end
69
70  % Output filename
71  outputFileName = 'blendedSweep.wav';
72
73  % Write blended signal to new audio file
74  audiowrite(outputFileName, blendedAudio, firstFs);
75
76  disp(['Blended audio created and saved as ' outputFileName]);
77
78
```

Figure 3-20: Blending the Two Faded Audio Files [26]

After fading/inverse fading, the two audio files are blended (Line 62) and the blended file is normalized to prevent clipping (Line 67), as can be seen in Figure 3-20. The new file is then saved as “blendedSweep.wav”.

```
86  % Create the figure for displaying the simulation with specified axis limits
87  figure('Color', 'w'); % Set the background color to white during figure creation
88  set(gcf, 'WindowState', 'maximized'); % Maximize the figure window
89  hold on;
90  grid on;
91  axis equal;
92  % Plot the spectrogram for the first audio file
93  subplot(3, 1, 1);
94  spectrogram(firstAudio, hamming(1024, 'periodic'), 512, 1024, firstFs, 'yaxis');
95  title('Spectrogram of the first sound file');
96
97  % Plot the spectrogram for the second audio file
98  subplot(3, 1, 2);
99  spectrogram(secondAudio, hamming(1024, 'periodic'), 512, 1024, secondFs, 'yaxis');
100 title('Spectrogram of the second sound file');
101
102 % Plot the spectrogram for the blended sound
103 subplot(3, 1, 3);
104 spectrogram(blendedAudio, hamming(1024, 'periodic'), 512, 1024, firstFs, 'yaxis');
105 title('Spectrogram of the blended sound');
106
107 % Adjust the colormap to improve visibility
108 colormap('jet');
```

Figure 3-21: Plotting the Original Audio and Blended Audio Spectrograms [26]

The last part of the code, as shown in Figure 3-21, plots the spectrograms for each original audio (before fading) and the blended audio file to analyze the frequency spectrum for the duration of the audio. Before the results of the above code are discussed, the individual spectrograms of each audio file are generated and displayed below in Figures 3-22 – 3-24.

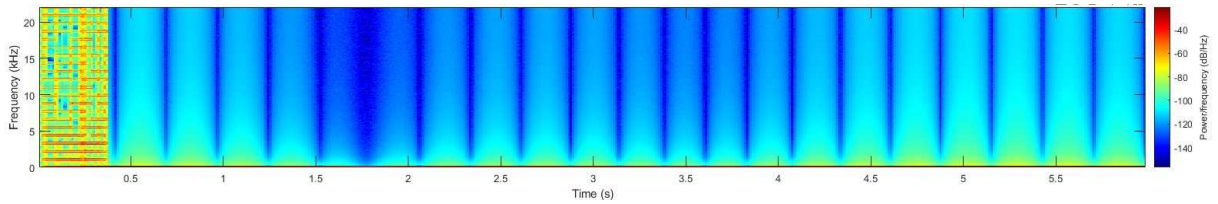


Figure 3-22: Spectrogram for snakeConcertina.wav

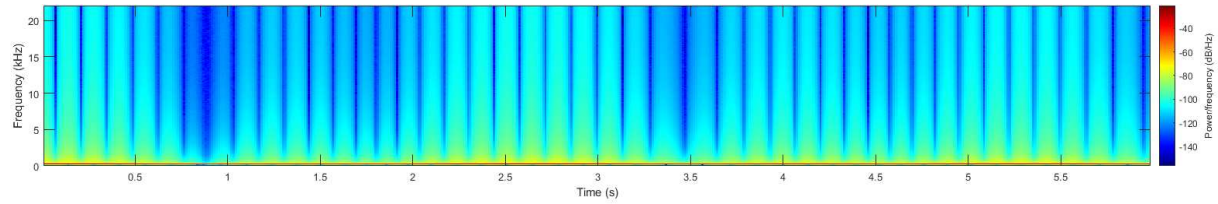


Figure 3-23: Spectrogram for snakeLateralUndulation.wav

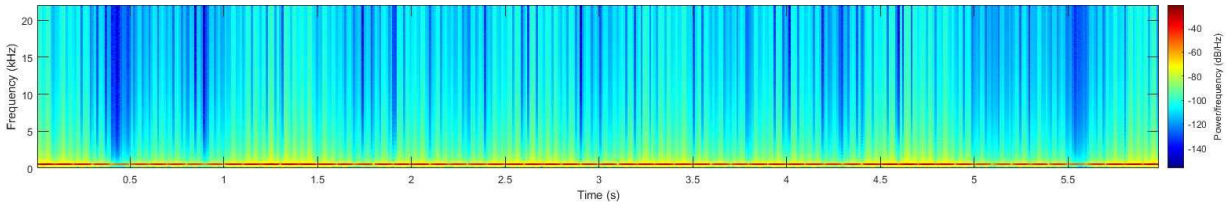


Figure 3-24: Spectrogram for snakeSideWind.wav

The spectrogram for snakeConcertina.wav in Figure 3-22 shows a distinct frequency pattern like accordion movement which is the expected pattern. The darker blue vertical lines represent a band of low energy while the lighter bands in between the darker lines represent areas of higher energy. The spectrogram for snakeLateralUndulation.way in Figure 3-23 shows pronounced oscillatory movement with a periodic shift from high to low energy. The spectrogram

for snakeSideWind.wav in Figure 3-24 shows less pronounced oscillatory movement and is overall lighter in color than the other two spectrograms indicating that this movement has more energy for the same time duration.

Once the last part of the code is executed in Figure 3-21 using the selected files, the blended audio spectrogram is generated and compared to the two original audio files as can be seen in Figure 3-25.

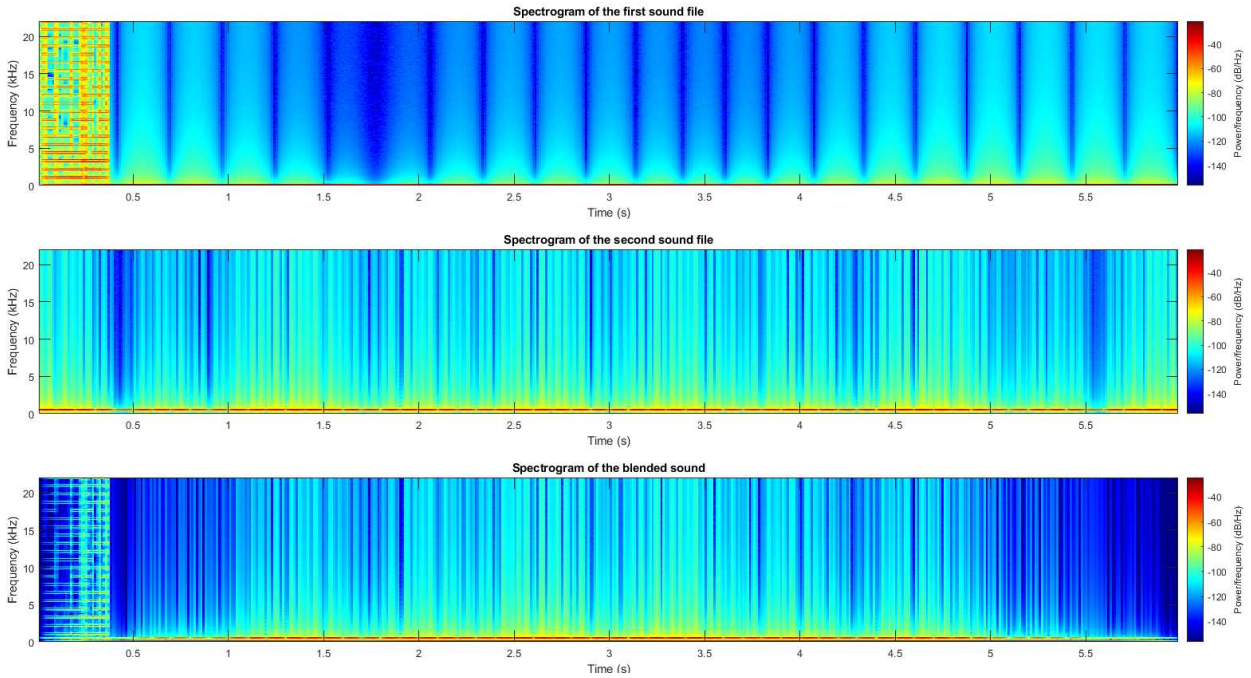


Figure 3-25: Spectrograms for Original Audio and Blended Audio Files

The last spectrogram is for the blended sound and is a representation of the combined Concertina and Side Wind movements. While both movements are represented, the frequency pattern from the Side Wind audio appears to be more represented than the Concertina audio however, there are overall more areas of lower energy for the duration of the spectrogram.

The two MATLAB files discussed in this chapter are used during the PBL activity. How the files are used during the activity will be explained in the next chapter. These files can be

extended to include additional snake movements buy adding more audio files that correspond to additional snake movements. Additional blending of audio files can be done to analyze the spectrograms for other combined movements.

Chapter 4: Project-Based Learning Activity Implementation

4.1 Motivation

The traditional high school science curriculum requires students to enroll in one or more of the three science disciplines, namely Physics, Chemistry, and Biology, to fulfill requirements for graduation. Beyond the traditional sciences, elective courses are usually offered that dive deeper into a science topic while keeping the level of technical understanding at a high school level. These courses are typically one semester (or trimester) in length and may use a textbook or a collection of reading and audio-visual materials accumulated by the teacher. The format of these courses can involve traditional note-taking, discussion-based meetings, experiments, and short or long-term projects. Students who enroll in these classes have an interest to further their knowledge in a certain type of science

Many high schools also offer Engineering classes either within the Science department or as a separate department. The topics of these classes are also designed to be at a high school level, using classical projects such as bridge building and robot building using popular microcontrollers such as Arduinos. Many of these topics are well-defined engineering challenges and are staples in a high school's engineering curriculum. Going past these types of engineering topics and exposing students to advanced college-level topics can be challenging for the teacher to have the time and access to materials to educate themselves such that they are able to successfully deliver information to their students. While there is an incredible amount of engineering research that would be very engaging for high school students to experience, it is challenging to make university-level research accessible to high school students, given that most research requires an advanced technical

understanding of concepts that rely on math and science skills that are not taught that the high school level. However, engaging high school students with university-level research at the appropriate depth, can potentially increase the level of STEM interest and desire to enroll in an Engineering curriculum in college by demonstrating to students the idea that advanced engineering topics are exciting fields of research that are accessible and possible pathways to a professional college degree. The method by which these advanced topics are delivered is important to convey to students the desired message that Engineering is an excellent college major and career choice. One very effective method that allows students to take an active role in learning and experiencing the material is through Project-Based Learning (PBL). The following sections in this chapter will explain the foundations of PBL and how it was used for the snake robot sonification project that was delivered in the Advanced Robotics Engineering class at Hopkins School in New Haven, Connecticut. For the remainder of this chapter, the school will be referred to as Hopkins.

4.2 Project-Based Learning at Hopkins School

The Fall Term class, called Robotics Engineering, at Hopkins is an example of a PBL class. In this class, teams of two to three students learn about the Engineering Design Process by identifying a problem in a community of their choice and designing a robot that can provide a solution to the problem. They are allowed the agency (voice and choice) to design, build, and test, a robot of their choice. Students are required to document their journey in an online Engineering Notebook.

The Advanced Robotics Engineering class at Hopkins is a fusion of both forms of PBL meaning that this class is a project-based class but with several medium-to-long-term projects that at times run parallel to one another. The topics of this class focus on artificial intelligence and

machine learning for robotics. One of the main projects is training a reinforcement learning (RL) model that can be used to teach a robotic dog how to walk. Seeing that the training and retraining can take several weeks to complete using the classroom computers, additional projects are introduced to the students that can be learned and worked on concurrently. The sonification of snake robot movement that is the subject of this dissertation is an example of a PBL activity that was introduced to the Advanced class while RL training was in progress. The entire PBL activity can be found in Appendix E.

4.3 Snake Robot Sonification Activity

The goal of this PBL activity was to introduce high school students to a college-level engineering research topic and assess their ability to understand the material and their level of engagement and interest. The PBL activity was titled "Exploring Robotic Snake Motion and Sonification with MATLAB" and was piloted for two class days of 55 minutes each. The class was comprised of 16 high school Juniors and Seniors and was delivered in May 2024 at Hopkins in the Advanced Robotics Engineering class. Lynn Connelly was the high school teacher present in person and Dr. Mohammadi of University of Michigan-Dearborn was present online using a Zoom connection. The objective of the activity was to introduce students to the subject of sonification and how it applies to the movement of a snake robot, which is a current research activity being investigated by Dr. Mohammadi in his Robotics Intelligence Lab. MATLAB scripts were used for the students to modify and investigate. For this pilot, students were not assigned any pre-reading or video watching before the activity began.

The students were first introduced to sonification through a class discussion and by watching short videos in class together. We talked about how we commonly interpret data using

visual methods such as graphs and tables. Students suggested that any sort of visual impairment would require an alternate method for data interpretation. Since high school math and science classes rely predominately on visual interpretation of data, thinking about a different type of method for understanding data was a thought-provoking exercise for the students. The class was not previously aware of sonification, nor its applications. Being able to "hear" the data was an engaging experience for the students.

The snake robot prototype was then shown, and its movement was demonstrated. We talked about how it was constructed and how the code was written. Further, students were asked questions about how this prototype resembles the movement of a real snake and what were some similarities and differences between features of the snake robot as compared to biological snakes. To highlight the thought process of an engineer, we discussed advantages and disadvantages of snake robots as compared to other types of robots with different types of mobility such as wheels and tread. To further students' understanding, we had an open discussion about the importance of these types of robots in search and rescue operations, industrial inspections, and medical procedures. Students showed great interest and curiosity in snake robots performing in these applications and were not previously aware of robotic snakes. We also discussed more novel applications such as underwater search and rescue and showed a short video on how a snake robot can move in water.

Once students understood the concept of sonification and the operation of snake robots, we talked about how these two ideas can be merged to engineer a solution for the applications that were suggested such as search and rescue. While these solutions are highly technical and advanced in their design and implementation, we stressed that all engineering solutions begin with a model, which is what the prototype presented to the students, was meant to achieve. As explained in Chapter 3, the snake robot motion data was previously videoed and processed using PASCO

Capstone software before the activity and the MATLAB scripts were generated before the class began. Students in this class were concurrently enrolled in or had previously taken a physics class, so they were aware of terms like center-of-mass (COM) and were familiar with using PASCO software for video analysis. Additionally, they were already using MATLAB as part of their Advanced Robotics class, so there was minimal learning curve when the MATLAB scripts were presented to them. Hopkins has a perpetual site license for the PASCO Capstone software and a secondary school license of MATLAB that includes fifty toolboxes with unlimited seat licenses and is renewed annually for $500. The software is available to the students download on their personal device and also to access as an online version on the MathWorks website.

Students were taken through the process of how the snake robot movement was captured using the Video Analysis feature of PASCO Capstone. They watched the video that was used for data capture, the data capture tracking picture, and the motion graphs that were created. It was decided that the sonification algorithm that was used to process that snake robot movement data, was too complex to present and teach during two class days; however, it was important for the students to understand how data can be interpreted using sound. As a result, two MATLAB scripts were created to provide a model for the sonification algorithm and to simulate the three different snake robot locomotion patterns, concertina, lateral undulation, and side winding that were presented in Chapter 3.

The PBL activity had four main parts – Familiarization with the Code, Simulation Observation, Code Modification, and Exploration of Sound Generation – which are summarized in Table 4-1 below, and results are explained in detail in the paragraphs that follow.

Table 4-1: Task and Description for PBL Snake Robot Activity

PBL Task	Description
Familiarization with Code	Study the MATLAB scripts and understand their structure and functionality. Identify which parts of the code are responsible for generating the three different snake gaits.
Simulation Observation	Run the MATLAB scripts for each of the three gait patterns and observe the resulting animations and sounds. Describe the distinct characteristics observed in both the movement patterns and the generated sounds for each gait.
Code Modification	Modify the line of code: activeSegments = sum(abs(th)>0.1) by changing the number 0.1 and setting it to different values and note how that changes the sonification of the robot's movement. Adjust the code so that the gait parameter can be asked from the user at the outset in an interactive manner.
Exploration of Sound Generation	Investigate the relationship between the snake's movement and the corresponding sounds generated in the script. Modify the sound generation code to change the modulationFrequency and the modulationAmplitude parameters to observe how that changes the sonification of the robot's movement.

Familiarization with Code – Since the students had been working with MATLAB for the past three months before the PBL activity started, they were comfortable scanning the code for understanding. The code was shared on a Google Drive to each student, who was able to open the MATLAB scripts on their personal device. The code was commented so that even the students who were not as proficient, could understand the code's functionality. Everyone was able to find the implementation of the three different snake gaits.

Simulation Observation – Each student was able to run both MATLAB scripts without issue. The students were allowed to independently run the code for each gait multiple times to familiarize themselves with each gait's functionality. Together the class then discussed the similarities and differences between each gait pattern and why the name of the gait suggested its

movement. Since no one in the class had previous experience with snakes, students were curious as to whether the three gaits were the only types of snake movement. We talked about how each gait represented core snake movement, but that actual movement is based on a combination of the three core gaits.

Code Modification – Students were able to modify the line of code that varied the number of active snake segments and ran the script to see the results. They were encouraged to work together in case some students needed peer assistance. Together the class discussed why the robot's sonification and movement were directly related to the number of segments. Since all students had Physics experience, the discussion was technical and involved kinematics and degrees of freedom. Students were able to correctly suggest that more active snake segments gave the robot more degrees of freedom which therefore translated into more curved snake movement. Students were not asked to modify the code for user interactive input since they were very interested in discussing snake gait patterns and how they can combine to make new patterns.

Exploration of Sound Generation – Students were asked to experiment with changing the modulation frequency and amplitude and observe the results by running the code. They developed an appreciation for the importance of sonification to interpret data. We had a good discussion about how snake robots can aid search and rescue operations especially when the search site is not feasible for a human to enter either because of space size or environmental conditions. When visibility is limited, students gained an appreciation for the importance of sonification to communicate snake robot movement which would translate into the type of environment it would be exploring since snakes take the shape of the path they are traversing.

4.4 Student Feedback

After the activity ended, students were asked to complete a feedback survey. This was used to evaluate the students' experience and to learn what should be improved or continued the next time this PBL activity is delivered. Instead of ranking questions using a numbering system, students were given questions that required a short answer response so that we could gain more detailed insight into their feedback. They were asked the following questions:

- Question1: What did you like about this activity?
- Question 2: What can be done to improve this activity?
- Question 3: What did you learn from these last two days?

For Question 1, students were excited to learn about sonification and how it can apply to robotics. They liked that the activity was interactive and that it was about a topic that was novel and not part of their prior learning. For Question 2, students were looking to do more investigation with MATLAB and suggested that giving students a chance to create their own snake algorithms would be an improvement to the activity. Since all of them knew how to use MATLAB, they were all ready and capable of creating their own code or make more in-depth modifications to the existing snake code. Next time this PBL activity is run, we will look into the possibility of extending the PBL duration time from 2 days to possibly 5 – 6 days and include more learning on how the sonification algorithm works from a mathematical process. If more time allowed, the activity could be extended even further and students could create their own snake robot and experience the entire process of data capture and processing through MATLAB. For Question 3, students met the goals for learning about sonification, snake robots, and how both are used together for various engineering applications. While no one commented on the use of MATLAB, it was a necessary delivery tool that allowed students to easily see and hear the different snake gait patterns.

Appendix F is the table of the entire survey results. The main issue with the activity was the timing which was in May 2024 during a week when Hopkins was administering AP exams. Some students were able to experience the full two days of PBL and others were only able to attend for one of the two days which also resulted in feedback responses from nine students instead of sixteen students. However, even with nine responses, the feedback provided helpful detail and reflection on how to modify the activity for the next year.

4.5 Reflections About PBL Activity and Additional Work

The format of the PBL activity was designed to follow the Gold Standard for PBL as outlined by the Buck Institute for Education, according to Section 4.3. The student feedback showed that this activity generated genuine interest in the fields of sonification and snake robots and introduced these topics, that were previously unknown, to the students and how engineering can use robotics in helpful ways for society.

After reflecting on the student feedback and how the two-day PBL activity was conducted, there are suggested improvements that can be incorporated into the activity next time it is delivered. It is possible that each improvement might extend the duration of the activity. Table 4-2 below, gives suggestions and recommended total duration time.

Table 4-2: Suggested Modifications and Additional Class Time Needed for PBL Activity

Modification	Additional Class Time Needed (Past 2 Days)	Additional Equipment Needed
Include non-snake example of sonification such as watching a short video of a Geiger counter.	None	None
Allow students to use MATLAB to combine the snake gait patterns instead of viewing them separately. In groups, students can then make a pattern of movements that another group has to listen to determine the correct pattern.	1 Day	None but some instruction maybe needed to help with MATLAB syntax.

Expose the mathematical model of the sonification algorithm used to create the gate patterns and associated sound and step through the code with examples.	1 Day	None
Create a mock search and rescue maze with the snake prototype and, in another room, have the students listen to the snake movement using the MATLAB code to map out the maze.	2 days	Materials to create a maze.
Use the snake robot prototype and allow the students to video the snake movement and collect data using PASCO Capstone.	3 days	Phone camera, tripod, PASCO Capstone software
Let students, in small groups, make their own snake robot from a kit that is created by the teacher.	5 Days using a pre-programmed Nano 10 Days using a Nano that the students program	Materials used in Chapter 3. Approx cost is $20 per robot.

This activity was given to a group of students that had math skills at the Pre-Calculus level and higher, already had experience with PASCO Capstone, had already taken a Physics class, and had been working with MATLAB for the past three months. They were well positioned to acquire the knowledge quickly over the course of the two-day period and based, on feedback, were eager to go even deeper into the material. Not all classes will have this same skillset so there are some considerations that should be made so that this PBL activity is accessible to all high school students no matter their skills. Some barriers that would need to be overcome when using the PBL activity in its current form are listed in Table 4-3 below.

Table 4-3: Problems and Solutions to Students Not Meeting the Expected PBL Skillset

Problem	Solution
Students do not have access to MATLAB	- The teacher uses a free trial copy and shows the code in the front of class as a group demo. - Students sign-up for their own free trial - The school purchases a secondary school license for $500 that can be renewed annually.
Students do not have their own device or their device is not fast enough to run MATLAB	- The activity can be conducted on a school computer lab that may have enough computers for each student or for a small group to share. - Students use laptops from a school computer cart. - Teachers show the code and demos in the front of the room using their device. - Students use the online version or phone version of MATLAB.
Students have not taken a Physics class	- The teacher should explain the minimum amount of physics needed in terms of objects that students can relate to.
Students have not taken a Pre-Calculus math class	- Teachers should not explain the sonification algorithm
Students do not have access to PASCO Capstone	- Use the PBL activity as it is. - If the teacher wants to have the students do their own Video Analysis, then students can sign up for a free 30-day Capstone trial.
Students are visually impaired	- The teacher can conduct the activity as it is and explore even more sonification using snake robots with the students by creating mock mazes.

The goal with this PBL activity is to engage students with Engineering as a possible career path and for the students to realize that Engineering is accessible and is a major in college that they can realistically pursue. Before the PBL activity is delivered to a class of students, it is recommended that the classroom teacher does the activity ahead of time and learn about each student's skillset that they will bring to the class.

Chapter 5: Conclusions and Future Work

5.1 Conclusions

Teaching university-level research in a high school classroom is possible using project-based learning. This concept which uses the "learning-by-doing" process engages students in an active classroom so that they experience the research by doing hands-on activities that demonstrate the main concepts with a method that is accessible and provides the motivation to further their learning. While these advanced math and science concepts are still present during a PBL activity, the way in which these concepts are taught are important for keeping students interested and invested in the project and realizing that a career such as Engineering is something that they can pursue. Instead of having students read lengthy university research papers, these advanced concepts are presented in a project-based learning environment that require the students to dynamically interact with the subject material in a manner that makes the concepts tangible and understandable while still maintaining the rigor of the research.

The PBL activity presented in this dissertation was focused on teaching the sonification of snake robot movement to the Advanced Robotics class at Hopkins School in New Haven, Connecticut. The duration of this pilot activity was two class days and students did not have previous knowledge of sonification and only had previous knowledge of non-snake robots. Overall, there was an overwhelming positive response to learning about these two concepts and how they can work together for various engineering applications. They were very receptive to learning advanced material and, if time allowed, would have preferred to dig deeper into the MATLAB code provided to them. High school students are incredibly curious individuals.

Exposing them to novel material beyond the typical high school curriculum has the potential to provide the motivation they need to enter college in a field that is exciting for them such as Engineering.

The PBL Activity in this dissertation was run for two class days of 55 minutes each as a small module in an advanced robotics class where the focus was not on building robots but instead on mathematically advanced robot concepts such as machine learning. There are many iterations that this activity can take to satisfy the teachers' goals and curricular requirements. If the focus of the class is building and coding a robot, then this PBL activity could be modified to span several weeks where students start from the beginning and build a custom snake robot, learn about embedded systems and code and test the snake robot motion. From there they can use similar methods as this dissertation and record video of the robot moving and create motion data that can be processed by the sonification algorithm. The class could then focus on sonification and learn how to create a search and rescue robot using sound to navigate a maze.

5.2 Future Work

5.2.1 Snake Robot Construction Improvements

The preparation that was needed to conduct this PBL activity was approximately one year. It was important to construct and code the snake robot so that it accurately reflected the motion of a biological snake. After several iterations both in physical construction and coding, a snake robot model was completed that could be used for video analysis and accurate data capture movement. The materials that were used to construct the snake robot were low cost but took time to assemble and perfect. Knowing this, it could be time consuming in a classroom setting for students to make the snake robot using the same materials in this dissertation. Instead, it would be interesting for

the teacher create computer-aided design (CAD) models of the snake head and linkages for easy, snap-together assembly. The CAD design could also incorporate the ability to modify the surface of the linkages and their interaction with the ground to model the kinetic friction force of snake scales. If knowing to CAD a model is one of the goals of the PBL, students could learn a CAD system such as Onshape and create simple three-dimensional snake linkages.

5.2.2 Embedded System Improvements

The use of the Arduino Nano and Nano Motor Carrier proved to be a convenient method to code simple snake movement and make changes as needed. This embedded system is a good entry level piece of hardware. If a machine learning feature was added to the PBL activity, it would be beneficial to switch to a Raspberry PI Pico W embedded system which has Wi-Fi capabilities and can communicate with a Raspberry PI that can house a machine learning model. The Pico W would also need a motor controller such as the L298N in order to control the current between the Pico and the motors.

5.2.3 Snake Robot Movement and Data Capture Improvements

Three different snake robot motions were created by turning the left and right motors on and off at different time intervals in the Nano code to experiment with achieving the best model for snake motion. While this was not explored with the students in detail, part of the PBL activity could be to give students the opportunity to research how and why biological snakes move and how to best mimic that in a robot and in code. While this would add on several days to the activity, it would increase the students' investment in understanding snake motion and how robots can be used to model biological animals.

5.2.4 MATLAB Script Improvements

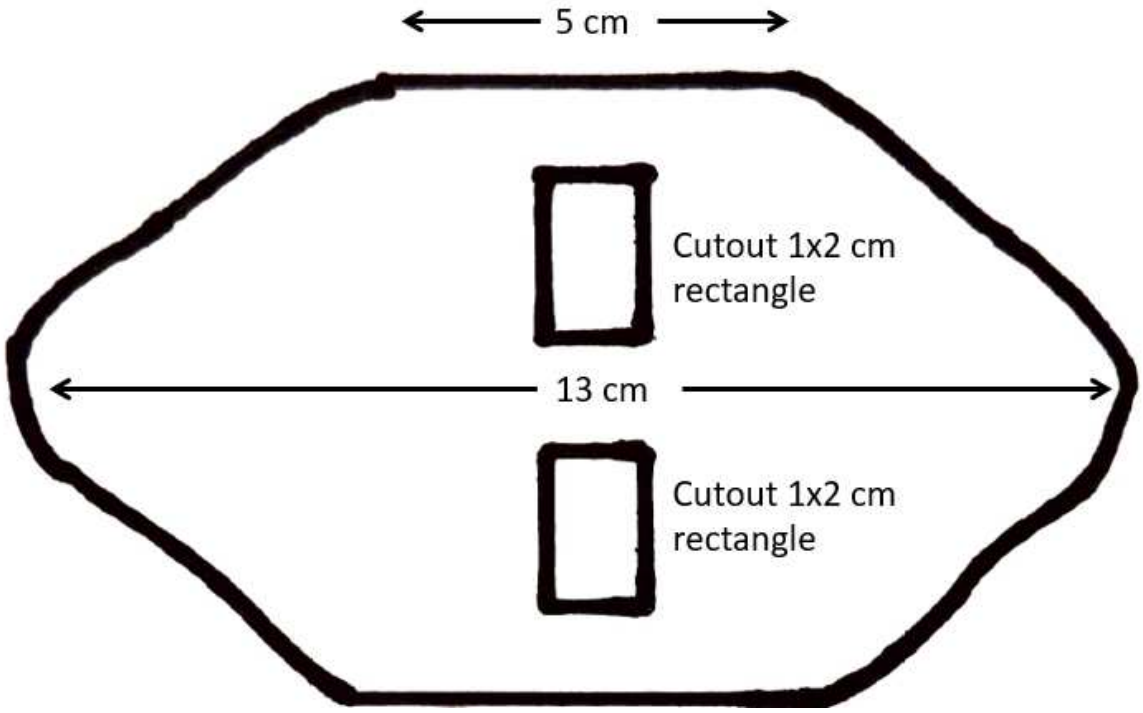
Using Dr. Mohammadi's sonification algorithm, three examples of snake movement were made and presented to the class. While the students did not learn the actual mathematical workings of the algorithm, they did learn how different snake movements can be represented as different sounds. They were essentially able to experience the mathematical model in action both visually and auditorily. Students were very intrigued at how math can be used to model movement and how sound can be used to interpret and understand data. There are further enhancements that can be made to this PBL activity based on the amount of class time that is available. Based on student feedback, the students were eager to learn more about the sonification algorithm functions. More time devoted to explaining the technical details is appropriate if the class can accommodate the additional PBL time.

5.2.5 Additional Advanced Topics

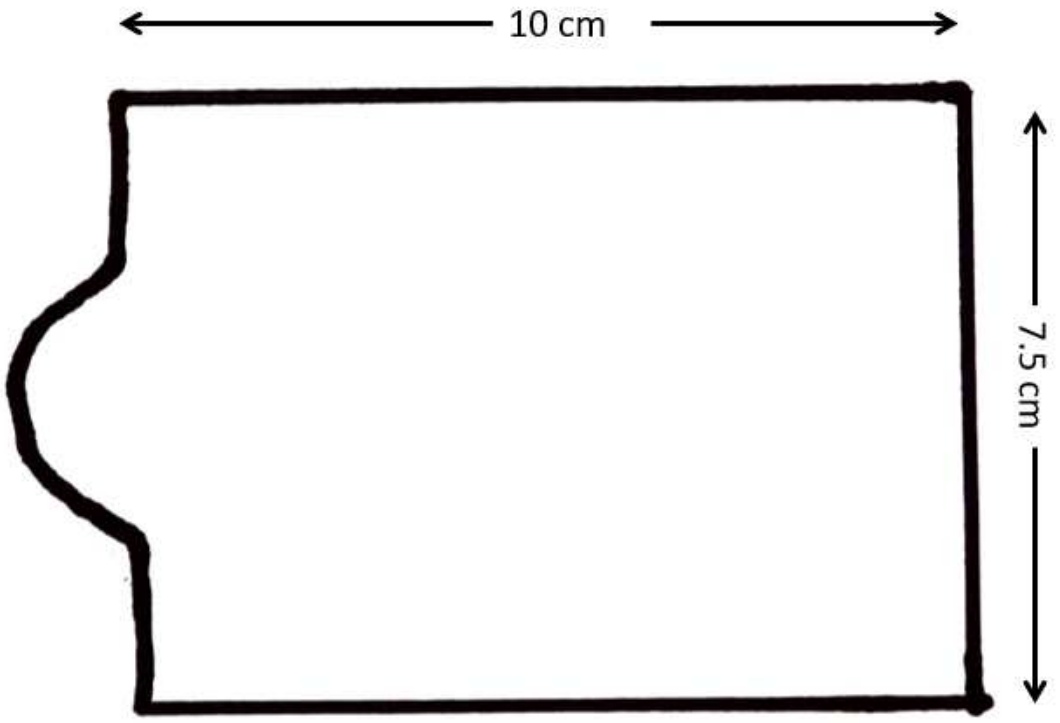
For a robotics class where the focus is not on building robots but instead, learning about more advanced topics, the motion of the snake could be trained using a reinforcement learning model (RL). Students could use MATLAB to first train a Simulink version of the snake that was imported from a CAD model. Once the simulation was running an acceptable snake movement, the trained model could be deployed to the actual snake robot and assessed for correct movement.

Appendices

Appendix A: Snake Robot Head and Linkage Patterns with Dimensions



Snake Robot Linkage Template (Actual Size)



Snake Robot Head Template (Actual Size)

Appendix B: Snake Robot Movement Code for Arduino Nano

This code was written using the Arduino IDE. It controls the snake robot movement. Three functions were written to experiment with achieving the correct snake motion.

```
//The one that was chosen can be found in the void loop() function
call.
//Author: Lynn Connelly Jan 2024

#include <ArduinoMotorCarrier.h>

//Variable to change the motor speed and direction
static int duty = 0;

void setup()
{
  //Serial port initialization
  Serial.begin(9600);
  //while (!Serial);

controller.begin(); //needed in order to start battery charging

  //Establishing the communication with the Motor Carrier
  if (controller.begin())
  {
    Serial.print("Motor Carrier connected, firmware version ");
    Serial.println(controller.getFWVersion());
  }
  else
  {
      Serial.println("Couldn't connect! Is the red LED blinking? You
      may need to update the firmware with FWUpdater sketch");
    while (1);
  }
  // Reboot the motor controller; brings every value back to default
```

```
  Serial.println("reboot");
  controller.reboot();
  delay(500);

  int dutyInit = 0; // at 50 it works as expected, at 60 shift sides
  and is too small duty to move, at 70 is very big duty.
  M1.setDuty(dutyInit);
  M2.setDuty(dutyInit);
  M3.setDuty(dutyInit);
  M4.setDuty(dutyInit);
  Serial.print("Duty init: ");
  Serial.println(dutyInit);
  delay(5000);  //Wait 5 secs until loop begins.
}

void loop() {

    snakeMove();

  //Keep active the communication between Nano & Motor Carrier
  //Ping the SAMD11
  controller.ping();
  //wait
  delay(50);
}

void snakeMove() {
  int motor_speed = 40;
  int motor_time = 500;
  int transition_amount = 15;

  for (int i = 1; i <= motor_speed; i++){
    M3.setDuty(-motor_speed);
    M4.setDuty(-motor_speed + i);
    delay(transition_amount);
  }

  //delay(motor_time);

  for (int i = 1; i <= motor_speed; i++){
```

```
    M3.setDuty(-motor_speed + i);
    M4.setDuty(-motor_speed);
    delay(transition_amount);
  }
}

void oldMove() {
    M1.setDuty(25);
    M2.setDuty(0);
    delay(350);
    M1.setDuty(0);
    M2.setDuty(-25);
    delay(350);
}

void simpleMove() {
  int motor_speed3 = 50;
  int motor_speed4 = 50;
  int motor_time3 = 1000;
  int motor_time4 = 1000;

  M3.setDuty(-motor_speed3); // turn one way
  delay(motor_time3);
  M3.setDuty(0);

  //delay(25);
  M3.setDuty(-motor_speed3); // drive straight
  M4.setDuty(-motor_speed4);
  delay(500);
  M3.setDuty(0);
  M4.setDuty(0);
  //delay(25);

  M4.setDuty(-motor_speed4);  // turn other way
  delay(motor_time4);
  M4.setDuty(0);

  //delay(25);
  M3.setDuty(-motor_speed3); // drive straight
  M4.setDuty(-motor_speed4);
```

```
  delay(500);
  M3.setDuty(0);
  M4.setDuty(0);
  //delay(25);
}
  //delay(motor_time);
```

Appendix C: MATLAB Code for Snake Robot Gait and Animation Generation

This MATLAB code, which is explained in detail in Chapter 3, generates three sample snake gaits and runs an animation for each gate.

```
%-------------------------------------------------------
% DAFXsnakeGaitGen2.m
% Snake Gait Generation (Animation and Sound)
% Author: Prof. Alireza Mohammadi
% Affiliation: Robotic Motion Intelligence Lab, Univ. Michigan-Dearborn
% First Version: February 26, 2024
% This Version:  May 8, 2024
%
% Description:
% This script generates animations for different snake robot locomotion
% patterns. It allows the user to select between concertina, lateral
% undulation, and side winding gaits, and saves the animation as a GIF
% file. The filename for the GIF is prompted from the user.
%
% Usage:
% Set the 'gaitPattern' variable at the top of the script to choose the
% desired gait: 1 for concertina, 2 for lateral undulation, and 3 for
% side winding. Run the script, and provide a filename when prompted to
% save the animation as a GIF.
%=======================================================


close all; clc; clear;
```

```
% Set the gait pattern: 1 for Concertina, 2 for Lateral Undulation,
% 3 for Side Winding
gaitPattern = 3; % <-- Set this to the desired pattern

% gaitPattern = ...
% input("Choose the snake robot gait pattern (1 for Concertina, 2
% for Lateral Undulation, 3 for Side Winding): \n");

% Simulation time step
delT = 0.02;
% Simulation final time
Tfinal = 1.5;

switch gaitPattern
   case 1
      filename = 'snakeConcertina.gif';
      fileNameSound = 'snakeConcertina.wav';
   case 2
      filename = 'snakeLatUndul.gif';
      fileNameSound = 'snakeLatUndul.wav';
   case 3
      filename = 'snakeSideWind.gif';
      fileNameSound = 'snakeSideWind.wav';
end
%
if isempty(filename)
   filename = 'snake_gait_animation.gif'; % Default filename if none is provided
end


% Initial position and configuration of the snake robot
```

```
x0 = -100; y0 = 0;
n = 45;    l = 2;

% Initialize the position vectors for the snake's segments
r = zeros(2, n+1);
r(:,1) = [x0; y0];

% Create the figure for displaying the simulation with specified axis limits
figure('Color', 'w'); % Set the background color to white during figure creation
set(gcf, 'WindowState', 'maximized'); % Maximize the figure window
hold on;
grid on;
axis equal;

% Use axis limits that will contain the snake for all gait patterns
% You may need to adjust these limits based on your specific gaits
axisLimits = [-200, 200, -150, 150];
axis(axisLimits);



% Initialize the segment angles array
th = zeros(1, n);
a = 30; % Counter used in the update step of the gait equations

% Initialization
for i=1:length(th)
   th(i) = gait(i, gaitPattern, a);
end

% Main simulation loop
```

```
for t=0:delT:Tfinal
    % Update coordinates
    for i=2:length(r)
        temp = [0;0];
        for j=1:i-1
            temp = temp + l*[cos(sum(th(1:j)));sin(sum(th(1:j)))];
        end
        r(:,i) = r(:,1) + temp;
    end

    % Compute the head's position and orientation
    headPos = [r(:,n+1); sum(th)];
    [headX, headY] = headShape(headPos);

    % Draw the snake's body
    snake = drawSnakeBody(r, n, l);

    % Draw the head
    head = fill(headX, headY, [0, 0, 0]);

   % Capture the frame
    drawnow; % Ensure the figure updates have been processed
    frame = getframe(gcf);
    im = frame2im(frame);
    [imind, cm] = rgb2ind(im, 256);

    % Write to the GIF file
    if t == 0
        imwrite(imind, cm, filename, 'gif', 'Loopcount', inf, 'DelayTime', delT);
    else
        imwrite(imind, cm, filename, 'gif', 'WriteMode', 'append', 'DelayTime', delT);
```

```
    end

    % Remove the snake for the next frame
    delete(snake);
    delete(head);

    % Update positions for movement
    r(:,1:n) = r(:,2:n+1);

    % Update the gait
    a = updateCounter(a, gaitPattern);
    th = updateGait(th, n, gaitPattern, a);
end

% Sound Generation
Fs = 44100; % Sample rate in Hz
T = 1/Fs; % Time step for audio
duration = 6; % Duration to match the snake animation
tAudio = 0:T:duration; % Time vector for audio

% Base notes for the major scale, starting from C
baseNotes = [261.63, 293.66, 329.63, 349.23, 392.00, 440.00, ...
    493.88, 523.25];

% Initialize the audio signal array
audioSignal = zeros(1, length(tAudio));

% Loop through each frame in the audio
for frame = 1:length(tAudio)
  % Select the current base note - C (index 1)
  currentNoteIndex = 1;
```

```
currentNote = baseNotes(currentNoteIndex);

% Gait-specific modulations
switch gaitPattern
  case 1 % Concertina
    % Calculate modulation factor based on active segments
    activeSegments = sum(abs(th) > 2.0); % Active segments thresholding
    % Simple amplitude modulation based on active segments
    modulationAmplitude = activeSegments / n;
    modulationFrequency = 0.5;
  case 2 % Lateral Undulation
    % Frequency modulation extent in Hz
    modulationFrequency = 1;
    % Keep constant amplitude
    modulationAmplitude = 1;
  case 3 % Side Winding
    % Tremolo effect based on time
    modulationAmplitude = abs(sin(2*pi*5*tAudio(frame)));
    modulationFrequency = 2;
end

% Calculate the actual audio signal for this frame with C major sixth chord
% Add three notes of the C major sixth chord: C, E, and A
audioSignal(frame) = modulationAmplitude * ( ...
    sin(2*pi*currentNote*tAudio(frame)*modulationFrequency) + ...  % C (root note)
    sin(2*pi*(currentNote + 4/12)*tAudio(frame)*modulationFrequency) + ... % E (major
third)
    sin(2*pi*(currentNote + 9/12)*tAudio(frame)*modulationFrequency) ...  % A (major
sixth)
);
```

```
  % Update the gait for the next frame
  a = updateCounter(a, gaitPattern);
  th = updateGait(th, n, gaitPattern, a);
end

% Normalize the audio signal
audioSignal = audioSignal / max(abs(audioSignal));

% Play the generated sound
soundsc(audioSignal, Fs);

% Uncomment below to save the sound to a WAV file
audiowrite(fileNameSound, audioSignal, Fs);




%================================================
% Helper Functions:

% Gait equation for each segment based on the selected pattern
function th = gait(i, gaitPattern, a)
    th = 0;
    switch gaitPattern
        case 1 % Concertina
            n = 40;
            th = 8 * pi / n * sin(6 * pi / (n+1) * (i)) * exp(-0.01 * (i - (n+1)/2)^2);
        case 2 % Lateral Undulation
            n = 40;
            phase = 0.07;
            th = 5.5 * pi / n * sin(4 * pi / (n+1) * (i)) + phase;
        case 3 % Side Winding
```

```
            n = 30;
            p = 4/3;
            if (i < (n+1)/p)
                th = 4.5 * pi / n * sin(p * 2 * pi / (n+1) * (i));
            else
                th = 0;
            end
        end
    end

    % Update the counter based on the gait pattern
    function a = updateCounter(a, gaitPattern)
        switch gaitPattern
            case 1
                a = mod(a+1, 40);
            case 2
                a = mod(a+1, 40);
            case 3
                a = mod(a+1, 30);
        end
    end

    % Update the gait based on the selected pattern
    function th = updateGait(th, n, gaitPattern, a)
        th(1) = th(1) + th(2);
        th(2:n-1) = th(3:n);
        th(n) = gait(a, gaitPattern, a);
    end

    % Generate the head shape based on the snake's head position
    function [headX, headY] = headShape(headPos)
```

```
    % Define the points for the head shape polygon
     % Define the x-points of the head polygon
    headX = [headPos(1) - 4*cos(headPos(3)),...
            headPos(1) - 4*cos(headPos(3)) + 0.2*2*sin(headPos(3)),...
            headPos(1) + 1*cos(headPos(3)) + 0.8*2*sin(headPos(3)),...
            headPos(1) + 4*cos(headPos(3)),...
            headPos(1) + 1*cos(headPos(3)) - 0.8*2*sin(headPos(3)),...
            headPos(1) - 4*cos(headPos(3)) - 0.2*2*sin(headPos(3)),...
            headPos(1) - 4*cos(headPos(3))];
    % Define the y-points of the head polygon
    headY = [headPos(2) - 4*sin(headPos(3)),...
            headPos(2) - 4*sin(headPos(3)) - 0.2*2*cos(headPos(3)),...
            headPos(2) + 1*sin(headPos(3)) - 0.8*2*cos(headPos(3)),...
            headPos(2) + 4*sin(headPos(3)),...
            headPos(2) + 1*sin(headPos(3)) + 0.8*2*cos(headPos(3)),...
            headPos(2) - 4*sin(headPos(3)) + 0.2*2*cos(headPos(3)),...
            headPos(2) - 4*sin(headPos(3))];
end

% Calculate the width of the snake's body segment
function w = width(i, l)
    % Width function used for drawing each segment
    n = 50;
    a = 16 / l;
    b = 0.003;
    t = -1 * n * 2 / 3;
    w = a * exp(-b * (i + t)^2);  % Tapered width equation
end

% Function to draw the snake's body by connecting the segments
function snake = drawSnakeBody(r, n, l)
```

```
    % Code to draw each line segment of the snake's body
     snake = zeros(n, 2);
    for i = 1:n-1
        segmentWidth = width(i, l); % Call the width function with the correct arguments
        snake(i, 1) = line([r(1, i), r(1, i+1)], [r(2, i), r(2, i+1)], 'LineWidth', segmentWidth,
'color', '0,0,0,1.0');
    end
    for i = ceil(n/4):n
        snake(i, 2) = line([r(1, i), r(1, i+1)], [r(2, i), r(2, i+1)], 'LineWidth', 2, 'color',
'0,1,0,0.5');
    end
end
```

Appendix D: MATLAB Code for Blending Sound Files

This MATLAB code, which is explained in detail in Chapter 3, blends the sounds files and generates a spectrogram.

```
%=========================================================
% DAFXsoundSweep.m
% Sound File Frequency Content Analysis
% Sine Sweep Audio Mixing Script
% Author: Prof. Alireza Mohammadi
% Affiliation: Robotic Motion Intelligence Lab, University of Michigan-Dearborn
% First Version: February 26, 2024
% This Version:  February 26, 2024
%
% Description:
% This script analyzes the frequency content of two sound files and a blended
% sound file generated from them. It computes the Short-Time Fourier Transform
% (STFT) for each sound file and displays their spectrograms for comparison.
%
% The script reads WAV files whose names are provided by the user and outputs
% visualizations of their frequency content over time. It assumes that sound
% files are monophonic and have the same sampling frequency and duration.
%
% Usage:
% Run the script in MATLAB and follow the prompts to input the filenames of
% the sound files to be analyzed. Ensure that the MATLAB workspace is clean
% prior to running the script to avoid any conflicts with existing variables.
%=========================================================
```

```
FileNames = {'snakeConcertina.wav','snakeLatUndul.wav','snakeSideWind.wav'};
userChoice = [1, 3]; % The elements can be chosen from 1, 2, 3.

% Ask for input WAV filenames
firstFileName = FileNames{userChoice(1)};
secondFileName = FileNames{userChoice(2)};

% Read in the two sound files
[firstAudio, firstFs] = audioread(firstFileName);
[secondAudio, secondFs] = audioread(secondFileName);

% Check if both files have the same sampling frequency
if firstFs ~= secondFs
   error('Sampling frequencies of the input files do not match.');
end

% Assuming both sound files have the same length
if length(firstAudio) ~= length(secondAudio)
   error('Audio files do not have the same length.');
end

% Generate a sine sweep from 20 Hz to 20 kHz over the duration of the files
sweepDuration = length(firstAudio) / firstFs;
t = linspace(0, sweepDuration, length(firstAudio));
startFreq = 20;
endFreq = 20000;
sweepSignal = chirp(t, startFreq, sweepDuration, endFreq);

% Assuming sweepSignal is a column vector at this point
% Generate a smooth fade-in for the first audio file and a fade-out for the second
fade = 0.5 * (1 - cos(2 * pi * linspace(0, 1, length(sweepSignal))));
```

```
% Apply fading to the first audio signal and its inverse to the second
firstAudioFaded = firstAudio .* fade';
secondAudioFaded = secondAudio .* flipud(fade');

% Blend the two faded audio clips
blendedAudio = firstAudioFaded + secondAudioFaded;

% Normalize the blended audio to prevent clipping
maxVal = max(abs(blendedAudio(:)));
if maxVal > 1
   blendedAudio = blendedAudio / maxVal;
end

% Output filename
outputFileName = 'blendedSweep.wav';

% Write blended signal to new audio file
audiowrite(outputFileName, blendedAudio, firstFs);

disp(['Blended audio created and saved as ' outputFileName]);



% Compute the STFT for the first audio file
% [s1, f1, t1] = stft(firstAudio, firstFs, 'Window', hamming(1024, 'periodic'), 'OverlapLength', 512, 'FFTLength', 1024);
% % Compute the STFT for the second audio file
% [s2, f2, t2] = stft(secondAudio, secondFs, 'Window', hamming(1024, 'periodic'), 'OverlapLength', 512, 'FFTLength', 1024);
% % Compute the STFT for the blended audio
```

```
% [s3, f3, t3] = stft(blendedAudio, firstFs, 'Window', hamming(1024, 'periodic'), 'OverlapLength', 512, 'FFTLength', 1024);

% Create the figure for displaying the simulation with specified axis limits
figure('Color', 'w'); % Set the background color to white during figure creation
set(gcf, 'WindowState', 'maximized'); % Maximize the figure window
hold on;
grid on;
axis equal;
% Plot the spectrogram for the first audio file
subplot(3, 1, 1);
spectrogram(firstAudio, hamming(1024, 'periodic'), 512, 1024, firstFs, 'yaxis');
title('Spectrogram of the first sound file');

% Plot the spectrogram for the second audio file
subplot(3, 1, 2);
spectrogram(secondAudio, hamming(1024, 'periodic'), 512, 1024, secondFs, 'yaxis');
title('Spectrogram of the second sound file');

% Plot the spectrogram for the blended sound
subplot(3, 1, 3);
spectrogram(blendedAudio, hamming(1024, 'periodic'), 512, 1024, firstFs, 'yaxis');
title('Spectrogram of the blended sound');

% Adjust the colormap to improve visibility
colormap('jet');
```

Appendix E: In-Class PBL Activity Handout

Project: Exploring Robotic Snake Motion and Sonification with MATLAB

Instructors: Ms. Lynn Connelly (Hopkins School) &

Dr. Alireza Mohammadi (The University of Michigan-Dearborn)

Objective: Understand the principles of snake robot locomotion and the concept of motion sonification by modifying and extending a given MATLAB script.

Background: Snake robots are robotic devices that mimic the movement patterns of biological snakes. These robots can navigate through challenging terrain and confined spaces, making them useful for various applications, including search and rescue missions, industrial inspections, and medical procedures. A sample snake robot has been constructed by your teacher. You will be working with a MATLAB simulator that will sonify the motion pattern of snake robots. In this project, you will first see a simple snake robot in action!

Part 1) A Simple Snake Robot Prototype:

Ms. Connelly has created a prototype snake robot! Using the data and videos below, carefully observe the robot's movements and answer the questions that follow.

- **Video 1: Snake Robot Motion and Motion Data Plots** (The video "snakebot.wav" and the gif files "time_profiles_animation.gif" as well as "trajectory_animation.gif" will be provided to you by Ms. Connelly)
- **Video 2: Biological Snake Motion** (https://www.youtube.com/watch?v=5CchyctRFrQ)

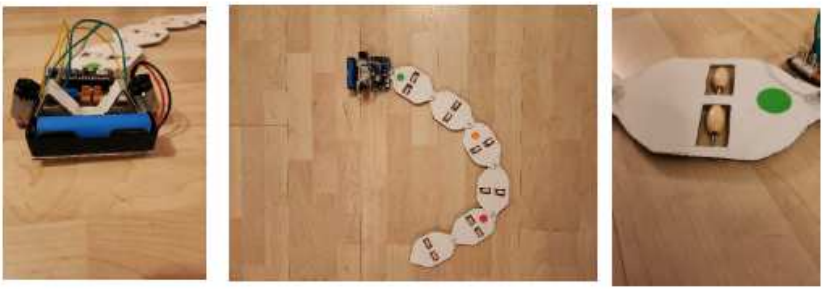


FIGURE 1 THE SNAKE ROBOT PROTOTYPE CONSTRUCTED BY MS. CONNELLY

Questions:

1. **Summarizing Robot Movement:** Describe the motion of Ms. Connelly's snake robot. How does it propel itself? In what ways does it resemble the movement of a real snake?

2. **Comparing Movement:** Identify the key similarities and differences in the way the snake robot and the biological snake move. (Hint: Snakeskin versus the beads, snake muscles versus the two motors)
3. **Advantages and Challenges:** What might be some advantages of a snake robot's motion compared to a robot with wheels or legs? Can you think of any challenges engineers might face when designing a snake robot?

Part 2) A MATLAB Snake Motion Simulator and Sonification of Its Motion Patterns:

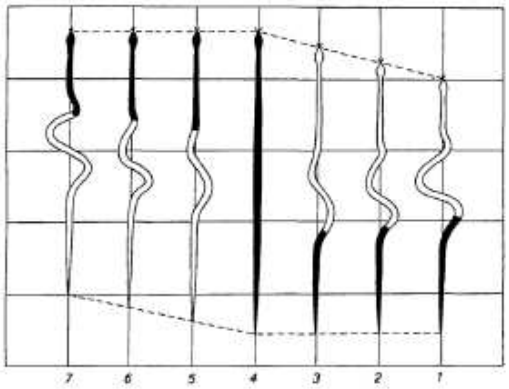


FIGURE 2 A SAMPLE SNAKE LOCOMOTION PATTERN (CONCERTINA)

Motion sonification refers to the process of translating the movement of a system into sound (check Dr. Mohammadi's Dec. 2023 slides). In this part of the project, you will work on a script that simulates different snake robot locomotion patterns and creates corresponding sound patterns. The MATLAB M-files will be shared with you by Ms. Connelly.

Tasks/Questions:

1. **Familiarization with the Code:**
   - Study the provided MATLAB script and understand its structure and main components.
   - Identify the segments of code responsible for generating the three different snake gaits (concertina, lateral undulation, and side winding).
2. **Simulation Observation:**
   - Run the MATLAB script for each of the three gait patterns and observe the resulting animations and sounds.
   - Describe the distinct characteristics you observe in both the movement patterns and the generated sounds for each gait.
3. **Code Modification:**

- Modify the line of code activeSegments = sum(abs(th) > 0.1) by changing the number 0.1 and setting it to different values and note how that changes the sonification of the robot's movement.
- Adjust the code so that the gait parameter can be asked from the user at the outset in an interactive manner.

4. **Exploration of Sound Generation:**
   - Investigate the relationship between the snake's movement and the corresponding sounds generated in the script.
   - Modify the sound generation code to change the modulationFrequency and modulationAmplitude parameters and observe how that changes the sonification of the robot's movement.
5. **Report:**
   - Write a report detailing your observations, modifications, and the implementation of the new gait and its sonification. Include screenshots of the animations and an explanation of the sonification choices you made.
   - Discuss potential real-world applications of snake robots and the value of sonifying their movements.

Acknowledgements: This work is supported by the National Science Foundation (NSF) through the award number CMMI-2153744.

References

Liljebäck, P., Pettersen, K.Y., Stavdahl, Ø. and Gravdahl, J.T., 2013. Lateral undulation of snake robots: A simplified model and fundamental properties. *Robotica*, *31*(7), pp.1005-1036.

Kacem, A., Zbiss, K., Watta, P. and Mohammadi, A., 2024. Wave space sonification of the folding pathways of protein molecules modeled as hyper-redundant robotic mechanisms. *Multimedia Tools and Applications*, *83*(2), pp.4929-4949.

Zahray, L., Savery, R., Syrkett, L. and Weinberg, G., 2020, August. Robot gesture sonification to enhance awareness of robot status and enjoyment of interaction. In 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN) (pp. 978-985). IEEE.

Appendix F: Student Feeback Survey Results

Timestamp	What did you like about this activity?	What can be done to improve this activity?	What did you learn from these two days?
5/16/2024 9:26:29	learning about sonification	I don't know, I don't know what we did the second day	
5/16/2024 9:27:03	I loved the interactive element as well as learning the applications of sonification.	Maybe more MATLAB stuff?	I learned the different gaits of snakes, how they can be represented with sound, and the applications of both snake robots and sonification.
5/16/2024 9:27:04	It was interesting to see what is being done in the current sonification field.	Scheduling during a non AP week. Otherwise it was awesome.	The use cases and benefits of sonification.
5/16/2024 9:28:22	The content matter was interesting and bringing in another professor was a good experience.	Not doing it during ap week	Data = audio audio can be changed. I also missed the first day due to an ap exam and was a bit confused on the next day.
5/16/2024 9:28:58	I liked how we worked with a university professor which I think is a unique experience for high school students	I think doing a little bit more in depth learning and experimentation would be better	I learned the applications of snake robots and using sonification to track their movement
5/16/2024 9:31:21	I thought the topic was really interesting! I never really considered how important snake-like robots were, and I also thought it was interesting how sound can be a better device for data than visual sources.	I think a little more participation with Matlab would've been cool. Maybe if there was some more time, students could make their own snake algorithms.	- the different types of snake movement - sonification - applications for snake robots
5/16/2024 9:32:09	I liked the aspect of learning a new topic and being able to talk to a professor in the field of study.	Don't do it during AP week.	At first I thought sonification was pointless but now I realized that it is actually pretty cool to be able to recognize a pattern from hearing instead of sight.
5/16/2024 9:34:25	I liked how the activity was interactive and he was really good at answering our questions and helping us learn.	Maybe make it at a different time when seniors are more locked in—earlier in the semester?	I learned about snake robots and sonification. I thought it was really interesting learning about how snakes move and how their bodies interact differently with friction.
5/16/2024 9:34:36	I enjoyed how the activity explained both the mathematical models for snake movement and also learning about how this movement is replicated in practice. I also enjoyed the interactive discussion.	I was only here for the first day, but I enjoyed all of it! Maybe not doing it during AP week would be a slight improvement...	I learned about how snakes move! I had never stopped to consider how the slither actually works, but learning about how snakes lift their bodies to redistribute their weight and how similar this is to human movement was very interesting.

References

[1] Tight, M. (2020). Twenty-first century skills: meaning, usage and value. European Journal of Higher Education, 11(2), 160–174. https://doi.org/10.1080/21568235.2020.1835517

[2] Herianto, Ikhsan, J., & Purwastuti, L. A. (2024). Developing student 21st-century skills through STEM engineering design learning cycle (STEM-EDELCY) model. The Journal of Educational Research, 117(3), 137–150. https://doi.org/10.1080/00220671.2024.2346659

[3] Wagner, Tony. The Global Achievement Gap : Why Even Our Best Schools Don't Teach the New Survival Skills Our Children Need and What We Can Do About It, Basic Books, (2014). ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/umichigan/detail.action?docID=1486487.

[4] Bartneck, C., Belpaeme, T., Eyssel, F., Kanda, T., Keijsers, M., & Sabanovic, S. (2024). Human-Robot Interaction – An Introduction. (2nd edition) Cambridge: Cambridge University Press.

[5] Luo, Y., Liu, J., Gao, Y., Lu, Z. (2014). Smartphone-Controlled Robot Snake for Urban Search and Rescue. In: Zhang, X., Liu, H., Chen, Z., Wang, N. (eds) Intelligent Robotics and Applications. ICIRA 2014. Lecture Notes in Computer Science(), vol 8917. Springer, Cham. https://doi.org/10.1007/978-3-319-13966-1_35

[6] Pettersen, Kristin Y., Snake robots, Annual Reviews in Control, Volume 44, 2017, Pages 19-44, ISSN 1367-5788, https://doi.org/10.1016/j.arcontrol.2017.09.006.

[7] Zahray, L., Savery, R., Syrkett, L.,. and Weinberg, G.,"Robot Gesture Sonification to Enhance Awareness of Robot Status and Enjoyment of Interaction," 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Naples, Italy, 2020, pp. 978-985, doi: 10.1109/RO-MAN47096.2020.9223452.

[8] Hermann, T., Hunt, A., Neuhoff, J. G., The Sonification Handbook. Berlin: Logos Verlag, 2011.

[9] Kacem, A, Zbiss, K, & Mohammadi, A. "Transforming Motion Into Sound: A Novel Sonification Approach for Teams of Mobile Robots." Proceedings of the 2024 International Symposium on Flexible Automation. 2024 International Symposium on Flexible Automation. Seattle, Washington, USA. July 21–24, 2024. V001T07A005. ASME. https://doi.org/10.1115/ISFA2024-140358

[10] Bender, William N.. Project-Based Learning : Differentiating Instruction for the 21st Century, Corwin Press, 2012. ProQuest Ebook Central

[11] Dewey, John, and Patricia H. Hinchey. Moral Principles in Education and My Pedagogic Creed by John Dewey : With a Critical Introduction by Patricia H. Hinchey. Myers Education Press, 2019. EBSCOhost

[12] Zadok, Y. and Voloch, N. (2018) 'Applying PBL to teaching robotics', Int. J. Innovation and Learning, Vol. 24, No. 2, pp.138–151.

[13] The Buck Institute for Education, 2025. [Online]. Available: https://www.pblworks.org/. [Accessed: Mar 14, 2025].

[14] Catlin, D. (2019). Beyond Coding: Back to the Future with Education Robots. In: Daniela, L. (eds) Smart Learning with Educational Robotics. Springer, Cham. https://doi.org/10.1007/978-3-030-19913-5_1

[15] Mohammadi, Alireza, Heilman, Destin, "Protein Molecules as Robotic Mechanisms: An Interdisciplinary Project-Based Learning Experience at the Intersection of Biochemistry and Robotics", ASEE Annual Conference and Exposition, Conference Proceedings, 06/2023

[16] Brainergiser, *How to Make a Snake Robot at Home*, 2025. [Online]. Available: https://www.youtube.com/watch?v=20GU1Dcpep4&t=221s. [Accessed: Mar 14, 2025].

[17] Nano 33 IoT, 2025. [Online]. Available: https://docs.arduino.cc/hardware/nano-33-iot/. [Accessed: Mar 14, 2025].

[18] Nano Motor Carrier, 2025. [Online]. Available: https://docs.arduino.cc/hardware/nano-motor-carrier/. [Accessed: Mar 14, 2025].

[19] Arduino MKR Motor Carrier, 2025. [Online]. Available: https://www.electronics-lab.com/arduino-mkr-motor-carrier/. [Accessed: Mar 14, 2025].

[20] Connelly, Lynn, Video of snake robot movement with Motor 1, https://drive.google.com/file/d/1HXKgq_5pfx1fJCc4rDTqot8xkr15SV7m/view?usp=sharing.

[21] Connelly, Lynn, Video of snake robot movement with Motor 2, https://drive.google.com/file/d/15pRwknrEm_oxpNHxZMP-94bXewC5RJsz/view?usp=sharing.

[22] PASCO, *Capstone*, [Online download], Roseville, CA, 2025, https://www.pasco.com/downloads/capstone

[23] Type 130 Miniature DC Motor, 2025. [Online]. Available: https://www.amazon.com/BOJACK-Pcs-Type-130-EK1450/dp/B09FPXF1QK?th=1. [Accessed: Mar 14, 2025].

[24] Type 280 Micro DC Motor, 2025. [Online]. Available: https://www.amazon.com/Motor-3V-12V-Micro-Boat-Model/dp/B01M0XOOS5/?_encoding=UTF8&pd_rd_w=Ul8M3&content-id=amzn1.sym.255b3518-6e7f-495c-8611-30a58648072e%3Aamzn1.symc.a68f4ca3-28dc-4388-a2cf-24672c480d8f&pf_rd_p=255b3518-6e7f-495c-8611-30a58648072e&pf_rd_r=KT194JSHZXCPAZ0BAM6T&pd_rd_wg=wWatr&pd_rd_r=3ce6530b-20f2-49a2-baf3-1796141b33b4&ref_=pd_hp_d_atf_ci_mcx_mr_ca_hp_atf_d. [Accessed: Mar 14, 2025].

[25] Hamidreza Marvi, James P. Cook, Jeffrey L. Streator, David L. Hu, Snakes move their scales to increase friction, Biotribology, Volume 5,2016, Pages 52-60,ISSN 2352-5738, https://doi.org/10.1016/j.biotri.2015.11.001.

[26] Mohammadi, Alireza. *Sonification MATLAB Code for Snake Robots*. February, 2024. Rbotic Motion Intelligence Lab at the University of Michigan-Dearborn.

[27] The Buck Institue for Educaiton, 2019. [Online]. *Gold Standard PBL – Seven Essential Project Design Elements*, Available: https://www.pblworks.org/what-is-pbl/gold-standard-project-design. [Accessed: Mar 14, 2025].

[28] Schwenk, M, Arras, K. O, "R2-D2 Reloaded: A flexible sound synthesis system for sonic human-robot interaction design," *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, Edinburgh, UK, 2014, pp. 161-167

[29] Kacem, A., Zbiss, K., Watta, P.*et al.*Wave space sonification of the folding pathways of protein molecules modeled as hyper-redundant robotic mechanisms. *Multimed Tools Appl* 83, 4929–4949 (2024). https://doi.org/10.1007/s11042-023-15385-y