Project Overview

&

Improvement Plan

For

Viper Vision

**Table of Contents**

[1. PROJECT OVERVIEW 1](#_Toc159655855)

[1.1. Brief Description of the Project 1](#_Toc159655856)

[1.2. Objectives and Goals 1](#_Toc159655857)

[2. TECHNOLOGY STACK 2](#_Toc159655858)

[2.1. Programming Languages Used 2](#_Toc159655859)

[2.2. Frameworks & Libraries 2](#_Toc159655860)

[2.2.1. Servo 2](#_Toc159655861)

[2.2.2. Adafruit PWM Servo Driver Library 2](#_Toc159655862)

[2.2.3. Wire 2](#_Toc159655863)

[2.2.4. ESP8266WiFi 3](#_Toc159655864)

[2.2.5. BlynkSimpleEsp8266 3](#_Toc159655865)

[2.3. Other Relevant Technologies 3](#_Toc159655866)

[2.3.1. Arduino IDE 1.8.X 3](#_Toc159655867)

[2.3.2. Blynk 3](#_Toc159655868)

[3. DESIGN ARCHITECTURE OVERVIEW 4](#_Toc159655869)

[3.1. High-level architecture of the system 4](#_Toc159655870)

[3.2. Components & Their Interactions 4](#_Toc159655871)

[3.2.1. Ultrasonic + Esp8266 4](#_Toc159655872)

[3.2.2. Servo Driver + Esp8266 4](#_Toc159655873)

[4. TIME AND RESOURCES 5](#_Toc159655874)

[4.1. Project Duration 5](#_Toc159655875)

[4.2. Team Members & Their Roles 5](#_Toc159655876)

[4.3. Breakdown of time spent on different phases 5](#_Toc159655877)

[4.4. Resources and Cost 6](#_Toc159655878)

[5. TESTING 7](#_Toc159655879)

[5.1. Unit Testing 7](#_Toc159655880)

[5.1.1. Overview 7](#_Toc159655881)

[5.1.2. Process 7](#_Toc159655882)

[5.1.3. Results 7](#_Toc159655883)

[5.2. Integration Testing 7](#_Toc159655884)

[5.2.1. Overview 7](#_Toc159655885)

[5.2.2. Process 7](#_Toc159655886)

[5.2.3. Results 8](#_Toc159655887)

[6. CODE FUNCTIONALITIES 9](#_Toc159655888)

[6.1. Overview of key functionalities/modules 9](#_Toc159655889)

[6.1.1. Snake Motion Control 9](#_Toc159655890)

[6.1.2. Blynk Control: 9](#_Toc159655891)

[6.1.3. Loop Function: 9](#_Toc159655892)

[6.1.4. Blynk Integration 9](#_Toc159655893)

[6.2. Explanation of critical algorithms or logic 9](#_Toc159655894)

[7. CHALLENGES AND LESSONS LEARNED 12](#_Toc159655895)

[7.1. Challenges faced during the project 12](#_Toc159655896)

[7.2. Lessons learned from both success and failures 12](#_Toc159655897)

[8. IMPROVEMENT SUGGESTION FOR V2 13](#_Toc159655898)

[8.1. Identified areas for improvement 13](#_Toc159655899)

[8.1.1. Code Structure and Readability: 13](#_Toc159655900)

[8.1.2. Error Handling: 13](#_Toc159655901)

[8.1.3. Ultrasonic Sensor Calibration: 13](#_Toc159655902)

[8.1.4. Loose Wires: 13](#_Toc159655903)

[8.2. Feature enhancements 13](#_Toc159655904)

[8.2.1. Increase number of Servos 13](#_Toc159655905)

[8.2.2. Striking Pose and Striking: 13](#_Toc159655906)

[8.2.3. Remote Control Modes: 14](#_Toc159655907)

[8.2.4. Mouth, Tongue, Fangs, Eyes, and Hissing Sound: 14](#_Toc159655908)

[8.2.5. Redesign of 3D Printed Segments: 14](#_Toc159655909)

[8.3. Technical debt to address 14](#_Toc159655910)

[8.3.1. Redundant Delays: 14](#_Toc159655911)

[8.4. Performance optimizations 14](#_Toc159655912)

[8.4.1. Sensor Readings Frequency: 14](#_Toc159655913)

[8.4.2. Servo Motion Optimization: 14](#_Toc159655914)

[8.4.3. Power Efficiency: 15](#_Toc159655915)

[8.4.4. Balance of Center of Gravity: 15](#_Toc159655916)

[8.4.5. Increase Wheel Grip: 15](#_Toc159655917)

[9. CONCLUSION 16](#_Toc159655918)

[10. APPENDICES 17](#_Toc159655919)

[11. VERSION HISTORY 18](#_Toc159655920)

# PROJECT OVERVIEW

## Brief Description of the Project

The Viper Vision project is an innovative endeavor aimed at developing a versatile and agile robotic system designed for surveillance and reconnaissance applications. Inspired by the flexibility and adaptability of snakes in navigating complex environments, this cutting-edge robot employs a serpentine structure to navigate through tight spaces, crawl around obstacles, and access difficult-to-reach areas with ease.

Equipped with sensors and cameras, the viper vision is capable of capturing images and video footage, providing real-time data for surveillance purposes. Its modular design allows for customization based on specific mission requirements, enabling the integration of various sensors, communication devices, and other technologies to enhance its capabilities.

The project focuses on addressing challenges related to urban surveillance, disaster response, and security operations where traditional robotic platforms may face limitations. The snake robot's ability to slither through confined spaces and navigate uneven terrain makes it a valuable asset for search and rescue missions, monitoring critical infrastructure, and enhancing situational awareness in various environments.

Additionally, the surveillance snake robot leverages advanced artificial intelligence algorithms for autonomous navigation, obstacle avoidance reducing the need for constant human intervention. This aspect enhances its operational efficiency and responsiveness in dynamic and unpredictable scenarios.

## Objectives and Goals

Here, I have mentioned some of the key objectives and goals for making this project come to life. Such as:

* Enhance surveillance capabilities
* High quality Imaging
* Remote operation and communication
* Search and rescue focus
* Situation awareness
* Mimic snake locomotion

# TECHNOLOGY STACK

## Programming Languages Used

List the programming languages used in the project, along with their versions.

* **C++**

In the development of the Viper Vision project, the technology stack encompasses C++ programming languages tailored to different components of the system. C++ is a versatile and efficient language, well-suited for embedded systems like those found in the Arduino microcontroller. It allows for direct hardware manipulation, making it ideal for programming tasks that require precise control over the robot's sensors, actuators, and overall behavior.

In the context of the Viper Vision, C++ is employed to code the firmware that governs the robot's motion, sensor integration, and communication with other modules. The language's proximity to hardware and its efficiency make it an excellent choice for ensuring that the robot's movements are responsive, accurate, and synchronized with the data gathered by its sensors.

By leveraging the strengths of C++ programming in the Arduino environment, the Viper Vision project can achieve the necessary precision and efficiency required for successful navigation, surveillance, and interaction within its operational environment. This highlights the strategic selection of programming languages within the technology stack to optimize performance across diverse aspects of the robotic system.

## Frameworks & Libraries

Some of the frameworks and libraries used along with their version in this project are given below:

### Servo

* **Purpose:** This C library allows an Arduino board to control servo motors.
* **Version:** 1.2.1

### Adafruit PWM Servo Driver Library

* **Purpose:** This library is the perfect solution for any project that requires a lot of servos.
* **Version:** 3.0.1

### Wire

* **Purpose:** This library allows you to communicate with I2C devices, a feature that is present on all Arduino boards

### ESP8266WiFi

* **Purpose:** ESP8266 is all about Wi-Fi. It allow to connect new ESP8266 module to a Wi-Fi network to start sending and receiving data.

### BlynkSimpleEsp8266

* **Purpose:**It allows you to connect your ESP8266-based board to the Blynk app and control it remotely.

## Other Relevant Technologies

### Arduino IDE 1.8.X

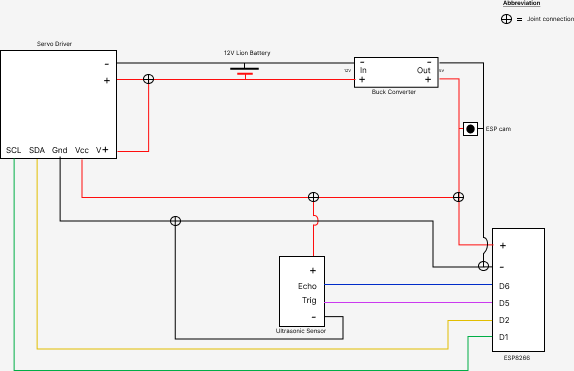
Arduino Integrated Development Environment (IDE) is an open-source software platform designed for programming Arduino microcontrollers. It provides a user-friendly interface for writing, compiling, and uploading code to Arduino boards. The IDE supports the C and C++ programming languages, making it accessible for both beginners and experienced developers. With built-in functions and libraries, it simplifies the process of interacting with Arduino hardware components, allowing users to create a wide range of projects, from simple LED blinkers to complex robotics applications.

### Blynk

Blynk is a versatile and user-friendly Internet of Things (IoT) platform that simplifies the development of connected projects. It consists of a mobile app and a cloud service, allowing users to control and monitor IoT devices using smartphones. Blynk enables quick prototyping and customization of IoT applications, facilitating the creation of interactive and remotely accessible projects.

# DESIGN ARCHITECTURE OVERVIEW

## High-level architecture of the system



*Fig: Circuit Overview*

## Components & Their Interactions

### Ultrasonic + Esp8266

**Ultrasonic Esp8266**

Trig ================> D5

Echo ================> D6

Vcc ================> Vin

Gnd ================> G

### Servo Driver + Esp8266

**Servo Driver Esp8266**

SDA ================> D2

SCL ================> D1

Vcc ================> Vin

Gnd ================> G

# TIME AND RESOURCES

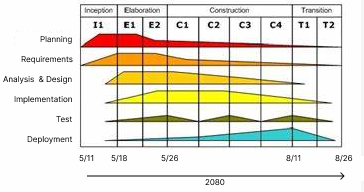
## Project Duration

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Name** | **Start Date** | **End Date** | **Total Duration** |
| Viper Vision | 2080-05-11 | 2080-08-26 | 3 months |

## Team Members & Their Roles

|  |  |
| --- | --- |
| **Team Members** | **Roles** |
| Anup Maharjan | Project Leader, Developer, Tester |
| Anish Shrestha | Resources Manager |
| Rima Awal | Design Checker |
| Abiral Shrestha | Helper |
| Bijen Risal | Helper |

## Breakdown of time spent on different phases



*Fig: SDLC of Viper Vision*

## Resources and Cost

|  |  |  |  |
| --- | --- | --- | --- |
| **Resources** | **Quantity** | **Cost / piece** | **Total** |
| ESP8266 | 2 | 800 | 1600 |
| Ultrasonic Sensor | 1 | 250 | 250 |
| ESP32 Cam | 1 | 1600 | 1600 |
| MG995R servo | 10 | 400 | 4000 |
| Zip ties | 10 | 10 | 100 |
| Gauge wire | 20 | 10 | 200 |
| 12V Li-ion Battery | 1 | 7000 | 7000 |
| wheels | 10 | 350 | 3500 |
| Servo Brackets | 10 | 550 | 5500 |
| Buck Converter | 1 | 190 | 190 |
| Electrical Tape | 1 | 20 | 20 |
| screws | 20 | -­ | 100 |
| FTDI usb to ttl | 1 | 600 | 600 |
| Double sided tape | 1 | 120 | 120 |
| 3D printing (didn’t use in this model) |  | 50 per gram |  |
| **Total Cost** | **-** | **-** | **24,780 /-** |

# TESTING

## Unit Testing

### Overview

Unit testing is a software development practice where individual units or components of a program are tested in isolation to ensure they function as intended. Developers create test cases to verify the correctness of each unit, aiming to identify and fix bugs early in the development process. This approach enhances code reliability, facilitates easier debugging, and contributes to overall software stability by isolating and validating discrete functionalities. Unit testing is a fundamental part of the larger testing framework that ensures the robustness of software applications.

### Process

The unit testing started along with the development process. First of all, the testing of the ESP8266 board is done in order to ensure that the board is in working condition and issues caused in near future is not due and damaged board. Following the board, the testing of the core functionalities of Viper Vision i.e. snake locomotion, ultrasonic data, servo driver esp32 cam.

### Results

The unit testing made sure to provide us assurance that the each modules/components of Viper Vision is in working conditions and works as their functionalities are intended to in specific conditions.

## Integration Testing

### Overview

Integration testing is a software testing phase where different modules or components of a system are combined and tested as a group. Its purpose is to detect interface and interaction issues between integrated components, ensuring they work seamlessly together. By validating the collaboration of interconnected units, integration testing verifies the overall system's functionality, performance, and data flow. This testing phase is crucial for identifying and resolving integration-related issues before deploying the software in a real-world environment

### Process

The integration testing started right after the completion of unit testing. The incremental integration testing is used for this project where testing is done by integrating two or more modules that are logically related to each other and then tested for proper functioning of the application.

### Results

This made the tester to easily find the module that is sources of issues if any issues were occurred while integrating the components. As a result, it became much faster to reduce the time required to complete the project.

# CODE FUNCTIONALITIES

## Overview of key functionalities/modules

### Snake Motion Control

* The slither function generates a sinusoidal wave for snake slithering motion with parameters for offset, amplitude, speed, and wavelength.
* The staticWave function generates a static sinusoidal wave with similar parameters.
* The ring function produces a circular motion by controlling the servo angles.

### Blynk Control:

* The device can be controlled through the Blynk app using a virtual button (V3).
* The BLYNK\_WRITE function handles changes in the Blynk app and updates the initial\_value variable accordingly.

### Loop Function:

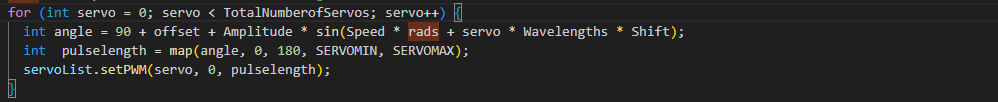
* The main loop constantly runs and checks the value of initial\_value.
* Based on the selected mode (initial\_value), the code executes different snake motions, reads the ultrasonic sensor, and performs specific actions.

### Blynk Integration

* The code sends the current **initial\_value** to the Blynk app, updating the corresponding virtual display widget (V2).

## Explanation of critical algorithms or logic

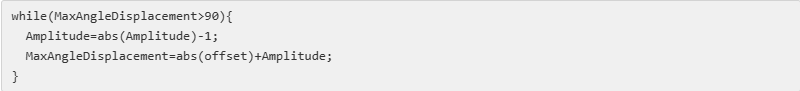
There is a question about how we program the snake? The short answer is we pump a sine wave through the servo chain. But for a more detailed breakdown of how exactly that works let’s look at the code.

This line writes to each of the 10 servos a sine wave. The base line angle is 90 degrees, the offset variable will control if the snake goes forward (offset=0) or turns left or right (offset=10 or -10). The sine wave outputs a value between [-1, 1], this value can be scaled up by multiplying by the amplitude. 

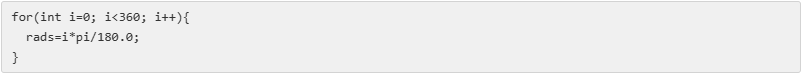
Pulselength return the pulse length for the servo for provided angle through arguments. This is so that the servo don’t rotate more than 90 degree and this step is needed to make it move more than 90 degree. Minimum and maximum pulse length of the servo should be define as ‘SERVOMIN’ and ‘SERVOMAX’



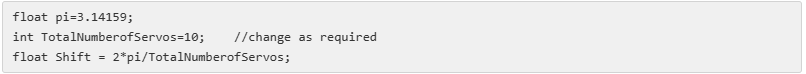
Since the servo has a range of [0,180] degrees we must ensure that the above values don't give an output below 0 or above 180. The following while loop is used to constrain the amplitude to within these bounds. Mathematically we must satisfy this condition: |offset|+|amplitude|<=90



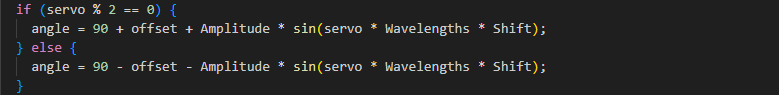
To get the desired output from the sine wave we must use radians instead of degrees. Essentially 2\*pi radians = 360 degrees. The following line makes this conversion.



To get the desired output from the sine wave we must use radians instead of degrees. If you are unsure what radians are here is a brief intro. Essentially 2\*pi radians = 360 degrees. The following line makes this conversion.



If-else is used inside the loop because the servos are place alternatively in this project such as one servo is faced downwards and another servo is faced upwards, making it necessary for the change in angle as well.



Here, in second case in switch, distance is measured through the ultrasonic sensor. Depending on the distance measured by the ultrasonic sensor, the snake performs different movements. For example, turning right when an object is detected within a certain range or moving forward otherwise.



We have now set the foundational work. I encourage you to play around with changing the values of the following variables: Amplitude, Speed and Wavelengths to see the effect they have on the sine wave output.

# CHALLENGES AND LESSONS LEARNED

## Challenges faced during the project

There were many challenges while producing this project. Some were minor while other were major that took up even 2 weeks for our team to solve that problem. Here, below are the challenges that our team faced during development process:

1. **Mechanical Design:** Designing a snake robot with the right flexibility, size, and durability for effective surveillance in diverse environments.
2. **Power Management:** Balancing the need for a compact design with sufficient power to drive the servos, sensors, and communication modules.
3. **Wireless Communication Stability:** Ensuring reliable communication between the snake robot and the central control system, especially in areas with potential signal interference.
4. **Cost Constraints:** Developing a cost-effective Viper vision while ensuring the inclusion of necessary features and capabilities.
5. **Damaged Products:** The components should be checked before the starting the project. Also, it is good to keep working components for backup even if it is only for the testing purpose.

## Lessons learned from both success and failures

1. **Interdisciplinary Collaboration:** Successful IoT projects require collaboration between team members with different specialties. Encourage interdisciplinary teamwork for holistic problem-solving.
2. **Real-world Testing Importance:** Simulations are valuable, but real-world testing is indispensable. Unexpected challenges may arise in actual environments, highlighting the importance of thorough testing in diverse conditions.
3. **Scalability Consideration:** Design the project with scalability in mind. Consider future enhancements and the ability to integrate additional sensors or features as technology evolves.
4. **Documentation and Knowledge Transfer:** Maintain comprehensive documentation for the project. This includes code comments, schematics, and a user manual. Facilitate knowledge transfer within the team and for future developers.

# IMPROVEMENT SUGGESTION FOR V2

## Identified areas for improvement

### Code Structure and Readability:

* **Issue:** The code could benefit from better organization and comments to improve readability and maintainability.
* **Suggestion:** Add comments to describe complex sections of code, and consider breaking down functions into smaller, more modular components for better readability.

### Error Handling:

* **Issue:** The code currently lacks robust error handling mechanisms.
* **Suggestion:** Implement error handling for critical operations, such as Blynk initialization, sensor readings, and servo movements. Provide meaningful error messages or logs for debugging.

### Ultrasonic Sensor Calibration:

* **Issue:** The ultrasonic sensor readings might require calibration for accurate distance measurements.
* **Suggestion:** Implement a calibration routine or constants for adjusting distance calculations based on real-world testing.

### Loose Wires:

* **Issue:** The presence of loose wires can lead to connection issues, shorts, or accidental disconnections.
* **Suggestion:** Secure all connections and use appropriate cable management techniques to prevent loose wires. Establish a dedicated designer team to address wiring concerns and ensure proper cable management throughout the project.

## Feature enhancements

### Increase number of Servos

* **Enhancement:** Increase the number of servos and extend the length of the snake for enhanced maneuverability and coverage.
* **Suggestion:** Redesign the mechanical structure to accommodate additional servos and segments while maintaining a compact design. Consider optimizing the distribution of servos for balanced motion.

### Striking Pose and Striking:

* Enhancement: Implement striking poses and striking motions for realistic snake behavior.
* Suggestion: Develop predefined striking poses and dynamic striking motions triggered by specific stimuli or user commands. Ensure smooth and coordinated servo movements for lifelike striking actions.

### Remote Control Modes:

* **Enhancement:** Expand the number of remote control modes in the Blynk app.
* **Suggestion:** Add more modes that introduce different snake movements or behaviors. Enhance the Blynk app to provide a user-friendly interface for selecting these modes.

### Mouth, Tongue, Fangs, Eyes, and Hissing Sound:

* **Enhancement:** Add realistic features such as a mouth, tongue, fangs, eyes, and hissing sound to enhance the snake robot's appearance and interactive capabilities.
* **Suggestion:** Integrate animatronic components for lifelike movements and sounds. Utilize sensors or user inputs to trigger interactive behaviors such as hissing or striking poses.

### Redesign of 3D Printed Segments:

* **Enhancement:** Redesign 3D printed segments to optimize durability, flexibility, and aesthetic appeal.
* **Suggestion:** Evaluate the existing design for potential improvements in material selection, structural integrity, and ease of assembly. Make sure that the design works perfectly with the snake you are developing.

## Technical debt to address

### Redundant Delays:

* **Issue:** The use of delays in certain parts of the code may introduce responsiveness issues.
* **Suggestion:** Refactor the code to use non-blocking techniques to avoid delays and maintain better responsiveness.

## Performance optimizations

### Sensor Readings Frequency:

* **Issue:** The code reads the ultrasonic sensor in a blocking manner within the loop.
* **Suggestion:** Implement non-blocking sensor readings with a proper interval to avoid potential delays in the main loop.

### Servo Motion Optimization:

* **Issue:** Continuous servo movements in the loop may lead to jerky or inefficient motion.
* **Suggestion:** Implement smooth servo motion by using gears that doesn’t lose its grips. Make sure to use metallic gears.

### Power Efficiency:

* **Issue:** The code does not currently address power efficiency.
* **Suggestion:** Explore ways to put the microcontroller or components to sleep during idle periods to conserve power. Hint: link adding a power button or other methods

### Balance of Center of Gravity:

* **Issue:** Imbalanced weight distribution may affect the stability and maneuverability of the snake robot.
* **Suggestion:** Redesign the mechanical structure to achieve a balanced center of gravity. Distribute components and payloads evenly to minimize the risk of tipping or instability during operation like going backwards instead of forwards.

### Increase Wheel Grip:

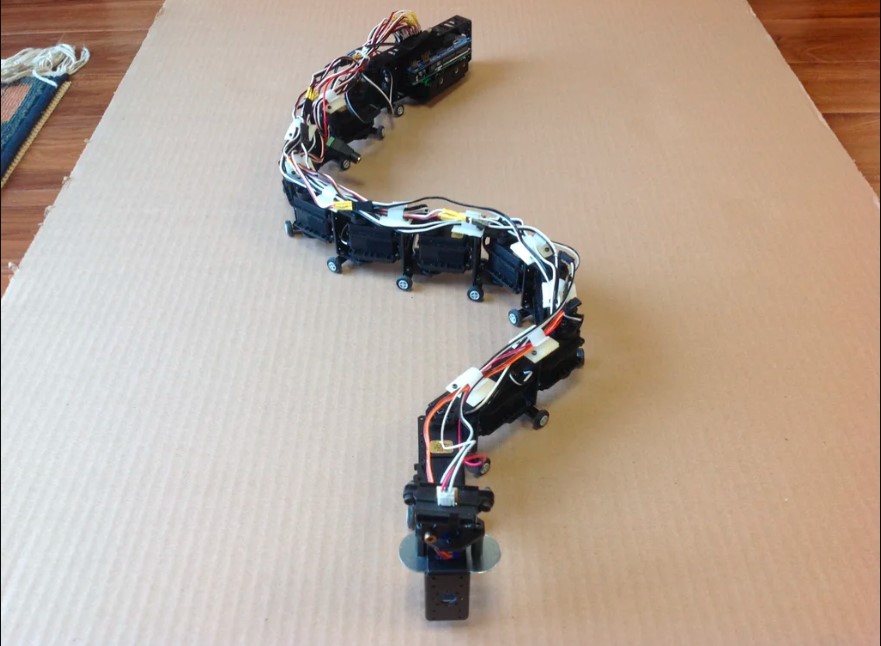
* **Issue:** Insufficient wheel grip may result in reduced traction and maneuverability, especially on slippery surfaces.
* **Suggestion:** Enhance wheel grip by using high-traction tires or implementing traction control mechanisms. Evaluate surface conditions and adjust wheel materials or configurations accordingly to improve grip and overall performance.

# CONCLUSION

In conclusion, the Viper Vision project represents a significant stride in the realm of robotic innovation, providing a versatile and adaptable solution for surveillance and reconnaissance in challenging environments. The meticulous development process, incorporating a serpentine structure for agile navigation and advanced technologies like C programming on Arduino, underscores the project's commitment to efficiency and precision.

As the project concludes, the Viper Vision stands poised as a pioneering example of human-robot collaboration, offering a promising avenue for addressing real-world challenges in security, disaster response, and beyond. The achievements and lessons learned from this endeavor pave the way for future advancements in the field of robotics and autonomous systems.

# APPENDICES





# VERSION HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Changes** | **Modules/Components Affected** | **Contributors** |
| V1 | 2080-10-07 | - This is the 1st version and newly created | - All List of modules/components | - Anup Maharjan |