I 8

Avionics Getting Started Guide

McGill Rocket Team

Luca D’Angelo

Jennie Chen

Mei Qi Tang

(OG GANG)

Table of contents

Table of contents 1

Avionics Getting Started Guide 2

Embedded Avionics Getting Started Guide 7

Telemetry Getting Started Guide 13

Antenna Getting Started Guide 14

Arduino Getting Started Guide 18

Avionics Getting Started Guide

This document is meant to help you get started with developing avionics. As new recruits, your job is to obtain the necessary skills for developing more advanced circuits. Don't be afraid to ask questions while you're going through this, and never forget that Google is your best friend. The purpose of completing this guide is so the team can trust you to design, program, and build avionics hardware. You will be required to complete a mini-project as part of your initiation to the avionics subteam. The task will be to build an ejection circuit using your Arduino kits by a certain date, where we will test all of them together as a group.

DON'T PANIC AND HAVE FUN!

KNOWLEDGE DUMP #1

1. Learn to Code
    - We will be using Python and C/C++, so teach yourself as much as possible.
    - C/C++ is used to program the microprocessors.
    - Python is used for the GUI.
    - Be comfortable using the command line interface.
    - Understand how to use Git and GitHub.
    - MRT's Github: https://github.com/McGillRocketTeam
    - Python: https://www.python.org/
        - The Python Tutorial: https://docs.python.org/3/tutorial/
        - Learn Python in 10 Minutes: https://www.stavros.io/tutorials/python/
        - Virtual Environments:
            - https://virtualenv.pypa.io/en/stable/
            - https://realpython.com/python-virtual-environments-a-primer/
    - C++: http://www.learncpp.com/
        - C/C++ PDF textbook can be found in the Resources folder in the Google Drive.
    - Some text editors: VSCode, Atom, CLion, Pycharm, etc.
        - Pycharm Education Version: https://www.jetbrains.com/pycharm-edu/download/#section=mac

2. Circuits
    - Learn some basic circuit physics if you don't already know any.
    - What is a Circuit?:

        - https://learn.sparkfun.com/tutorials/what-is-a-circuit (follow the links within the website as well)
        - https://learn.sparkfun.com/tutorials/integrated-circuits
        - https://learn.sparkfun.com/tutorials/binary
        - https://learn.sparkfun.com/tutorials/voltage-current-resistance-and-ohms-law
    - Transistor: https://learn.sparkfun.com/tutorials/transistors/introduction
    - Logic Levels: https://learn.sparkfun.com/tutorials/logic-levels
    - Shift Registers: https://learn.sparkfun.com/tutorials/shift-registers
    - Analog to Digital: https://learn.sparkfun.com/tutorials/analog-to-digital-conversion
    - Communication protocols:
        - Good to be aware about this stuff in general, especially when interlinking circuits that need to communicate with each other. This will also be important for ARM development later on.
        - Serial: https://learn.sparkfun.com/tutorials/serial-communication
        - SPI: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi
        - I2C: https://learn.sparkfun.com/tutorials/i2c
    - PWM: https://learn.sparkfun.com/tutorials/pulse-width-modulation
    - Battery Ratings: https://www.allaboutcircuits.com/textbook/direct-current/chpt-11/battery-ratings/
    - Reverse Polarity Protection:
        - http://www.ti.com/lit/an/slva139/slva139.pdf
        - https://www.instructables.com/id/Reverse-Polarity-Protection-Circuits/
    - Connectors: https://learn.sparkfun.com/tutorials/connector-basics (useful to know when designing your own PCBs and how you want to power/connect things)
    - GreatScott video series (a YouTube playlist) on circuit basics: https://www.youtube.com/playlist?list=PLAROrg3NQn7cyu01HpOv5BWo217XWBZu0

3. Arduino
    - What is an Arduino?: https://learn.sparkfun.com/tutorials/what-is-an-arduino
    - Install the Arduino IDE: https://www.arduino.cc/
        - https://learn.sparkfun.com/tutorials/installing-arduino-ide

- Play with the Arduino starter kits, go through some online tutorials, watch some YouTube videos, and read through the Arduino programming documentation for help.
- Get comfortable using libraries for electronic components like sensors.
- Check out the Arduino starter-kit user manual in the Resources folder of the Google Drive and do some of the projects listed on your own. We will be having a joint Avionics+Payload tutorial session too for practice.
- Check out the FML document in the Resources folder

4. The Ejection Circuit
    - Using the Arduino, design and program the pressure-sensor based ejection circuit.
    - Show off your design on testing day (TBD).
    - If you fail, fix it and try again.
    - Check out the Ejection Circuit Tutorial document in the Training folder of the Avionics Google Drive folder if you want a head-start before the actual tutorial.
    - Relays:
        - https://en.wikipedia.org/wiki/Relay
        - https://www.youtube.com/watch?v=n9renPKEtUc
    - Pressure Sensor:
        - BMP180: https://learn.sparkfun.com/tutorials/bmp180-barometric-pressure-sensor-hookup-

5. PCB Design
    - Learn to use EAGLE for designing through-hole and surface-mounted PCBs by reading the SparkFun tutorials. This section will also teach you the importance of reading datasheets for different electrical components, most importantly the microprocessor.
    - Note that the Sparkfun tutorials are a bit outdated. The most recent version of the Sparkfun library is more up-to-date than the one used in the tutorials. This just means that the tutorial will reference a part name that actually has a different name in the most recent library, so it takes a bit longer to find the part they want you to use.
    - Getting used to EAGLE is a good place to start for beginners as it is simpler and relatively simple/easy to use. But for proper engineering practice, it is best to use Altium. This software is more advanced.
    - Altium Beginner Tutorial Playlist:

    - https://www.youtube.com/watch?v=KpgTud1iQ-4&list=PLXvLToQzgzdfKKQn2wmpuSXz6sROQmO6R
- More Advanced Tutorials: https://www.youtube.com/playlist?list=PLXvLToQzgzdcyHW7qScRZOlYaK3VimeLE
- EAGLE: https://www.autodesk.com/products/eagle/overview:
    - Luckily, because you are a student, Autodesk offers a free student edition for download. Use your McGill student email address to make an account.
- Download the **SparkFun EAGLE settings and library** folders from their GitHub page: https://github.com/sparkfun
- Download the **Adafruit EAGLE library folder** from their GitHub page: https://github.com/adafruit/Adafruit-Eagle-Library
- Or follow this guide which goes through the above two steps: https://www.autodesk.com/products/eagle/blog/library-basics-install-use-sparkfun-adafruit-libraries-autodesk-eagle/
- PCB Basics (Terminology): https://learn.sparkfun.com/tutorials/pcb-basics
- Sparkfun EAGLE Guide: https://www.sparkfun.com/EAGLE
    - Using EAGLE to design PCBs 1: https://learn.sparkfun.com/tutorials/how-to-install-and-setup-eagle
    - Using EAGLE to design PCBs 2: (Through Hole) https://learn.sparkfun.com/tutorials/using-eagle-schematic
    - Using EAGLE to design PCBs 3: (Through Hole) https://learn.sparkfun.com/tutorials/using-eagle-board-layout
    - Using EAGLE to design PCBs 4: (SMD) https://learn.sparkfun.com/tutorials/designing-pcbs-advanced-smd#gerber-generation
    - Using EAGLE to design custom footprints: (Optional, not necessary) https://www.youtube.com/watch?v=VxMV6wGS3NY
- **PCB Trace Width Calculator:** https://www.digikey.ca/en/resources/conversion-calculators/conversion-calculator-pcb-trace-width
- Learn to solder
    - Soldering various types (video): https://www.youtube.com/watch?v=VxMV6wGS3NY
    - Soldering PCBs with a stencil: https://www.youtube.com/watch?v=WDIqtGMROjM
- Clone a simplified Arduino Uno board

- Cloning an Arduino on a Breadboard 1 (for reference): https://www.arduino.cc/en/Main/Standalone (note there is a small mistake in this guide. A capacitor is needed on the DTR pin as seen in the next link)
- Cloning an Arduino on a Breadboard 2 (for reference): https://www.youtube.com/watch?v=sNIMCdVOHOM
- Burning the bootloader: https://www.arduino.cc/en/Tutorial/ArduinoISP
- Arduino Uno Pin Mapping:
    - https://www.arduino.cc/en/Hacking/PinMapping168
- Voltage Regulator: https://www.modmypi.com/blog/how-to-use-voltage-regulators-in-a-circuit

6. Parts/Supply Stores and PCB/Stencil Manufacturers

- Electronics+Footprints+BOM: https://octopart.com/
- Electronics: https://www.digikey.com/
- Electronics: https://www.sparkfun.com/
- Electronics: https://www.adafruit.com/
- Mechanical hardware: https://www.mcmaster.com/
- Component Footprints: https://www.ultralibrarian.com/
- PCB manufacturer (user friendly): https://oshpark.com/
- PCB manufacturer (local): http://golabo.com/en/
- Stencil manufacturer: https://www.oshstencils.com/

7. Moving Forward

- If you made it this far, AWESOME!!! The McGill Rocket Team now trusts you to design, program, and build avionics hardware but...
- Now, choose a project you want to work on.
- Feel free to start reading the other getting started guides below if you are curious. Depending on which project you decide to join, you will end up having to read them anyway.

Embedded Avionics Getting Started Guide

This document is meant to help you get started with developing embedded avionics with the ARM microprocessor. Ideally you have already completed the Avionics Getting Started Guide above. Hopefully, you and Google are still best friends.
WORK IN PROGRESS.
KNOWLEDGE DUMP #2

1. The ARM Microprocessor
    - Switching over to an ARM processor gives us more speed and memory resources, however, the complexity increases substantially from the easy-to-use Arduino.
    - The processor we want to use is the STM32F767ZI, which has a clock speed of 216 MHz - essentially 10 times faster than the Arduino Uno processor and a bit overpowered for our needs; it is manufactured by STMicroelectronics. This processor is part of the Cortex-M7 series of high performance ARM processors and has 144 pins.
    - However, due to its complexity, we will use a simpler ARM processor and learn as much as possible.
        - The first option is the STM32F042K6, which has 32 pins and a clock speed of 48 MHz. This will only be used for learning purposes.
        - The second option is the more complicated STM32F405RG, which has 64 pins and a clock speed of 168 MHz. This will be used for current development and prototyping.
    - We want to design circuits around the processor, and to program it. In terms of programming there are two approaches: low-level or mbed. mbed is an IoT framework for ARM processors that provides a higher-level API, based on the low-level API language needed for programming at the bare-metal. The mbed platform also provides boards, modules, and other components to interface with mbed-enabled boards.
    - The main components to understand about the ARM processor:
        - Essentials: power supply, voltage regulator, decoupling capacitors, interrupts, pull-up/down resistors, flash memory, SRAM
            - Decoupling capacitors (super important):
                - https://www.autodesk.com/products/eagle/blog/what-are-decoupling-capacitors/

            - https://www.modmypi.com/blog/how-to-use-voltage-regulators-in-a-circuit
            - https://macrofab.com/blog/bypass-caps-decouple-your-way-to-cleaner-power/
        - Interrupts:
            - https://en.wikipedia.org/wiki/Interrupt
            - https://os.mbed.com/docs/v5.9/tutorials/application-flow-control.html
        - Pull-up/down resistor:
            - https://en.wikipedia.org/wiki/Pull-up_resistor
            - https://learn.sparkfun.com/tutorials/pull-up-resistors
        - Memory:
            - https://en.wikipedia.org/wiki/Flash_memory
            - https://en.wikipedia.org/wiki/Random-access_memory#Shadow_RAM
            - https://en.wikipedia.org/wiki/Static_random-access_memory
    - Programming connection: JTAG, SWD
    - Communication: UART, USART, I2C, SPI, USB
    - Clock: External resonators
- Resources we found useful:
    - Arm Architecture Fundamentals (complicated): https://www.youtube.com/watch?v=7LqPJGnBPMM&t=2397s
    - Mbed: https://www.mbed.com/en/
        - Read the programming docs for development.
        - Tutorials: https://os.mbed.com/docs/v5.9/tutorials/index.html
    - STMicroelectronics: https://www.st.com/
    - Info: Arm Architecture Fundamentals
        - https://www.youtube.com/watch?v=7LqPJGnBPMM&t=2397s
    - Info: https://www.electronicshub.org/arm-introduction/
    - Info: https://newbiehack.com/MicrocontrollerTutorial.aspx
    - Info: http://embedded-lab.com/
    - Info: Check the processor datasheets folder in Avionics>Design Files>Arm Development. Processor footprints are downloaded using the Ultra Librarian website.

2. Debugging
    - In order to program the ARM processor on our PCBs we need a special type of connector. The two main types of connectors are called SWD and JTAG. When designing the PCB schematic, you would include the SWD/JTAG connector part.
    - JTAG is the standard for most microprocessors while SWD is specific to ARM microprocessors. These interfaces allow you to program and debug the code as it runs on the processor itself. The actual programmer devices need to be purchased from the processor manufacturer usually like the ST Link connector.
        - ST Link programmer: https://www.st.com/en/development-tools/st-link-v2.html
    - An open-source alternative and simpler method to uploading code to the ARM processors is through the Black Magic Probe (BMP):
        - Info: https://github.com/blacksphere/blackmagic/wiki
        - Purchase: https://1bitsquared.com/products/black-magic-probe
        - GNU-ARM Toolchain (required for 'uploading' program to microprocessor, read through BMP GitHub docs for a user guide): https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads
        - Usefull GDB Commands: https://github.com/blacksphere/blackmagic/wiki/Useful-GDB-commands
    - Since you are uploading code to the microprocessor yourself and not through an IDE like the Arduino, you can actually get an understanding of what's going on under the hood. When the code compiles, whether you do it on the mbed platform or in an IDE (see next numbered point below), a binary zip file is created. This file is what you have to upload to the microprocessor.
    - There is a way to program the processor using a USB to Serial connection and with the help of other third party software, you can look it up yourself and implement it if you want, but the BMP solution seems simpler at this point.

3. Development Environment
    - In order to write programs for the ARM processor we need a text editor. The mbed website provides an online IDE and compiler. The online version allows you to compile your code to the binary zip file and download it. However, it does not offer the features of a modern IDE and

there is no offline version. Most alternatives require a license to use, but thankfully PlatformIO exists - an open-source IDE for embedded development.

- PlatformIO: https://platformio.org/platformio-ide (Download the recommended version using VSCode, follow the guides)
- VSCode: https://code.visualstudio.com/
- When using the development boards, you can upload directly from PlatformIO to the board with one button click.
- The Black Magic Probe can be used with PlatformIO to streamline debugging and 'uploading' of binary files to microprocessor.

4. ARM Nucleo Development Board
    - The development board we are considering to use for prototyping the different avionics circuits and testing/debugging code:
        - **(144pin, 216 MHz, STM32F767ZI) M7: FUTURE DEVELOPMENT**
            - Mbed board: https://os.mbed.com/platforms/ST-Nucleo-F767ZI/
            - Documentation: https://www.st.com/en/microcontrollers/stm32f767zi.html
        - **(64pin, 168 MHz, STM32F446RE) M4: NEED TO RESEARCH**
            - Mbed board: https://os.mbed.com/platforms/ST-Nucleo-F446RE/
            - Documentation: https://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32-high-performance-mcus/stm32f4-series/stm32f446/stm32f446re.html
        - **(32pin, 48 MHz, STM32F042K6) M0: FOR LEARNING PURPOSES AND RESEARCH**
            - Mbed board: https://os.mbed.com/platforms/ST-Nucleo-F042K6/
            - Documentation: https://www.st.com/en/microcontrollers/stm32f042k6.html
    - You drag-and-drop the binary file to upload the program on the board, this feature is only available with mbed-enabled development boards.

5. MRT Avionics Development Board
    - The ARM version of the cloned Arduino. Going through this tutorial will teach you about how to design a circuit and an SMD PCB using a simpler ARM processor with 32 pins. If it works, the board can be used as a

development board for uploading and debugging your code on the processor.

- Understanding this will help you to understand the other ARM processors that have more pins and faster clock speeds.
- Tutorial (32pin ARM processor - STM32F042K6):
    - Part 1: https://predictabledesigns.com/tutorial-how-to-design-your-own-custom-microcontroller-board-video-part1/
    - Part 2: https://predictabledesigns.com/tutorial-how-to-design-your-own-custom-microcontroller-board-video-part-2/
    - Datasheets:
        - Processor product page: https://www.st.com/en/microcontrollers/stm32f042k6.html
        - Specs: https://www.st.com/resource/en/datasheet/stm32f042k6.pdf
        - Hardware development: https://www.st.com/content/ccc/resource/technical/document/application_note/c9/19/d7/b8/6b/0e/4c/d3/DM00051986.pdf/files/DM00051986.pdf/jcr:content/translations/en.DM00051986.pdf
    - This tutorial really helps you go through the design process for making a circuit using the ARM processor:
        - Thinking about the different components you want to include
        - How do the components connect to the processor (how do they communicate with it)
        - Looking at the processor datasheet for the pin layout and how to connect the power supply, decoupling capacitors, programmer, clocks, etc.
- Tutorial (64pin ARM processor - STM32F4): https://makezine.com/2016/09/19/design-microcontroller-circuit/
- (Will include links to our GitHub repository for EAGLE schematic and board files when completed so you can use them as an example to follow)

6. Processor Quick Facts

	Arduino Uno	STM32F767ZI	STM32F446RE	STM32F042K6
Processor	Atmega328P	Cortex-M7	Cortex-M4	Cortex-M0
Clock Speed	16 MHz	216 MHz	168 MHz	48 MHz
Operational Voltage	5.0 V	1.7-3.6 V	1.7-3.6 V	2.0-3.6 V
Flash Memory	32 kB	2 MB	512 kB	32 kB
SRAM	2 kB	512 kB	128 kB	6 kB
Pins	20	144	64	32

Telemetry Getting Started Guide

This document is meant to help you get started with learning everything you need to know about telemetry. Ideally you have already completed the first Avionics Getting Started Guide above. **WORK IN PROGRESS.**
KNOWLEDGE DUMP #3

1. Radio modules: transmitters and receivers

Antenna Getting Started Guide

This document is meant to help you get started with learning everything you need to know about antenna design.
WORK IN PROGRESS.
KNOWLEDGE DUMP #4

1. What is an antenna:
    1. (Wikipedia): It is the interface between radio waves propagating through space and electric currents moving in metal conductors. In transmission, a transmitter supplies an electric current to the antenna's terminals, and the antenna radiates the energy from the current as electromagnetic waves (radio waves). In reception, an antenna intercepts some of the power of an electromagnetic wave in order to produce an electric current at its terminals, that is applied to a receiver to be amplified.
2. Antenna properties (EM theory):
    - ➢ Please watch the Antenna Fundamentals by the Film Board of Canada made for the Royal Canadian Air Force (Extremely well explained) in 3 parts:
        i. https://www.youtube.com/watch?v=7bDyA5t1ldU
        ii. https://www.youtube.com/watch?v=md7GjQQ2YA0
        iii. https://www.youtube.com/watch?v=9iV_YICgifA
    - ➢ Microstrip Patch Antenna Parameters and Experimental Setup is a simple and concise guide explaining basic antenna theory and microstrip patch antennas:https://drive.google.com/file/d/1FZHF71uBQAGWRa_NreMzCkhCnqm_Mqfv/view?usp=sharing
    - ➢ If after watching the videos and reading the document you are still confused about basic antenna theory concepts, Google is your friend and you should search online. Here's a list of important concepts you should be familiar with in order to design antennas:
        i. Frequency, wavelength, channel
        ii. Gain (possible units: dB, dBi, dBd)
        iii. Polarisation and directivity (vertical, horizontal, circular)
        iv. Bandwidth, Impedance, Radiation pattern
        v. Resistance (loss and radiation)
        vi. Surface waves

vii. Interference

➢ Examples of online resources you can read on:

i. https://www.pulseelectronics.com/antenna_basic_concepts/
ii. https://www.radio-electronics.com/info/antennas/
iii. http://www.antenna-theory.com/

3. Types of radio propagation waves:

As a form of electromagnetic radiation, like light waves, radio waves are affected by the phenomena of reflection, refraction, diffraction, absorption, polarization, and scattering when interacting with the different layers of our atmosphere. Their behaviour varies at different radio frequencies.

1. Line-of-sight (direct modes)
2. Ground wave (surface modes)
3. Skywave (ionospheric modes)
4. Tropospheric waves (tropospheric modes)

4. Types of antennas:

These are just a few of the existing antenna types. Each type of antenna has a specific behavior and each antenna has its own different parameter values or characteristics, which is what makes antenna design a fun challenge.

➢ https://www.elprocus.com/different-types-of-antennas-with-properties-and-thier-working/

1. Omnidirectional or directional (high-gain)
2. Single or Multiband
3. Dipole, monopoles
4. Microstrip patch
5. Yagi
6. Helical, double crossed
7. Antenna arrays: Microstrip patch arrays
8. Reflector antenna

5. Feed and transmission line:

1. Impedance matching: one important concept to note while designing circuits or electronic systems is to have impedance matching between the impedance of the source and the impedance of the load. This would allow us to have maximum power transfer. If impedance is not matched, standing waves could be forming and would be consuming power. In our case, that would be to match the impedance of the transmitter/receiver circuit to the impedance of the transmission line (eg. coax cable), which would also need to match the impedance of the antenna feedline as well as of the antenna.

2. Coaxial cables: 50 ohms coax cables are generally standard for achieving impedance matching with the commercial antennas. Coax cables are a very common type of transmission line used in a lot of radio applications.
3. PCB (printed circuit board) feedlines: These type of transmission lines are generally seen in microstrip antenna designs.

6. The (other) laws of Physics & effects:
    1. Doppler effect: The Doppler effect will be a potential issue when the rocket goes a few times faster than the speed of sound. For now, after calculating it, the frequency only got shifted by an order of 10^-5, which is minimal and can be ignored.
    2. RF Blackout / plasma sheath formation: During the re-entry of a hypersonic vehicle to the atmosphere, ionized air could form a sheath around the rocket due to high temperature and interfere with the radio waves, causing RF (Radio frequency) blackout. The general idea is that the ionized particles will collide and diffract the radio waves, gaining energy from them, then creating partial or total loss in the radio transmission. There are a few articles in the *Avionics/Resources/Antennas* folder talking about it if you'd like to learn more. However, this applies to vehicles going at least 2.5x faster than the speed of sound (Mach number of 2.5) while we're only going at about Mach 1.7.
7. Our goal:

    We need to have antennas with a good gain (as high as we can realistically get) that will transmit on the right frequency to satisfy our set of requirements. The terrain at competition is relatively flat but will have bushes (1-2m) and small hills (about 5-10m). The air will be very sandy and small sandstorms could happen during flight. Weather will be very hot but not humid. The sky will be generally clear with no clouds. Our rocket could fly up to 10km. Therefore, we need antennas with a high gain to be able to receive the small signals transmitted 10km away. The radio waves transmitted when rocket is at ground level also can't be blocked by hills.
8. Tools and softwares:

    It is not realistic to be calculating an antenna's performance or radiation pattern by hand so softwares would be used to do performance calculations. Matlab has a plugin made for Antenna design so we can look into that.
9. Other resources:

    I've put some useful electronic handbooks and articles under *Avionics/Resources/Antennas* so go read them if you want a more throughout understanding of antenna theory and rocketry specific antenna design.
10. Examples of amateur projects:

1. The Thought Emporium YouTube channel has made several awesome antenna/radio telescope from scratch:
    i. https://www.youtube.com/watch?v=o6WHhqDHSQ4
    ii. https://www.youtube.com/watch?reload=9&v=cjClTnZ4Xh4
2. This guy has a lot of radio projects. One really nice video shows how he made a circularly polarized microstrip patch antenna:
    i. https://www.youtube.com/watch?v=4iGBzx3TGRM
3. This personal website (with photos) shows how the antennas were integrated on the carbon fiber fins
    i. http://gag.com/rockets/airframes/YikStik3/

Arduino Getting Started Guide

KNOWLEDGE DUMP #5
Check out this cool link:
https://docs.google.com/document/d/1iZU2BAMZ2XO1TYKbicYHP_ybh39lTT9yfzI-Y55UP7A/edit?usp=sharing

The contents in that link are not suitable for this VERY SERIOUS document.
Which is why it's in another folder :'(