ARC Tutorials Documentation

Release 0.0.1

ARC

Dec 09, 2021

Contents

Aerospace Robotics Competition is a competitive program designed to teach high school students about aerospace engineering principles through designing, building, programming, and flying UAVs. The competition is structured as a combination of technical communication, student piloted flight, and autonomous flight through which students learn team leadership, hands-on vehicle design and and manufacturing, autonomous programming, and execution of design of experiments. With a multi-tiered mentor structure, students work with aerospace university students as well as industry professionals to be supported through the competition challenges and grow their network.

This website was built to collect all the information/knowledge/experience needed to help teams run/build their team and drones. The below topics are covered:

CHAPTER 1

Introduction

1.1 Drone 101

Sriraj Srihari

1.1.1 Introduction

This article will go over the basics of drones. However, it is wise to start with the bare bones basics of what a drone even is. A common name that professionals use for drones is unmanned aerial vehicle (UAV). Another name is a copter, with the prefix of copter meaning how many rotors it has. For example, a quadcopter has four rotors, while an octocopter has eight. Quadcopters are probably the most common style of copter since it offers good stability, while also being easy to work with. Drones tend to be lightweight, since they do not use tons of power like planes do. A small rechargeable battery can allow lightweight hobby drones to fly for at least 30 minutes, while certain drones have larger batteries that can allow them to fly for even longer.

1.1.2 Frame

The most important part of a drone is its frame. Without it, there will be nothing to hold up the different components that a drone needs to fly. They tend to be made from very lightweight materials such as carbon fiber, plastic, metals, or even 3D printed filaments. The frame can be made extremely lightweight because the plastics/composites tend to have a decent strength to weight ratio, meaning they can hold a lot of weight even though they are light. Since most drone components are lightweight, plastics can hold up most to all the components. Some camera drones or professional drones require the use of metal, as the components that go into those kinds of drones can be heavy, or just require a lot of strength to keep still. In this competition, the most common materials will most likely be plastics or 3D printed filaments. Small drones that only need to be in the air for a little amount of time tend to not require large batteries or complicated systems, so they tend to be made from plastics and are flimsy. Regardless, the strength of even plastic keeps it from shattering or breaking completely when dropped from a height. However, strength should always be checked before manufacturing a final frame, just so that there are no last-minute breaks that might impact the finished product of the drone. One big weak point in a frame can be along the rotor arm, as it extends away from the frame of the drone. When designing a frame, keep in mind that the rotor will sit at the end of the arm, weighing it down. The

spot in the middle of arm tends to be the weakest in this scenario, as too much weight at the rotor end of the arm may cause the arm to snap.

1.1.3 Battery

All electrical components of the drone will need power, and that is where a battery comes in. Batteries will allow each component to get the power it needs so that the drone can stay in flight for as long as needed, while also being able to perform its tasks. The most common type of battery used is a Lithium Polymer battery, also called a LiPo battery. LiPo batteries are used in drones as they are lightweight compared to other types of batteries. They offer strong current while providing the right amount of voltage as well. While LiPo batteries have lower energy density (energy over a certain volume), they are much lighter than other batteries, are safer to work with, and have an overall energy capacity greater than most other batteries. They also recharge quickly, which decreases time between flights.

1.1.4 Motors

Drone motors vary in style, but the most common type is brushless. Motors will give the drone lift, allowing it to fly. Varying speeds of motors can make the drone turn, lift, drop, or do tricks. When choosing a motor type, either look for brushless or brushed. Brushless motors will last longer due to the way they spin, but both motors will provide lift and motion. When choosing a motor, important things to look for are thrust to weight ratios and kv, which stands for constant velocity. Thrust-to-weight ratios are important since the motors will need to be able to lift the drone. The constant velocity (kv) rating tells you what the rpm of the motor is when one volt is applied to it. While a high kv value might be good in certain situations, note that higher kv motors tend to be larger, and may not always be suited for your exact purpose. Sometimes, it is better to go with a lower kv if your application requires it.

1.1.5 Propellers

Propellers attach to the motor, and the spin of the propellers is what moves the drone around. The size of a propeller blade matters, as smaller blades tend to offer more aerobatic flights, while larger blades offer stability. There are "middle ground" blades as well that offer both stability and maneuverability, but if your application needs high maneuverability, then smaller blades might be helpful. Blades tend to be made of plastic or carbon fiber, as they are very light. Most importantly, blades are specific in the sense that one side needs to be facing up and one side down. This is because blades are not flat, they have curves that help air move around the blades and placing the blades the right way will give you the best results.

1.1.6 Electronic Speed Controller (ESC)

The ESC provides the drone the ability to fly. It will give the flight controller the ability to control the direction and speed of the drone. An important thing to keep in mind when choosing an ESC is the voltage and the maximum current the ESC can handle. The motors will need a specific current to work correctly, and you must make sure that the ESC can take the total current that all the motors need, as well as the flight controller current as well. The ESC connects the flight controller to the motor, and sometimes the ESC will need to output its voltage and current to a power distribution board so that the split is even, and each motor will get its correct power.

1.1.7 Flight Controller/Autopilot

The flight controller talks to the motors of the drone and any other electrical components that might need external controlling. The flight controller is what receives data from a radio controller and uses the ESC to "talk" to the motors. Flight controllers come with certain firmware and software that can be used to control the system, but radio controllers can be connected as well. Certain flight controllers also have modules in them that take down data such as velocity or position. The flight controller can also take inputs from other controllers and sensors to get data back. For example,

you can connect a gyroscope to a controller so that the controller can relay the position and angle data back to the gyro, and the gyro can send that information back to a computer. The flight controller is the most powerful piece of the drone's electronics, as it acts as the "motherboard" to the rest of the drone. It can handle telemetry and control, and can even be programmed in junction with a gyroscope to create a control system, where deviances in drone flight can be corrected by the controller automatically, without needing to manually fix it.

CHAPTER 2

Fundamentals

NOTE: This information is only applicable for the 2021-2022 competition

2.1 Competition

This year's competition continues to involve three portions of the contest: technical presentation, semi-autonomous, and autonomous flight rounds. The competition rules can be found here: Rules

2.2 Kit

This is intended to give teams a walkthrough on the parts that will be included in the kit. The kit was designed to be affordable to teams (~$600) while capable enough for the teams to accomplish the missions outlined in the competition rulebook. There are plenty of spares included in the kit to help teams deal with failures that will occur. Instructions are included in the following pages (see sidebar).

The primary feature of this kit is the inclusion of two different autopilots. Both the APM 2.8 and the Pixhawk will allow teams to compete in the competition; however, the APM 2.8 is easier in that it performs the basic functions required where as the Pixhawk allows for advanced functionality if teams prefer to use that. The addition of another older yet capable system should allow teams to use the older system to learn how to fly the quadcopter without damaging more expensive equipment. **It is highly recommended that teams follow this practice.**

Table 1: **Kit Parts**

Category	Part name	Number of Parts
Drone	S500 frame	1
	Motors	6
	Electronic Speed Controller	4
	Propellers	6
	LiPo Batteries (3S 2000mAh)	2
Autopilots	APM 2.8	1
	Pixhawk Cool Kit	1
	Telemetry radio set (vehicle and GS radio)	1
Controller	Transmitter/Receiver combo	1
Support Equipment	Battery Charger	1
	LiPo Bags	1

The above table is a headlined summary of the kit. A more detailed kit description will be available soon.

Please let us know if certain components are missing from your kit. Please check that your team has all the items required to compete! Teams should receive their kits by the end of October/beginning of November once the staff has verified that the first payment has come through.

2.3 Financial Plan

ARC is an exciting and immersive experience where students will learn many facets of UAV operations. This year, teams can buy a kit. This will ease the hassle and allow teams to receive a fully flyable drone (assembly required) as soon as possible and focus on developing their competition plan and flying. The kit costs $600 and will be built into the financial plan described later. Teams however, are NOT required to purchase a kit, and can go out and purchase their own drone; HOWEVER, the drone MUST meet competition requirements to compete. Non-regulation drones will NOT be allowed to compete. The kit provided drone is designed/selected around competition requirements.

In order to fully experience ARC, teams need to have their quadcopter as early into the school year as possible. To allow this, teams who are selected to participate will immediately get their entire quadcopter kit (the ARC Kit) and be required to pay for it throughout the year. This will allow teams to fundraise throughout the year while still preparing for the competition with a fully functioning vehicle. Once a team is selected to participate they, along with their school leadership, are required to sign a financial agreement, below. They must submit this agreement with $100, non-refundable, at which time they will be sent their ARC Kit.

If a team is selected and does not send back a financial agreement within 5 calendar days they will be removed from participation. The following identifies the payment plan:

If a team decides they no longer want to participate they are simply asked to send back their ARC Kit and no more payments will be required. If a team has already made payments totaling over $700 they can keep their ARC Kit and will be refunded any money received over $700. If a team is late to make payments they must work with the ARC staff on an appropriate path forward. No team may be officially placed in a top 3 spot unless they have made payments in full.

INSTRUCTIONS FOR PAYING REGISTRATION/KIT FEES

Please visit the main competition page to find more information about paying.

CHAPTER 3

Aerospace Sciences

3.1 Dynamics

3.2 Propulsion

Coming soon! In the meantime, please feel free to ask us any specific questions or look at sources provided below:

- Ecalc for motor selection: https://www.ecalc.ch/xcoptercalc.php
- Advanced introduction to quadcopter dynamics: http://andrew.gibiansky.com/blog/physics/quadcopter-dynamics/

CHAPTER 4

Software

In this section, we provide links to resources that will allow teams to download and use the software needed to compete. The two primary software that teams will use is Arducopter for onboard software and ground station software.

4.1 Onboard

4.1.1 Arducopter

Arducopter is the autopilot software used to control the quadcopter. Arducopter is the helicopter variant of the Ardupilot suite of open-source autopilot software. Arducopter works for not just quadcopters but also for many different types of helicopters. More details can be found in the following links on how Arducopter works.

Arducopter Introduction: http://ardupilot.org/copter/docs/introduction.html

4.2 Off-board

4.2.1 Ground Station

There are two primary options for teams to use for ground station software: Mission Planner and APM Planner. Both fulfill the requirements for the competition.

Mission Planner: http://ardupilot.org/planner/

APM Planner: http://ardupilot.org/planner2/

MAVProxy is also another option for teams that want to be more creative with their ground station setup.

MAVProxy: http://ardupilot.github.io/MAVProxy/html/index.html

Step by step guide on how to use these ground station to have the quadcopter to do autonomous functions.

How to command your quadcopter: http://ardupilot.org/copter/docs/common-mission-planning.html

CHAPTER 5

Programming

5.1 Introduction

Programming is needed to accomplish the tasks to compete in the autonomous portion of the competition. We will provide a very *brief* overview of what programming is in a nutshell. **Please refer to the additional resources for actual coursework, this page only serves as a brief overview.**

In a nutshell, computer programming is developing a set of instructions and encoding them into the computer such that the computer can carry out the instructions. The set of instructions or algorithms dictate how the computer will carry them out, and the translation/encoding makes the instructions understandable to the computer. From the description given above,

Many people often fall into the trap of diving right into the actual coding of the computer rather than firmly developing the set of instructions or algorithms that it wants the computer program to execute. This is often times a trap as the programmer will most likely end up spending more time programming if they dove right into the coding compared to someone who spends some upfront time to fully define the instructions to which they want the computer or system to carry out.

This section is divided into two sections: Algorithms and Software Engineering. The algorithms section is intended to help teams figure out how to properly fully define the instruction set they want the computer to carry out. The Software Engineering portion will dive into the the coding of the computer.

It is impossible to teach everything there is about computer programming in a single web-page, so this page summarizes the important points that teams should understand and utilize to compete in the competition. Additional resources are given in the Resources section.

Finally, a walkthrough of the example code is given in the 2021-2022 Competition Specific Material section. The hope is that teams can use that example code as a starting point for developing their own path-planning code as outlined in the competition rules.

5.2 Algorithms

Algorithms are defined as a set of instructions that allows the computer to accomplish its goal. These sets can be vary from very complex algorithms such as machine learning to AI or can be as simple as a calculation of a formula.

A set of instructions can be broken down into 3 different components. All instructions are formed by a combination of these three. The different types of instructions:

- Command: This is pretty self explanatory as it forms is what really executes the instructions. This is one of the simplest form of an algorithm as it essentially tells the computer specifically what action to take.
- Conditional: A conditional allows commands to occur only if certain conditions are fulfilled. Most often called if/else statements, a computer will run a command defined in the if portion of the code if the condition is satisfied; otherwise, it will run the else statements. There also exists if/elseif/else where you can define multiple conditions to check for when the computer is deciding what to do. If there are many different conditions you want to check, a "switch" statement may be more useful.
- Loops: This construct allows command(s) to be carried out multiple times in a row. The number of times is dependent on how the loop is set up. There are two types of loops: for loops and while loops. In for loops, you specify how many times the command should be repeated. While loops allow you to combine both a conditional and a loop: the command(s) will continue to be executed until a certain condition is met or a condition is not being met.

These three combined can form an algorithm. Let's look at an example: a recipe. In a recipe, you have a set of instructions that you have to follow. For example, if the recipe calls for two eggs to be added into a bowl, that would be an example of a command. A conditional could be that if the dough is not clumping as much it should be, add water. The condition here is checking the clumping-ness of a dough and the action if the condition is not satisfied as adding the water. An example of a loop is putting something into the oven until it rises. Although you may not explicitly be continually adding stuff into the oven (keeping say the batch in the oven could itself be a command), the while loop conditional can be the status of how much the batch has "risen".

When developing an algorithm, the primary strategy is to see if you can apply these 3 components in certain ways that allow you to accomplish the task that you want carried out by the computer program. Thinking through how these components can be combined and taking advantage of the pros and cons will allow great algorithms to be developed.

5.3 Software Engineering

Software engineering is at its core the **translation of instructions to something a computer can understand**. Once the set of instructions or algorithms is defined above, the usage of computer languages is relatively straight forward.

We will not go over the specifics of how to program here; the additional resources provided will do a much better job than what we can do here. However, some key points and tools will be provided here so that teams can get started on programming as well as some tips.

5.3.1 Programming Languages

Similar to how there are many languages that we humans can use to talk to one another, there are also many different computer languages that allow humans to program computers. Below are some examples. For this competition, we highly recommend the use of Python due to its extensive support, usefulness, and simplicity. Additional resources to learn Python are given in the additional resources section.

- Python: https://www.python.org/
- C++: http://www.cplusplus.com/
- Java: https://www.java.com/en/

- Perl: https://www.perl.org/
- Fortran: https://www.fortran.com/
- MATLAB / Octave: https://www.gnu.org/software/octave/

From this point on, we will assume that the team will be programming in Python.

5.3.2 Compilers/IDEs/Editors

Code is a human-readable version of a set of instructions that you want to give to a computer. However, the computer cannot read code; it needs to be converted or "compiled" to a format that computers can. There are two ways we can do it: interpreters and compilers. Interpreters are computer programs designed to read line by line your computer code and send the computer the translation line by line. Compilers on the other hand convert your entire script into something machine-readable. The execution of that program is done later and in another process. Python is an example programming language that is interpreted, and C++ is an example of a programming language that is compiled.

Many computer applications called Integrated Development Environemnt (IDE) /Editors exist to help with coding/compiling/running process. Below are some great examples that we recommend teams use to help with their coding:

- VS Code
- Sublime Text
- Atom
- PyCharm

5.3.3 Open Source/Libraries

One of the great ideas that the computer science community supports is the idea of open source software/code. The sharing of code that anyone has developed has created a positive reinforcement cycle that allows more advanced applications to be created without everyone having to start from scratch. Computer languages have adopted the idea through the creating of software libraries. Libraries are code that some third party has made that can be used for your own purposes. One of the biggest strengths of Python is the extensive development of software libraries that allow programmers to use Python to do things like manage radios to machine learning. In fact, the autopilot software was created through open source code! We've listed a couple libraries that may be of interest to teams:

- matplotlib
- scipy
- numpy
- dronekit
- arducopter

5.3.4 Version Control

When you are writing code, saving often is a good way to make sure that your progress isn't lost. However, what if you decided to try something, but it didn't work out the way you had hoped to? Version control will allow you to revert back to not just what you saved previously, but every version that you saved! That way, you can go back to any version of code that you had previously. Git (the computer version), Github , and Gitlab (both web based versions) are primary examples of easy to use version control software. For help in setting version control up as well as a better introduction, https://try.github.io/ should help.

5.3.5 Debugging Tips

Coming soon!

5.3.6 Best Practices

1. Use version control: This will allow teams to manage their code and allow different people to work on different parts at the same time.
2. Test often and early. Don't wait til you're "done" to test out your code! There may be some bug in the beginning that you will need to take care of that may affect the rest of the code! Find that bug early before moving on!
3. Plan out your code structure beforehand. This will help save time and reduce errors as you move along coding.
4. Use comments! Document your code so that other people who read it can understand what you are trying to do.

5.4 Resources

Below are some resources that should be helpful for teams to fully learn how to program

5.4.1 Intro

- Codeacademy
- Scratch
- Learn Python

5.4.2 Intermediate

- Dive into Python
- stack overflow

5.4.3 Advanced

- Introduction to Algorithms by Cormen
- MIT OCW

5.5 2021-2022 Competition Specific Material

Example code for the competition can be found here: https://github.com/aeroroboticscomp/ARC-example.

THe primary task for teams is to create a computer program that will allow teams to determine the optimal path to traverse the waypoints. In past years, we have asked teams to come up with a program that will compute the path, transfer the path to the drone, and then have the drone execute the mission. However, this year, we are only asking teams to compute the path. Therefore, teams should focus on designing a computer program to do so.

In the github link at the top, the 2021-2022 folder contains example codes as well as example mini-lessons on coding to help teams get started and understand how code can be written. All of these are made in Python; however, teams

are NOT required to use Python. Any non-GUI based programming language would suffice (ex. C, C++, Java, Perl, FORTRAN, etc.) .

NOTE: For Python users, Jupyter notebooks and equivalents are allowed. Please email ARC Staff if you have any questions.

To help with getting started on the tasks and learning, please see the Mini-Lessons page that has been created.

CHAPTER 6

DroneBlocks Basics

6.1 Overview

DroneBlocks is an online tool that provides a complete drone related education, from curriculum to physical drones. Currently, their product line contains drone curriculum, drones for sale, software, support, and professional development. Their URL is: https://www.droneblocks.io/.

The courses offered by DroneBlocks cover most drones, however it is primarily geared towards the Tello drone. Course offerings include programming with Python, using the DroneBlocks Simulator, and programming challenges.

DroneBlocks Products:

- DroneBlocks App: Block based programming app for drones, very intuitive for beginners to learn the logic behind programming a drone to fly somewhere, available as an app and a chrome toolbar.
- DroneBlocks Simulator: Essentially a visual output for the App, where block code can be used to simulate drone flight, either through challenge rings or just on a free plane.
- Curriculum: Drone lessons geared towards the Tello Drone. Lessons go from beginners guides to drone programming with various languages to challenges.

Ideally, it would make sense to start with the App/Simulator to get familiar with programming as a beginner, then head to the courses and work with the Tello Drone itself.

6.2 Login Process

To use the DroneBlocks curriculum, go to the login page and enter your login information: https://sso.teachable.com/secure/94245/identity/login

To get a login, check with your teachers or send an email to ARC at aero.robotics.comp@gmail.com.

After logging in, you will be greeted by the home screen:

6.3 Course Overview

This section will be an overview of each of the courses offered, giving a small brief on what each course covers.

Troubleshooting Tello: This course goes over various faults that may occur during the usage of a Tello drone and how to fix them. Topics are covered such as safely using a drone in a classroom/enclosed space, and part such as batteries or propellers.

Introduction to Tello Drone Programming: Ideally the first lesson to be taken. Introduces Tello and the DroneBlocks software, and covers basic block programming skills such as loops, variable definitions, and logic statements.

Tello Block Coding – Math Edition: Programming challenges using the block programming style, highlighting math related challenges particularly.

The DroneBlocks Simulator: Covers the DroneBlocks Simulator, with challenges to test knowledge. This would be the simulator version of the "Introduction to Tello Drone Programming" course.

Introduction to Tello EDU Drone Programming with DroneBlocks: Like the Intro to Tello Drone Programming course, this course goes over programming the EDU drone using DroneBlocks and doing various tests and challenges.

Advanced Tello Programming with DroneBlocks: Advanced coding techniques like nested loops, sine waves, and 3d visualization.

Advanced DroneBlocks with Functions: An unfinished lesson containing the creation of functions and management of functions, including recursive and polygonal functions.

Tello & Art Presents: Dance: Programming challenges with relation to dance and music.

Tello Challenges from Italy with Mr. Torelli - Part 1: Various programming challenges made by Mr. Torelli, with inspiration from Italy.

Tello Drone Programming with Python - Video Course: Introduction to programming a Tello Drone using Python.

OpenCV, Python, and DroneBlocks for Tello Camera Control: A course overviewing how to control the camera on a Tello Drone using tools such as OpenCV, Python, and DroneBlocks.

Advanced Tello Programming with Python 3 and OpenCV - Course 1/3: Advanced programming techniques such as using the Jupyter Notebook, Tello Video, and Script Runners.

Advanced Tello Programming with Python 3, OpenCV, and ArUco Markers - Course 2/3: Part 2 of the advanced programming course with techniques like point and click flying and ArUco Marker detection .

Advanced Tello Programming: When Art Meets Technology - Course 3/3: Part 3 of the advanced programming course which goes over image detection and processing.

Introduction to JavaScript Programming with DroneBlocks Code: Programming the Tello Drone with JavaScript. Topics such as variables, loops, and functions are covered.

Node-RED Programming with Tello and Tello EDU: Introduces a visual programming interface known as Node-RED. Topics such as mission box flows, dashboard navigation, and swarms are introduced.

Healthcare in the Himalayas Challenge: Provides a drone challenge with respect to providing healthcare in the Himalayas.

DroneBlocks Membership: Handily organizes all lessons.

DJITelloPy Drone Coding: Programming the Tello Drone using the DJITelloPy API.

6.4 Recommendations

To effectively use DroneBlocks, our recommendation is to start with the Introduction to Tello Drone Programming, and Introduction to Tello EDU Drone Programming with DroneBlocks lessons. This would be ideal for beginners to dip their feet into the DroneBlocks system, learning the basics of block style programming. It will get you started and give you enough knowledge to perform some tests on your own. Then, go to The DroneBlocks Simulator lesson. With the simulation lesson, you can learn how to use the simulator, which would be good if you do not always have access to the drone. It builds upon the block style programming, just the output is a simulation instead of a real-life drone. Once you are comfortable with the basics, move to the advanced lessons. Start with the Advanced Tello Programming with DroneBlocks lesson, then move to some of the challenges. While not all the challenges directly use the block programming, they may still be able to be completed using the blocks. After you believe you are strong with the blocks, move to some of the programming languages, such as Tello Drone Programming with Python - Video Course. Like block programming, start with the introduction, practice on the Tello drone, then move to the advanced course and challenges. The simulator does not use Python, so the programming languages will have to be practiced on a physical drone. If there are issues, check the Troubleshooting lesson.

While not all the lessons have to be watched, going through the lessons will surely improve your knowledge in drone programming, and maybe help you understand more of the dynamics behind how the code affects the drone. The best way to go through things is to watch a lesson video, then try practicing it on the Drone/Simulator. Taking notes can also help retain information.

CHAPTER 7

Mini-Lessons/Milestones

7.1 Mini-Lessons

7.1.1 ARC Mini-Lessons

The ARC Mini-Lessons are meant to provide teams with a constructive source of learning that they can use to gauge the skills of the team members and allow then to learn programming and computer science. The mini-Lessons can be found here: https://github.com/aeroroboticscomp/ARC-example/tree/master/Mini-Lessons

The lessons are currently split into these categories: - Algorithms: Focus more on numerical computation and algorithmic thinking. Lessons will give students the oportunities to implement algorithms and computational ideas, giving them a step towards what's asked of them in the - Computer Programming: Focus more on the programming side of things suh as syntax, tools, and libraries that are available to teams and that teams are allowed to use for competition

In the future, additional categories may be added depending on competition tasks.

7.1.2 Format

In each mini-lesson folder, the below will be contained: * Project specification files (PDF and README.md) * Starter Files * Jupyter Notebook (notebook copy + online link) * Solution Folder

In each solution folder, there will be the * PDF containing a walk through of the solution * Solution Jupyter Notebook and online Jupyter notebook link * Stand-alone code package that can be run through python

A Jupyter Notebook (https://jupyter.org/) is a hybrid code environment + word document that allows programmers to instantly see the results of their code without the need to install on their machines. An online Google-based link will also be provided that will allow students to use work on this at home or in school without a computer that has Python installed.

7.1.3 Prerequisites

There are some basic prerequisites that are assumed for team members attempting this competition. They are listed below: * Basic knoweldge of math (addition, subtraction, fractions) * Computer access/ability to install python and its associated libraries (Python 3 will be used for all code)

7.1.4 Best way to tackle these lessons

Below are some tips that will help in learning the material. These mini-lessons are meant to be tough and potentially challenging; however, they are rewarding for those that finish and will help teams think through the actual competition tasks asked of them.

The best way to go about these lessons is the following order: 1. Read through the lesson description and material thoroughly 2. Coming up with a plan/algorithm/methodology that will help implement 3. Implement and test the program 4. Repeat Step 2 and 3 as much as necessary 5. Ask questions/try different ways to solve the problem

We highly recommend teams check the solution ONLY after you've gon through Step 1 thru 4 **more then three times**. This is recommended to build the "debugging instinct", where students attempt to try to figure out why something isn't working without the solution by logically thinking about potential reasons why the code isn't working.

Teams should first work through the Computer Programming lessons before moving to the Algorithm lessons. The Algorithm lessons will build on the material found in the Computer Programming lessons.

7.1.5 List of Lessons

Below is the list of lessons that will be included. A brief summary of what each mini-lesson will be having students do is included in the description.

- Computer Programming - CP_1: File I/O
  - Description: Create a script that will parse a data file with a complex format and then create another data file.
  - CP_2: If/Else/Switch - Description: Create a script that will go through a complex logic system
  - CP_3: Class Construction - Description: Create a class that allows the containement of information contained in a file and process the classes.
  - CP_4: Usage of Libraries - Description: Create a script that computes a graph and plots it
  - CP_5: Object Oriented Programming - Description: A culmination of all the previous examples
- Algorithms - Algo_1: Fibonacci Sequence
  - Description: Create script that calculates the terms of the Fibonnaic Sequence and compute some statistics
  - Algo_2: Euler's Method - Description: Create script that propagates a differential equation and check how close it is the actual solution
  - Algo_3: Simple Optimization - Description: Design a simple optimization algorithm that allows you to find the optimal solution
  - Algo_4: Sorting Algorithms - Description: Implement 3 different sorting algorithms and find out which is the "best"
  - Algo_5: Root Finding - Description: Create a script that solves non-linear equations that can't be solved by hand
  - Algo_6: Breadth Search - Description: Create a script that goes through each different permutation and find sthe optimal "combination"

CHAPTER 8

Kit Hardware

Note: The 2021-2022 kit hardware is still being finalized. Below, you can find the overview of the 2019-2020 kit, which should be similar to this year's kit, and can be used as a starting point.

This is a brief run-down on the components that will be found when you open the kit box.

8.1 Autopilot/Radios/Electronics

Inside the kit, there are two sets of autopilots, a transmitter/receiver combo, two GPS units, and 900MHz telemetry radio system. The two autopilots are the Pixhawk and the APM 2.8. Because of the more delicate nature of the Pixhawk as well as its GPS unit, both of them and the 900MHz telemetry radios have been placed into the transmitter/receiver box to ensure that they are not damaged upon shipping.

The two autopilots are provided so that teams can experiment with the different capabilities of the autopilots without worrying about losting one of them. The Pixhawk is the more advanced variant of the APM 2.8, so please keep that in mind when assembling.

There are two sets of power distribution lines (the thick red/black wires connected to a circuit board in the middle). That will be used to power the Pixhawk/APM 2.8 during flight as well as pass the battery power to the board. Other electronic components included are a safety buzzer and safety switch for the Pixhawk.

8.2 Frame

The frame components can be see in the following pictures. This is the S500 frame with a built-in power distribution board. Teams will need to solder the ESC connections into the board itself. More details on frame assembly can be found here: https://www.youtube.com/watch?v=ibXNw8jLwBs

8.3 Motors

The kit provides 4 motors as well as 2 additional spare motors as shown in this picture.

8.4 Propellers/Accessories

The kit provides 4 propellers to put onto the motors and quadcopter. Furthermore, the kit provides some accesories such as zip-ties and battery straps to help keep items secure.

A single servo is also provided for the team's mechanisms. Teams are allowed to buy more servos for their mechanism as long as the quadcopter meets competition requirements.

8.5 ESC

The kit provides 4 ESCs.

8.6 Batteries

The kit provides with a team with two 2200mAh 3C batteries as well as a battery charger. The batteries should provide approximately 10 minutes of flight. Teams are allowed to buy their own larger batteries if they prefer.

A lipo bag is also provided so that teams can store their batteries safely. Always remember to place the batteries back into the provided bags so that if something goes wrong, the damage can be contained.

CHAPTER 9

Autonomy

9.1 Introduction to Autonomy

As many of you probably know, autonomy is a very important and popular topic. Systems today are becoming more complex and as we continue asking systems due to more, engineers become more inclined to make systems

An autonomous system usually consists of 4 components that function: perception, awareness, decision-making, and action. Each is briefly described:

- Perception: This component allows the system to "see" what is going on in its surroundings. This may be through what we consider our 5 senses (see, touch, taste, smell, hear) as well as another other artificial means (radar, infrared, etc.).
- Awareness: Once the system can percept what your surroundings are, the system must be able to understand what is going on. Being able to "see" something and understand what is going on are most often two different tasks. For example, an autonomous car may see a pedestrian on the street; however, all the sensor has done is detect that there is a pedestrian on the street which is useful information in itself, but not as useful as understanding where the pedestrian is headed or trying to understand what it is trying to do.
- Decision making: This is the component that is generating the most buzz today and what people consider "artificial intelligence". How can we teach a robot to make decisions on what to do and not to do? For example, let us go back to the example of the autonomous car. When the autonomous car goes sees a pedestrian, what should it do? Depending on what the pedestrian is doing the car may decide to continue on or attempt a maneuver to avoid the pedestrian. Although this may seem like a simple problem to solve with different if/else logic for each scenario, the seemingly infinite possibilities of scenarios makes this problem impossible to solve with simple if/else statements.
- Action: This component is what gets the most oohs/ahhs/Youtube video clicks in society. The action part of is the mechanism that allows systems to act on their decision. It could be as simple as running a computer program or as complex as flying from point A, delivery a package at point B, and landing at point C, all while avoiding traffic. This component is usually the physical hardware component of most systems.

The integration of all 4 components is one of the hardest challenge. Currently, all 4 components have various levels of maturity in the aerospace industry. Examples of interesting projects are the following:

- Skyborg

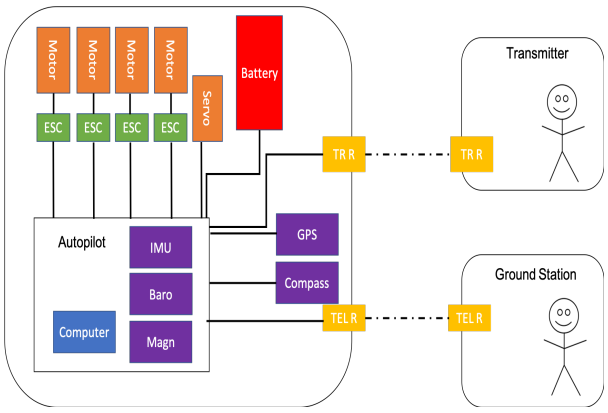
- AALIAS
- AACUS

Let us look at two example systems to try to determine what parts of the system are which: the quadcopter that you will be using and a human being. The idea of having 4 different components of autonomy will be reinforced through the examination of these 2 systems.

9.1.1 Quadcopter

Let's begin by looking at each of the systems onboard the quadcopter. We can classify all the systems onboard through the above introduced framework.

- Perception: IMU, GPS, Barometer, Magnetometer, Compass
- Awareness: Flight Computer, flight software
- Decision-Making: Flight computer, flight software, team's program
- Action: Motors, servos, mechanism



9.1.2 Human Being

Like many things that humans have designed and built, they are often modeled after the creator: human beings. You can distinctly define which parts of humans are which using the framework described above.

- Perception: Skin, eyes, ears, nose, mouth, tongue
- Awareness: Brain
- Decision-Making: Brain
- Action: All muscles, arms, legs, fingers, toes, etc.

9.2 Autonomy in ARC

In the Aerospace Robotics Competition, the autonomy portion will always involve all the components. For example, for the 2019-2020 competition, all 4 components are involved. The perception is taken care by the GPS, the awareness is handled by the the autopilot, the decision making is handled by the team's computer program, and the autopilot and the action is carried out by the team design mechanism.

As a note for future competition, ARC will continue to rotate between allowing teams to implement their own perception, awareness, decision-making, and/or action components. The 2019-2020 competition allowed teams to come up with the decision-making and action with the computer program and the drop mechanism design.

9.3 2019-2020 Competition Specific Material: Planning

In this year's competition, teams are asked create a computer program that allows the drone to traverse of a set of waypoints that are given to teams beforehand. **If you have not read the Programming section of the tutorial wiki, please read that before reading further.**

There are many ways to introduce this topic; however, in the interest of making this understandable to high schoolers like you, we will condense it into format that will help teams in accomplishing the task.

The general problem that you are trying to solve is called the "Traveling Salesman" problem . This problem is a classic problem in the computer science and optimization world, and in essence comes down to "finding the path that takes the least amount of time for a person to travel to all the places it needs to go".

One approach to go about accomplishing this is to iterate through all the different possible combinations of routes (keep in mind to make sure you start from the home base and return to the home base). In reality, this is not a realistic way as often times, the number of places you will need to visit is many times a very very large number and may take forever to go through every single combination. However, for this competition, we have designed it such that it is possible to do so.

Another way to go about accomplishing this task is to come up with a heuristic (or general idea/strategy) that should lead to a close to optimal route. This approach is often times used when the problem is so complex that we have a general idea that we know will give us the answer, but we have no guarantees that the answer will be the best one. For example, say you want to go from point A to point B. Point A is on the north and east side of Point B. Rather than going through Google Maps/Waze/Yahoo/your-choice-of-mapping-software to find the shortest route there, a strategy you could take is to take any road that takes you south and west, and hopefully get there (or close to there). This approach has surprisingly given many breakthroughs in computer science, as often times, you don't need to find the absolute optimal route to reach your destination/goal. Example heuristic ideas/strategies could involve going to the waypoint closest, or going to the waypoint that takes the least amount of time. The strategy can be more complex as well such as going minimizing the criss-crossing of routes or simply just taking the simplest route.

We invite you to be creative and think hard on what are the best algorithms for this task!

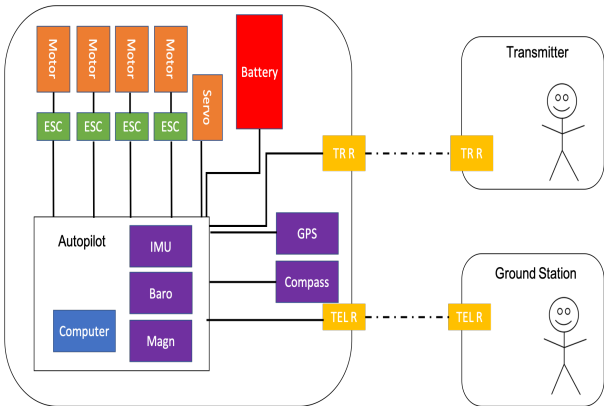
CHAPTER 10

Integration/Assembly

In this section, we will utilize a lot of material that has already been created for this specific purpose. Many of links are placed here. Please go through every webpage linked as most if not all links provide step-by-step instructions on how to integrate all the systems together.

10.1 Systems Overview

The overall system diagram is shown in the below picture:



All these individual components were introduced in the Drone 101 page. This diagram shows how the individual components are connected to each other.

There are two operators: the pilot and the ground station operator. Contrary to popular belief, often times, automation requires **more people** to operate than non-automated systems (why? a good question to think about). The two operators interact with the quadcopter system differently: the pilot has direct control to the tactical/short-time-span actions while the ground station operator has a more strategic view of the system operation since it is ensuring that the quadcopter is executing its mission.

The quadcopter operates by listening to the central autopilot which in this case is the APM or the Pixhawk. The autopilot sends signals to the ESC which translates it into a signal that motors can interpret. Depending on the time,

a signal may be sent to the servo to actuate a mechanism. The battery powers all of these actions and is regulated through the power distribution board.

As a reminder, teams are allowed to swap out parts for different systems not provided in the kit. Key questions to think about when considering that:

- What are the advantages/disadvantages of swapping parts? Every part selection choice carries with it some pros and cons. Actually thinking through them is a useful exercise
- What additional systems can be attached on to allow for improved performance? Keep in mind, additional systems can be added during testing to improve the testing cadence
- How does the system choices influence operations? For example, when I'm flying, is there a way to know how much battery I have left without constantly looking at the ground station screen? Understanding system design choices and how it effects operations will go a long way in creating the optimal system for competition.

10.2 Systems Integration/Assembly

Many of questions/instructions on how to integrate and assemble all the components have already been written out in the Arducopter documentation (see Software page for more details about Arducopter). If you are using the kit, all components of the kit are compatible with Arducopter. Please follow the below instructions to integrate all your components. Feel free to reach out to us (or even the people who wrote the tutorials!) if you have any specific questions.

Steps to first time flight with step-by-step instructions: http://ardupilot.org/copter/docs/initial-setup.html

Pixhawk details: http://ardupilot.org/copter/docs/common-pixhawk-overview.html

Pixhawk Wiring Instructions: http://ardupilot.org/copter/docs/common-pixhawk-wiring-and-quick-start.html

Receiver Connections: http://ardupilot.org/copter/docs/common-pixhawk-and-px4-compatible-rc-transmitter-and-receiver-systems.html

Connecting ESCs and Motors: http://ardupilot.org/copter/docs/connect-escs-and-motors.html

Flying preparation: http://ardupilot.org/copter/docs/flying-arducopter.html

CHAPTER 11

Team Dynamiccs

11.1 Team Building

11.1.1 Team-Building 101

What is team-building?

Team-building is the process of strengthening relations within a team resulting in increased team effectiveness, motivation, and cooperation.

Why is team-building important?

- Improves inner-team communication: team-building creates discussion which encourages open communication within the team
- Develops problem-solving skills: activities that challenge teams to work together strengthens the team's ability to tackle a task as a group, building those skills for the "real problem"
- Increases comfort zone: team-building encourages team members to be comfortable in a group setting and increases trust in other team members
- Encourages creativity: activities that are outside of students ordinary setting encourages students to think outside of the box, leading to new ideas and creativity
- Defines team as "fun": team-building encourages team members to have fun as a team, making the team environment more communal moving forward

When should we do team-building activities?

Team-building frequency can very team to team. The following are recommendations for when team-building may be especially beneficial:

- When the team is initially formed (start of school year)

- When a new member joins the team
- When the team has something to celebrate
- When the team has something specific to work on, i.e. communication or creative thinking

What are examples of team-building activities?

Team-building activities can vary from simple fun to challenges to strengthen specific aspects of the team. Below are some examples of different team-building activities!

- Scavenger hunt: Divide team into small groups and give each group a list of items to find or tasks to complete. Groups compete against each other to be the first group done.
- Two truths and a lie: Each student tells 3 "facts" about themselves to the group, two of which are accurate, one of which is a lie. Team members try to guess which "fact" is the lie.
- Ice cream social or team lunch: Fun, low-key way for team-members to socialize
- Blindfold Adventure: All but one team members are blindfolded. The non-blindfolded team member verbally instructs the other team members to complete a task. As an example, lead member (non-blindfolded) leads team members to a playground playscape. Lead member verbally instructs team members one by one to climb stairs and slide down the slide.

11.2 Financial Planning

11.2.1 Why Financial Plan?

Financial planning is an essential first step when preparing for a major project, such as competing in ARC. A financial plan and budget should be established at the conception of a project team and monitored throughout the project completion to ensure the finances remain on track. Financial planning ensure teams are cognizant of and prepared for the costs associated with the project. The categories listed below are important steps of financial planning.

11.2.2 Create a Budget

The first step of financial planning is creating a budget for the project. To create a budget, it is important that the team understands all of what will be required to complete the project. This may require the team spending time discussing what supplies will be needed and the cost of such supplies. Other costs such as travel, team building (if sponsored by team), and other supplies the team may choose, like t-shirts, should be accounted for in the budget. A sample budget for a team building an RC aircraft is below.

COMPETITION COSTS		ELECTRONICS	
Transportation	$ 3,000.00	Circuit Boards	$ 800.00
Hotel Accomodations	$ 3,000.00	Servos	$ 250.00
T-Shirts	$ 500.00	Sensors and Telmetry	$ 500.00
Subtotal	$ 6,500.00	Batteries	$ 400.00
STRUCTURE MATERIALS		Wiring	$ 200.00
Wood (balsa, bass, plywood)	$ 2,500.00	Subtotal	$2,150.00
Composites (carbon fiber, epoxy, etc)	$ 3,000.00	ENGINE AND PROPULSION	
Metals (aluminum)	$ 600.00	Engines	$1,000.00
Landing Gear	$ 100.00	Propellers	$ 250.00
Tool Replacements	$ 750.00	Propulsion Electronics	$ 350.00
Miscellaneous (screws, bolts, glue, etc)	$ 750.00	Subtotal	$1,600.00
Subtotal	$ 7,700.00	Contingency	$1,795.00
Total Cost		**$**	**19,745.00**

Note the following on the sample budget:

- Budget is broken up by category with a subtotal for each category. This allows the team to easily identify what aspects of the project cost the most.
- This team budgets for a contingency fund. This is important to ensure teams have money to support any unexpected expenses or expenses that are over budget. A suggested contingency amount is 10% of you other budget items.

11.2.3 Obtaining Financial Resources

1) Brainstorm Sources of Financial Support

Once the team has determined the cost of competing, the team will need to consider how they will fund the cost. While some teams may have funding from their school, some may require funding from external sources. Teams in this position should work together to brainstorm sources of funding. Consider local businesses such as car dealers or restaurants that like to support the local schools. Consider also any companies that are involved in UAVs or robotics, like local RC hobby shops.

2) Create a Sponsorship Packet

Before contacting local businesses for sponsorship, it is highly recommended that you create a Sponsorship Packet that you can share with potential sponsors. A Sponsorship Packet is a professional document that can be given to potential sponsors that captures the goal of your team and the financial requirements to reach that goal. Here is an outline of what should be included in your packet:

1. Cover Page: School name, team name, logo, etc
2. Team Introduction: Team picture, Team details (grades of students on team), Names of key members
3. Competition Details: Brief description of the competition, Team goal
4. Last Year's Competition (if applicable): Team's results from last year
5. Sponsorship Details: Benefits to sponsoring the team

- Examples:
- Company logo on t-shirt
- Company logo on competition presentation

6. Budget: Screenshot of budget
7. Thank You (if applicable):

- Thank you slide from sponsors of previous year's competition

Note - include your school as a sponsor if they supported your team!

3) Meet with Sponsors

Once a Sponsorship Packet has been created, set up meetings with the list of potential sponsors that the team brainstormed. Take each potential sponsor a printed copy of the packet so they can review after your meeting. Thank all individuals for taking the time to meet with you and don't be disappointed if the potential sponsor is not able to commit to a financial contribution at that time.

11.2.4 Sponsor Relations

During the school year, maintain a consistent and professional relationship with your sponsors. Communicate often about the status of the team and the vehicle as you prepare for the competition. Consider writing a newsletter to send to sponsors with these updates. It is recommended that your team designates one person to communicate with the sponsors, so that sponsors are not receiving emails from a variety of people. If your sponsor is local, consider inviting them to attend the competition at the end of the school year. Maintaining a positive, working relationship with your sponsors will make them more likely to sponsor your team in the future.

CHAPTER 12

Safety

12.1 General safety

1. Put on propellers LAST: There is no reason whatsoever to put on propellers until right before you start flying. They should not be on at all during the build or testing and should only put on when you decide you want to fly. Wrong wiring or accidental powering can cause the motors to spin up suddenly which may cause injuries. We do not want any injuries as these motors are very powerful and can easily cause non-fatal and in some special circumstances fatal injuries.
2. If you are unsure what whether you are doing the right thing, DOUBLE CHECK OR ASK FOR HELP: Lithium polymer batteries can be very unstable and although most of the electronics have some kind of protection circuit, you can still fry electronics with wrong wiring. Please make sure you double check with this manual or feel free to email Chris or ARC.

12.1.1 LiPo Safety

Storage:

- Battery should be stored in stable environmental conditions.
- Battery should be stored in LiPo battery bag or surplus ammo case.

-Make sure the LiPo battery charge is at 3.8 V per cell. You can do this through the LiPo charger discharging setting.

Charging:

- Battery should not be left unattended for extended periods of time.
- Battery should be charged only with a balance charger.
- Follow the instructions to charge given in the charger manual. Make sure the correct cell number and current are set before charging.

- Full charge is 4.2 V per cell. Make sure all cells are balanced/same voltage (see charger manual on how to check).

Damage:

If any damage is visible to the battery, the battery should not be used! Using the battery increases the risk of fire/explosion. Possible damage includes:

- Dents, punctures
- Puffiness of battery (this is hydrogen gas causing the expansion which is flammable)

If any LiPos are damaged, please contact ARC for specific instructions on disposal and handling. Do not use the LiPos if you feel it is damaged.

Fire Procedure:

These batteries do sometimes ignite when damaged and can cause a fire. Do not use standard fire extinguishing procedures; rather use the following set of instructions. Attempt to contain fire in metal container or a non flammable surface. Douse with water (constantly) Use fire extinguishers to help contain Battery will eventually burn itself out when chemical reaction is complete

CHAPTER 13

Flight Testing/ Operations

Flight testing the quadcopter is the most important part of preparation for competition. Teams that fly often and practice often are the ones that do best. This page will walk through some basic testing as well as tips for getting the most out of testing.

13.1 Ground testing

Before teams fly for the first time, ground testing can be done to make sure that all systems are working properly before attempting flight. Before attempting to fly, below are some tests that could be done to make sure the systems are operating properly:

WARNING: For all ground tests, **the propellers should not be attached to the motors. Only place propellers on motors when you are ready to fly**

- Motor spin test: Attach pieces of tape onto the motors to determine whether the motors are spinning in the right direction
- Motor throttle up: Ensure motors are increasing in power when you throttle up
- Familiarization with arming/disarming procedure: Ensuring you know when to arm and disarm the autopilots

13.2 Testing setup

There are two ways to do flight testing: flying indoors and flying outdoors. For both, tethering is highly recommended and will be required when flying at competition. There are many ways to securely tether the quadcopter. The two recommended ways are as follows:

- Two person tether: Tie two 20 ft long tethers on both sides of the quadcopter and have two people hold the quadcopter as the quadcopter is flying. This is more dangerous then the cinder-block tether; however, ideal for small scale flying when the operator is still learning how to fly.
- Cinder-block tether: This is the competition set-up. Tie a tether to swivel that is tied and held down by a cinder block. Place the cinder block in the middle of the field of flying.

For indoor flying details, see the link by Arducopter for some tips: http://ardupilot.org/copter/docs/indoor-flying.html

13.3 Flight testing

Arducopter has made a very nice tutorial on how to achieve a good first flight here: http://ardupilot.org/copter/docs/flying-arducopter.html

Please watch the videos and read through all tabs in the previous link before beginning testing.

We also recommend having a testing checklist/order to help with testing safely. Below is an example testing line-up for some simple testing.

1. Make sure that at least a 20 ft x 20 ft area is blocked off.
2. Everyone including those watching should have some form of eye protection (safety glasses are preferable)
3. DOUBLE-CHECK LIPO BATTERY FOR DAMAGE AND PUFFINESS BEFORE EACH FLIGHT
4. Attach two tethers on two sides of the quadrotor
5. Have two safety officers (anyone) hold on to the tethers during flight.
6. Before turning on, make sure the propellers are on securely and tightened (tighten by passing a rod or allen wrench through the prop nut (cone thingy)).
7. Once the pilot determines that everyone is in place and in a safe distance, the battery can then be plugged in
8. Pilot needs to call out everything he/she intends to do so safety officers holding the tether know what to expect.
9. Flight test
10. Motor Spin-Up
11. Check that motors spin the same rate
12. Check that motors spin the correct direction
13. Takeoff and Hover at around 4 feet
14. Check that the quadrotor holds a semi-constant position without input
15. Proceed to fly forwards and backwards and side to side
16. Check that there is no bad thrust differential causing a bad tilt or “toilet bowl” effect during flight
17. Proceed with flight safely
18. Once landed, make sure to disarm (hold yaw to left with throttle on lowest setting)

Having these checklists will ensure that proper procedure is followed and everyone is safe and unharmed in case something goes wrong. It is highly recommended that teams devise a checklist for flight testing as well as collect as much data (video, pictures, data logs, etc.) as possible to troubleshoot in case something goes wrong.

13.4 Simulation (optional)

The cheapest way to test in the most realistic setting is to set up a simulation that would simulate the flying virtually. This allows teams to test their program rigorously without any danger or risk to the quadcopter itself. However, setting up the simulation can be complicated; and therefore is only recommended for teams that have significant computer programming/systems experience. Below are some links that will help guide through the process. Please contact us if you have any questions. It is recommended that teams get flying before attempting to set up a simulation environment.

Simulation (Advanced) : http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html#sitl-simulator-software-in-the-loop

MAVProxy/Simulation (Advanced): http://ardupilot.org/dev/docs/copter-sitl-mavproxy-tutorial.html

CHAPTER 14

Technical Communication

14.1 What is technical communication?

Technical communication is, at its core, a means of conveying technical information. It's used primarily to make it easier to share a technical or complicated topic in an accessible way for a specific audience, especially in scientific and engineering fields. In the case of ARC, the topics include the mechanisms the teams design, the approach the teams take to programming the drones, and other engineering decisions made in the process of completing the tasks. The specific audience is the panel of judges (usually from engineering and/or aerospace backgrounds) reviewing the teams' presentations.

14.2 Methods of technical communication

Technical communication can range from users' manuals to technical papers to patents. For ARC, teams are expected to create and present a presentation that addresses the topics assigned.

14.3 Presentation content

When you're sharing technical information with other people, you generally want to answer a few questions:

- What are you doing? What are your goals?
- How are you achieving your goals?
- Why are you sharing this information?

Note that these questions are usually starting points - there are often other points to address as well! For ARC, the rulebook (in the appendices) includes a list of all the information we expect to see in your slides. The judges will be looking for each of these topics to be addressed, so make sure you check that you talk about everything!

14.4 Presentation style best practices

Technical communication isn't just about what you say - it's also about how you say it. In fact, how you get your information across is one of the more important parts of technical communication. If you have a lot of good information, but can't effectively share it with other people, then it's hard to make that information accessible.

Because ARC requires teams to submit and present a presentation, the following guidelines are best practices for presentations (both for the written component, on the slides, and the spoken component, when teams present to the judges).

Slides:

Here are some general guidelines for the slides that teams will submit as a PDF.

Do:

- Make sure your words are legible!
- All text should be large enough to read from a distance. We recommend a minimum font size of 28 points for slide titles and 16 points for all other text.
- Use a legible font (Arial and Times New Roman are usually good starting points), and use the same font for all text.
- The text should contrast with the background. We recommend using a white or pale slide background, with black or dark text.
- Use bullet points, with phrases/sentence fragments that get your point across without being too long.
- Include pictures! The easiest way to show what you're explaining is to show it!
  - We recommend taking photos of your drone, mechanism, or any other objects against a plain-colored background (like a white wall, or a black cloth, for example), so the photos aren't cluttered.
- Label your pictures clearly.
  - Typed labels are easier to read than handwritten labels, when possible.
  - Make sure the font/arrows are large enough to read, have contrast with the background (like white text against a black background, instead of dark blue text against a black background), and clearly point at whatever you're labeling.
- Make sure the presentation slides can stand on their own - that is, make sure someone could pick up a copy of your presentation and understand what you want to say without you needing to explain it. There should be enough information for someone to understand the general points.
- Submit as a PDF. This is required for ARC but is good for all documents. Formatting can change between different softwares and different computers, and some documents can't even be opened on different computers (for example, Apple's Slides can often only be opened on Mac computers, not computers running a Windows system).

Don't:

- Use full sentences or paragraphs on the slides. Too many words make slides busy and distracting, and your audience will be trying to read the paragraph instead of listening to what you're saying, or skim over words and miss important points.
- Make your slides hard to read. Examples include:
  - Making the font too small to read - both in the body of the text and any labels on the slide.
  - Using a patterned background. We recommend a plain white or pale-colored background, with no patterns or gradients.

  - Labelling too many things on the same photo. It's okay to use several photos if necessary!

Speaking:

Here are some guidelines to follow when teams are actively speaking and presenting their slides:

Do:

- Speak clearly.
- Face the audience, not the slides. This helps avoid reading off the slides and shows confidence in what you're saying.
- Point out important things on the slides when they come up.
- Are you talking about a specific part on your drone that's in a picture? Great - it helps to point it out so the audience knows exactly what you're talking about! If the picture is on the opposite side of the slide from where you are (for example, if you're standing on the left of the slides and the picture is on the right-hand side of the slides), a teammate standing closer can point it out for you.
- Make sure everyone gets a chance to talk! All of you have worked hard on your drone and your code - this is a chance to show off what you know! Make sure everyone gets about an equal amount of time to talk, and that everyone gets to talk about the technical work, decisions, and processes (instead of making one person do the intro/conclusion and nothing else, for example).

Don't:

- Read directly off the slides. It's okay to use the information on the slides to help structure you want to say, but you should say more than what's on the slides (aka use the bullet points as a jumping-off point for full sentences/paragraphs). Don't just read the slides word-for-word.
- Speak too fast or too quietly.
- We strongly recommend practicing several times in front of an audience (such as a teacher or friend) and/or recording one of your practices! The audience can help tell you if parts of your presentation are hard to understand, and recording a practice presentation will let you see what you're doing well and what you'll want to improve.
- We know not everyone has a loud speaking voice. If your voice is usually quiet, we recommend practicing a few times at a louder volume than usual to make sure the audience can hear you.
- Distract from the person speaking if you're not actively talking. This includes moving around a lot or talking to other teammates. You might be able to help the person speaking by pointing out important parts on the slide (if you're close to a picture, for example). Ultimately you want to show interest in what your teammates are saying!

All of these best practices come down to the same basic concepts: You want to get across your information as clearly, effectively, efficiently, and professionally as possible. With these guidelines and several practice run-throughs, you should be off to a good start with your presentations!

CHAPTER 15

Contact/Suggestions

The whole point of this tutorial site is to allow teams to have a single source to begin looking for knowledge to help teams compete in the competition. Although most of this content has been written by a few of us from the ARC Staff, we would like to invite team members/teams to add to this source!!!

If you have any suggestions/contributions/corrections you would like to make to this website, please email us at aero.robotics.comp@gmail.com. We would love to hear from you!