Unmanned Aerial Vehicle Ground Control Station

By

1Tori Gardner, 1Adolfo Mendiola, and 2Edgar Moreno

Senior Design Project Presented to the Department of Engineering
In Partial Fulfillment of the Requirements
For the Degree of

Bachelor of Science
In
1SOFTWARE ENGINEERING
2COMPUTER ENGINEERING

San Antonio, Texas
May 2021

Supervising Advisor:
Dr. Ben Abbott
INSTRUCTOROF ELECTRICAL ENGINEERING
ABSTRACT

Unmanned aerial vehicles, often referred to as UAVs or drones, are a rapidly growing technology applied to solve a plethora of challenges in today’s fast-paced world. They are being used in law enforcement for surveillance and search missions, by the military for tactical reconnaissance, and to responding to and managing environmental issues and disasters among an ever-increasing set of applications. Developing UAV systems is challenging due to the real-time nature of its operation. Seconds— and sub-seconds—count when piloting a UAV in a dynamic environment remotely, so the existence of a user-friendly, efficient, and functional ground control station (GCS) is vital. In this project, we proposed and designed the underlying architecture and user interface of a ground control station for St. Mary’s University’s UAV program. We implemented basic functions such as the passing of parameters to and from a simulated real-time database, emergency preset function, and preset layout-based window swapping to eliminate hiding time-critical data with other windows such as in the case of other commercial GCSs.
ACKNOWLEDGMENTS

As a team, we would like to thank our supervising professor, Dr. Ben Abbott, for the invaluable guidance and relentless honesty. His advice and opinions regarding our solutions honed our ability to think like engineers and challenged us to go the distance to develop a system that St. Mary’s can use for years to come. We also thank an anonymous graduate student for offering his experience in QtPy5 and drone-oriented terminology which proved to expedite solutions to the greatest struggles we experienced during development. Finally, we want to thank each professor whose passion for teaching and advice allowed us to grow as upcoming engineers and prepare us not only for this moment at which senior design comes to a close, but also for our future careers that will carry us for all of our lives.
# TABLE OF CONTENTS

1. Abstract ............................................................................................................................... i 
2. Acknowledgements .............................................................................................................. ii 
3. Table of Contents ............................................................................................................... iii 
4. List of Figures ....................................................................................................................... v 
5. List of Tables ........................................................................................................................ vi 

1. Introduction ............................................................................................................................. 1 
   1.1. Problem Statement ............................................................................................................ 1 
   1.2. Objectives ....................................................................................................................... 1 
   1.3. Literature Search ............................................................................................................. 2 
   1.4. Problem Constraints, Requirements and Specifications .................................................... 4 
   1.5. Solution Approach ......................................................................................................... 5 
2. Summary of Engineering Method ............................................................................................ 6 
3. Iterative Design Summary ..................................................................................................... 7 
4. Overview of System ............................................................................................................... 8 
   4.1. System Architecture ....................................................................................................... 8 
      4.1.1. Internal Architecture ................................................................................................. 10 
   4.2. GUI Design .................................................................................................................... 12 
   4.3. File-Based Parameter Settings ....................................................................................... 13 
   4.4. Multi-Window Layout Design ......................................................................................... 14 
5. Unexpected Problems and Solutions .................................................................................... 15
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. Results</td>
<td>16</td>
</tr>
<tr>
<td>7. Economical, Public Health, Safety, Welfare, and Environmental Analysis of Results</td>
<td>18</td>
</tr>
<tr>
<td>8. Further Implementation</td>
<td>19</td>
</tr>
<tr>
<td>9. References</td>
<td>22</td>
</tr>
<tr>
<td>10. Appendices</td>
<td>23</td>
</tr>
<tr>
<td>11. SMC Capstone Reflections</td>
<td>24</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1: Context Diagram of GCS..................................................................................9
Figure 2: UML Diagram of GCS.......................................................................................11
Figure 3: High Fidelity Wireframe....................................................................................12
Figure 4: Main Window ....................................................................................................17
Figure 5: Emergency Settings Menu.................................................................................17
Figure 6: Layout Customizer .............................................................................................18
LIST OF TABLES

Table 1: Division of Labor ........................................................................................................6
1. INTRODUCTION

1.1. PROBLEM STATEMENT

St. Mary's University (StMU) has recently expanded on the UAV program with the addition of a UAV laboratory as of Fall 2020. This expansion comes with the expectation of UAV-centric programs and students interested in pursuing an engineering degree with a focus on UAVs. Our customers, Dr. Bahman Rezaie and Dr. Dante Tezza, are in need of a new ground control station to manage and pilot their current users and UAVs as well as future additions. The requested software must be home-grown and controlled through a graphical user interface (GUI) intended for professor and student use. The system must be scalable such that additional general UAVs and controllers of various types can be connected and flown from the GUI. Although at this time the system is intended for university use, the GCS will eventually be considered for commercial distribution after sufficient development— an expectation that must be accounted for throughout the duration of this project.

1.2. OBJECTIVES

We delivered various artifacts of documentation listed below, as well as the software package developed. The software package consists of the source code, executable application, and virtual machine files accessible through a GitHub repository. Additionally, we delivered the test cases and corresponding results as well as a user manual. All documentation and supplementary artifacts can be found on our Confluence account.
List of Deliverables:

Software Requirements Specification (SRS)
  i. Requirements List
  ii. Vision Document
  iii. Personas
  iv. User Stories
  v. Use Cases

Design Documents
  i. Paper Prototype
  ii. Low and High-Fidelity Wireframes
  iii. UML Diagram
  iv. Context Diagram

Source Code and Virtual Machine Files

Test Cases and Results

User Manual

1.3. LITERATURE SEARCH

There are various open-source ground control stations on the market that are used for UAV flight and management. The most popular open-source options are Qgroundcontrol, Px4 autopilot, and UgCs. Qgroundcontrol is an open-source full-flight control and mission planning software that uses the MAVLink protocol to control the UAVs that are supported. The program
is cross-platform but is limited to a small subset of supported UAVs. The other project is a fully open-source flight control software for UAVs that provides a software stack to integrate with additional programs into other products to help maintain considerable hardware support and keep it scalable, but it is limited to Linux as a nonprofit and can be connected to Qgroundcontrol, and MAVLink. UgCS is a mission planner meant for land surveying and allows an extremely limited set of consumer drones. It allows for basic UAV ground control stations and for offline processing and communication with the UAV but is limited to a subset of drones and task sets such as data capture.

Drone ground control station with enhanced safety features is a system created with a combination of software and hardware elements that allows a user to control a drone and get telemetry data in an encrypted and secure manner using MavLink for basic drone communication with the security features activated as well as a hardware encoded chip that needs to be plugged into the drone as well as a controller to make sure no unauthorized user can access the drone mid-flight. “The software has various features in which it shows no-fly zones, Wind speed, GPS, altitude, as well as other various telemetry systems” (Haque et. al, 2017). The system gets all its data from no-fly government servers as well as various telemetry data from the drone itself. This is to give the software the ability to alert the pilot on what they can or cannot do. UB-ANC emulator is an in-development program in which the final goal is to create a system framework for multiple drones to work together with simple changes unlike the current system known as MavLink. “MavLink which allows users to communicate with a drone but only one at a time per program instance which is very taxing on hardware as well as a connection for a drone control station” (Modares, J, 2016). UB-ANC plans to fix this issue and implement a communication flight controller to communicate with GCS more effectively and efficiently.
The software cycle for a ground control station takes the same approach as many other ground control systems when it comes to determining the functionality and determining the factors involved in the Ground Control Station. The TreK project development from Nasa provides the ideology on how to approach our Ground Control Station development (Hendrix et. al, 2002).

1.4. PROBLEM CONTRAINTS, REQUIREMENTS, AND SPECIFICATIONS

The GCS project worked in parallel with the Drone to Ground Data Interchange (DGDI) team whose scope was the UAV – database communication and was constrained in part by their milestones and obstacles. Each aspect of the total project divided between the two teams needed to be completed and integrated by the end of senior design. To manage this cross-team communication and progress, we created a joint Jira account for requirements and task tracking as well as a Confluence account for documentation and version management and control. For project management, we created and updated a Gant Chart to manage deadlines across both teams.

Time constraints were our greatest obstacle as the curriculum year was condensed due to the pandemic and we were limited regarding access to engineering department professors and laboratory resources. Additionally, our team and the DGDI team primarily worked remotely, which caused communication deficiencies that we worked diligently to overcome. To address these circumstances, the two teams ensured heightened communication methods and met frequently using Zoom with each other, the customers, and the senior design supervisor. Joint
meetings were encouraged on occasion to ensure all stakeholders were aware of the development progress throughout the duration of the project.

Hardware constraints included the Linux-based computer and custom raspberry pi UAV that was provided to us by our customers. Though the overall system must have a pathway to be cross-platform, we were required to build a system that is able to run on the provided hardware.

Requirements and system specifications can be found at the StMU UAV Shared GCS Documents Confluence account.

1.5. SOLUTION APPROACH

As per the customer’s requirements, the GUI must be developed with a pathway to be scalable so that users, UAVs, and controllers can be added to the system as the UAV laboratory grows. The GUI’s greater context was designed to interface with the UAVs through a database that stores the UAV data provided by the DGDI team. This design decision deviates from the traditional communication model in which a UAV communicates directly to its GCS through a local and shared file system. The GUI will be a multi-windowed display containing UAV telemetry and sensor data as well as graphs of the appropriate data during operation in near-real-time. The system will allow the user to tune the UAV for flight by providing a set parameters function. The system will allow the user to save parameter sets as presets, called drone flight modes, which can be switched between during flight through the GUI. The user will be able to pilot the UAV by sending commands through the GUI and into the database. Additionally, the
GUI will allow the user to plan and monitor UAV autonomous missions, displaying the UAV’s past, present, and future waypoints. The software must also be portable to a field laptop.

Due to the size and magnitude of this project, it will be a multi-generation senior design. The purpose and goal of this iteration of the project is to design and implement a foundational level architecture and GUI as well as prove the concept of UAV-GCS data transfer and sharing through the implementation of the database. As a team of three, we have divided the work by compiling a list of common software development cycle tasks and allocating tasks to the best fit member. A summary of the division of work is found in Table 1 below.

Table 1: Division of Labor

<table>
<thead>
<tr>
<th>Member</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tori Gardner (SE)</td>
<td>Project management, artifact management, business analyst, GUI design and implementation</td>
</tr>
<tr>
<td>Adolfo Mendiola (SE)</td>
<td>Source code version management, GUI design and implementation, logical layer implementation</td>
</tr>
<tr>
<td>Edgar Moreno (CE)</td>
<td>Testing, GUI implementation, logical layer implementation</td>
</tr>
</tbody>
</table>

2. SUMMARY OF ENGINEERING METHOD

The classic software development cycle groups development into requirements, analysis, design, implementation, and testing phases. For the development within this project’s scope, our team iteratively progressed through each of these phases until the scope was realized.
We used an Agile approach and the Scrum methodology to develop the GCS software. Scrum was a perfect fit for our circumstances since we are a small team and had limited time to complete the software package. We developed a product backlog from our collected requirements and worked in short, one-week sprints in which each member developed in parallel on allocated tasks across the development cycle. To manage the product and Sprint backlogs, we used Atlassian Jira which allowed the project manager to create and start sprints as well as assign tasks to members of the team.

3. ITERATIVE DESIGN SUMMARY

We developed in an iterative fashion in which we progressed multiple artifacts simultaneously, beginning with simplified versions, which progressively increased in detail and complexity as well as gained a broader feature set with every iteration until the final system was completed. Providing user value was integral to the requirement-driven development as well as important to us as developers; we sought to provide the customers with a minimal viable product early on and frequently throughout development so that we could gain feedback for future development which we borrowed from the Lean methodology. We also kept a tight feedback loop with our supervisor and customers so that we could discover and fix errors early to keep our development consistent with the customer’s requests, so we met with the customers weekly. With every other iteration, we provided the customer with a minimum marketable feature to maintain this feedback and add recognizable value.
4. OVERVIEW OF SYSTEM

Since this project is foundational in the development of the UAV lab’s GCS, we were tasked with designing and implementing the underlying architecture of the system as well as implementing proof of concept level GCS-database communication. Our scope was set to document a complete set of requirements from the customers, fully design the system to account for each requirement, and implement a set of requirements with the expectation that the system will be fully realized with at least one additional senior design project.

We began by performing several requirements elicitation meetings with the customers to ensure the subsequent development would be customer-centric and produce a valuable product for the customers. From the requirements gathered, we authored three personas representing the user base in the categories of university program directors and other administrative positions, university students, and miscellaneous researchers and freelancers. From these three personas and requirements elicitation meetings with the customers, we collected eleven user stories and twenty-one use cases, documentation for all of which can be found in detail on the project’s Confluence page linked in the appendix.

4.1. SYSTEM ARCHITECTURE

Our team worked closely with the customers to improve upon the traditional file-based UAV-GCS communication. In most publicly available GCSs, the UAV is loaded with a local file containing the default parameter setup, emergency procedures, and other necessary data in the event that the UAV is disconnected from the GCS. The UAV dynamically sends its telemetry to the GCS on which it is displayed. The GCS then processes and manages this data in its own file
system that contains additional files regarding UAV processes, flight modes, flight data, etc. The approved solution for improving this system architecture was to implement a database to act as the intermediary between the GCS and the UAV as illustrated in the context diagram below in Figure 1.

![Figure 1: Context Diagram of GCS](image)

Using a database allows for greater and more complex storage of data and files without increasing the manageability that a user may experience with a convoluted file system. This results in the GCS software having the data management and storage to support more complex features such as user and UAV management rather than being limited to sending and receiving UAV flight data in the case of traditional GCSs. Even so, being the baseline functionality of a GCS, the communication was a primary concern. Because the UAV is sending telemetry to the database and the GCS is pulling this same information to display on the monitor continuously during flight, the user is receiving UAV telemetry in near real-time. This is similarly true in the case of the GCS sending parameters into the database and the UAV pulling the same information...
to receive commands continuously. The customers approved this context design and our team progressed to designing the GCS to support the improvements required.

The customers required the GCS to provide the university with a platform on which future programs and other university students could use the UAVs. This required characteristics and features absent from other GCSs such as user organization and security. Since the database is able to store more forms of data than what only pertains to the UAV in flight, it also allows the GCS to support entities such as multiple users and their read/write permissions, UAVs and their particular files, among other entities used in larger UAV programs. The solution further supports requirements such as scalability as new entities can easily be added to the database and portability. Any system that can connect to the database and run the GCS can manage and pilot UAVs. This architecture also lowers the entry barrier in that the user will no longer be required to load and save files to and from the UAV directly. All file, user, UAV, and other entity management can be achieved from the user interface.

4.1.1. INTERNAL ARCHITECTURE

Internal to the GCS software, the customers’ requirements implied the existence of several concepts that could easily be conceptualized as objects that made the object-oriented approach the best fit. We developed a UML diagram to illustrate the communication between the GUI objects and the logical objects shown in the figure below.
The GUI classes represent the distinct pages required by the customers which inherit the Qt container’s attributes since we used QtPy5 and the accompanying designer as the development environment. The logical classes represent the blueprint for the objects involved in the logic of the system and are related to the GUI in that they, as a system, update the GUI pages with their information. Using this internal architecture, we support the fulfillment of the scalability and security requirements at a foundational level. There can be many instances of the user and UAV objects and each can have attributes implemented for security reasons such as ID. Metrics such as those required by our customers cannot necessarily be added on as an afterthought during development— they must be built into the software from the beginning— so it was imperative that we account for these requirements at this level.
4.2 GUI DESIGN

While designing the GCS software architecture, it was necessary to design the system’s user interface early on to account for the features denoted in the requirements. The high-fidelity wireframe in Figure 3 illustrates the pilot x 4 window layout option on a single monitor.

![High Fidelity Wireframe](image)

Figure 3: High Fidelity Wireframe

The customers prioritized drone safety features such as an emergency preset button intended to reset UAV parameters to a safe state while in flight with a single button and a transparent message board overlay intended to display color-coded warnings over the camera/map which is where the pilot will be looking most frequently. Both of these entities are immutable and cannot be covered. Because UAV-GCS communication operates in real-time, a
primary concern for GCS users is how frequently and for how long the pilot must look away from the UAV’s position on the map or view through its camera. The longer and more frequently a pilot must do so, the riskier the flight. For this reason, the customers approved a new method of displaying data in a multi-windowed format, the layout of which can be customized and swapped with a single button. The pilot window is designed to exist constantly on the left half of at least one monitor and/or maximized to at least one monitor. This prevents a pilot window from being muted or covered by another window on all monitors, ensuring it is always visible to the flight team. For the same reasons, the right-hand side of the wireframe illustrates one of up to six layout presents that allow the user to swap through different windows and combinations of windows (such as the parameters, payload, graphing, and mission planner windows) without deviating from the pilot window. The function of each layout preset button is customizable, allowing the user to decide what window, or combination of windows, is set when a preset is selected.

4.3. FILE-BASED PARAMETER SETTINGS

There exists data, such as initial and emergency parameters, that are not associated with the UAV’s flight telemetry, are set before a flight, are UAV specific, and must be downloadable and uploadable. For this reason, the GCS uses a file system to load initial parameter data into their appropriate locations in the database for the connected UAV’s use before a flight. There are also datasets that must be dynamically loadable during flight. To accommodate this, the file system also allows a user to load a new set of parameters in the event of changing flight modes, and load emergency parameters in the event of an emergency during flight with a single button.
for each function. Files containing parameter data for emergency parameters and up to five flight modes can be uploaded to the GCS in the settings window and automatically connected to the emergency button and flight mode buttons, respectively. A file containing the initial parameter data will automatically be uploaded to the database.

During development, the milestones of the DGDI team and our team became unaligned which resulted in the GCS being ready to connect to the database before it was delivered. To prevent stunting the progress of the GCS, our team received permission to simulate the database using a basic file system to which we upload UAV parameter file information as if it were the database.

4.4. MULTI-WINDOW LAYOUT DESIGN

One of the biggest customer concerns was having the necessary flight data visible to the user without searching on multiple windows to minimize the time that critical data goes unmonitored. The multi-window layout preset solution required the use of the Qt-Advanced-Docking-System (QtADS), an open-source plugin we discovered on GitHub. Basic docking widgets provided by the Qt framework were insufficient for the customer’s requirements as the documentation states they can only be docked around the central widget. QtADS eliminates this constraint by allowing their docking widget to be docked and floated anywhere on the application. The ADS became the basis for the GUI’s architecture in which each container, implemented as advanced docks, has its own dock manager that allows any widget to be plugged in and displayed within the container bounds. We are able to save the state and geometry of the
containers which allows widget orientations to be saved in the layout customizer and loaded as the corresponding layout preset.

5. UNEXPECTED PROBLEMS AND SOLUTIONS

Although there are many fully functional GCSs available, our customers’ requirements largely requested concepts and features that are not available on any other GCS. In fact, most of our customers’ requirements were solutions to shortcomings of the GCSs they have used in the past. That being said, this project presented our team with several challenges for which there were limited, or no solutions found. The lack of documentation and other developers with similar issues or experiences required us to overcome complex obstacles entirely on our own when usually aid would be available on the internet in the form of documentation or tutorials. Our solution to this was to research to the best of our ability, continue moving forward with new ideas when others failed, and reach out frequently to our supervisor and alumni who worked on the UAV and GCS in the past. We often pieced together code with separate functions to perform the function necessary.

Additionally, our team members had no prior experience working with UAVs or GCSs which made development slow at first as we had to better understand the world of UAV piloting and the features that GCSs should offer at a basic level. We overcame the barrier by frequently reaching out to our customers for terminology and feature clarification and other students who have worked with UAVs or GCSs before.
6. RESULTS

Over the course of the project our team delivered a complete set of customer requirements, SRS consisting of documents listed in the deliverables section of this report, UI design documents, test cases and results, and a user manual, as well as supplementary documentation accumulated over time such as customer meeting notes. The implementation of the scope of requirements allocated to this senior design project were also delivered and joined with the StMU UAV Lab GitHub repository for customer and future project team access. We successfully implemented the underlaying architecture of the application meaning that the containers for all future developed widgets are manifested and can successfully swap (or not in the case of the pilot side of the app) as required. On button-push, the settings page appears containing tabs that allow the user to set and reset the layouts that are associated with each of the layout preset buttons on the homepage and upload UAV parameter files for use in the file system discussed in section 4.3.
Figure 4: Main Window

Figure 5: Emergency Settings Menu
7. ECONOMIC, PUBLIC HEALTH, SAFETY, WELFARE, AND ENVIRONMENTAL ANALYSIS OF RESULTS

Throughout the project’s duration, our team worked closely with St. Mary's University faculty and staff to effectively envision, plan, and build a product that will benefit the community here on campus and within the engineering department specifically. The underlying goal was to be a part of the expansion of the StMU UAV Laboratory and corresponding programs and provide a scalable project that future senior design teams can further expand. We plan to continuously adhere to feedback and concerns regarding this project to ensure we have a
working solution to any negative factors involving the development and utilization of our ground control station in the future. It was necessary to adhere to local UAV flight standards and regulations and integrate those into our requirements to ensure the project's compliance. We do not foresee this project impacting any global, cultural, environmental, or economic factors otherwise.

We believe this project could indirectly impact the safety of the public since we are providing users with UAV ground control software. We are aware that UAV flight is a regulated activity at the federal, state, and local levels and we took precautions to ensure our conformance to these laws and regulations. Under FAA regulations, St. Mary’s UAV Lab will be flying recreationally and so we collected, reviewed, and included the appropriate regulations in our documentation as well as warnings in our system documentation to ensure future users are aware of the regulations as well. In order to mitigate the extent to which the UAV Lab’s use of this GCS will impact public wellbeing, we researched and identified federal, state, and local laws as well as St. Mary's University campus policy on flying drones to make sure we were compliant with any restrictions.

8. FURTHER IMPLEMENTATION

Due to the magnitude of a fully functional ground control system, this project will be the first of at least two senior design projects undertaking its development. Because this project’s purpose was to design and implement the foundation of what will be a complete GCS, we were
required to design with several factors in mind to facilitate future development: scalability, portability, modularity, and maintainability.

Firstly, the UAV lab is St. Mary’s engineering department’s newest addition of infrastructure and will be accompanied by UAV-centric programs, so the use of the laboratory and by extension, the GCS, will expand in the future. Taking this into consideration, a requirement of the GCS is to be designed and developed with a pathway to be scalable and portable. Initially, the lab may have a handful of users and UAVs to manage, but eventually these numbers will grow, so the GCS needs to be able to handle this growth. To ensure this pathway, we designed the system to use user and UAV objects each with an ID so that additional users and UAVs can be registered with the system. The GCS will eventually feature secure login to identify the user present and what UAV they are piloting at the time. Although not in this project’s scope, each user will have corresponding permissions that restrict their read and write abilities accordingly to ensure further security.

The customers also expressed interest in eventually using the GCS across many users and even to a field laptop in the future. This required us to design and develop with a pathway to be portable which we satisfied by choosing development tools that are cross-compatible and with the use of many virtual machines to simulate the required workspaces during testing. This allows us to ensure the system functions appropriately on machines with the minimum requirements regarding monitor count, memory, etc.

Finally, due to the nature of a multi-generation senior design project, it was imperative to document as clearly as possible with the expectation that this project represents a blueprint for the next senior design project to develop the remainder of the GCS. To satisfy this, we used Atlassian Jira and Confluence to author and manage every document which has been made
public to the UAV lab personnel for future reference and use. Additionally, the source code has been pushed to the UAV lab’s GitHub account so that the working software is available for download and editing.
9. REFERENCES

https://www.hindawi.com/journals/ddns/2015/285746/


10. **APPENDICES**

Link to [GitHub](#)

Link to [GCS Jira](#)

Link to [GCS Confluence](#)

Link to [StMU UAV GCS Shared Documents Confluence](#)
11. SMC CAPSTONE REFLECTIONS

Tori Gardner (SE)

I used to think that I had to have it all figured out by the time I left high school. And four years ago, I thought my plan would never change. In my time at St. Mary’s University, I changed my major once and my intended career path three times. Each time, I was terrified. But is it not evidence that I have become the product of a diverse education and supportive community that I was inclined and encouraged to explore areas unheard of to me at the time? Now I proceed without the fear of failure. I thank my professors and friends for teaching me resilience and faith.

I deeply credit each professor I crossed paths with for my experience here. I especially credit my advisor, Dr. Ozgur Aktunc, who’s classes I most frequently had the pleasure of attending. He taught me to love the elements of software that I once thought tedious and uninteresting. He taught me that, like life, software development is a cycle, each phase of which is essential to the product and none of which should be overlooked or rushed. I did my best to not rush my time at university, and still, I look back and wonder where all the time went.

Even so, I get to take a lot of it with me as I move on from this part of my life. I created some irreplaceable relationships that helped me grow myself into who I am now, and no one can take that from me. Here I found my partner in crime who supported me through my senior design and cannot thank enough for the immeasurable selflessness and strength. I leave St. Mary’s with immutable memories, career-starting opportunities, and a life-fulfilling passion. I say goodbye only for a short while knowing that I will be back for my graduate degree.
Edgar Moreno (CE)

Attending St. Mary’s, I was fully prepared to advance my academics and grow into a skilled engineer. While my goal has not shifted, St. Mary’s has nurtured the way I have approached this goal. Initially, I believed that to be a skilled engineer I had to know all the material in my discipline and just enough material from other branches of engineering. I realize now that it is not simply enough to be a skilled engineer. What is needed is for the engineer to be capable. A capable engineer must be loyal to what the customer wants, while using their expertise to deliver a safe product. This likely means the engineer must propose new approaches that they do not necessarily agree with in order to adhere to both those conditions. This reality has come into practice during my senior year at St. Mary’s University with my contribution to the foundation of a functional Ground Control Station for the newly build Drone Lab.

Adolfo Mendiola (SE)

There are many events throughout life that have the potential to fundamentally change someone and cause that person to grow. I would consider my time at St. Mary’s one of those life-changing events. The decision to attend St. Mary’s meant leaving the safety and comfort of my home and family. I would have to leave my hometown, live alone for the first time, and learn to interact better with others. This activity expanded my worldview and made me realize that there is an infinite number of ways people communicate and coexist together that was previously unknown to me due to the limitation of my hometown. The software engineering discipline in general expanded my knowledge and understanding of how software works in the world. Before, what was available to me was limited in scope and mostly consisted of theory. This major helped me understand how software engineering topics are interconnected and how computers, like
people, are highly interconnected in ways that most people do not notice on the daily basis. This senior design project specifically has taught me the challenges of working remotely with people who have different living situations and has made me more compassionate to the conflicts and problems that everyone experiences. The amount of effort that large corporations expend to keep their workflow fast and stable must be enormous due to all the communication needed. That would be the biggest lesson I have learned— that constant, clear, communication is mandatory to any successful endeavor.