Daniel McGarry

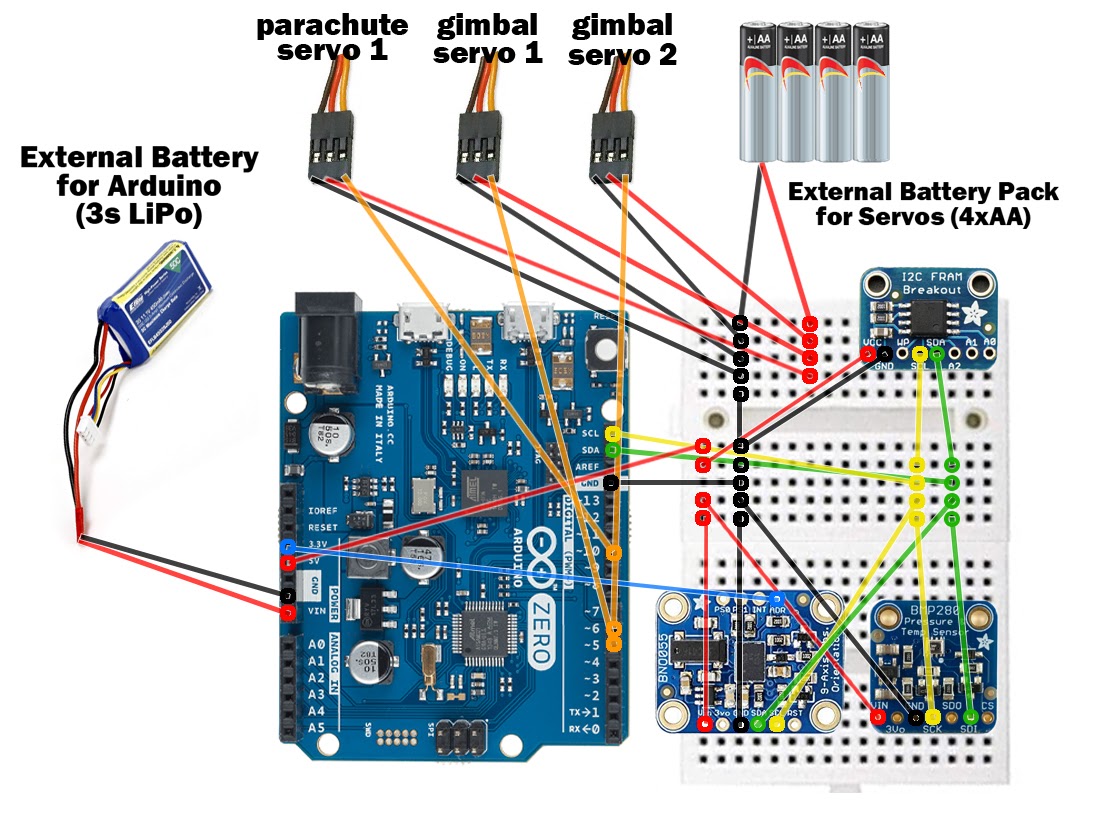
Advisor Dr. Philip Voss

FURSCA End of Summer Report

July 23, 2021

For this project, my partner Nathaniel Jennings and I were attempting to create a model rocket that emulates some of the more “modern” engineering techniques that have revolutionized the aerospace industry within the past decade. The concept that we both believed to be most achievable on a smaller scale and limited budget was thrust vector control (TVC). For rockets, TVC brings the capability of auto-stabilization through changing the thrust output of the engines by either throttling or gimbaling (rotating) them to keep the rocket on it’s intended trajectory. Our intention was to make a TVC system by creating a gimbal to house a typical model rocket motor and change the angle of thrust for said motor using a computer onboard the rocket. Nathaniel would be in charge of all hardware components necessary for the rocket and all testing needs, (motor gimbal, parachute deployment system, computer storage compartment) and I would be in charge of all software needs (auto-stabilization algorithm, testing protocol for all necessary systems before each flight, data collection during flight). Lastly, we believed it necessary to publicly share all our gained hardware and software knowledge to help advance the field of model rocketry.

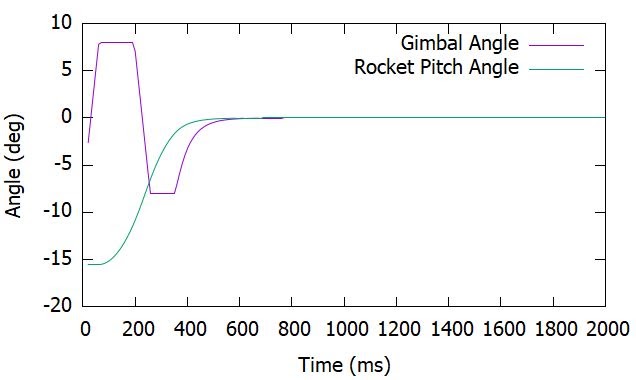
When starting this project, I thought I had the necessary programming knowledge to surely do almost anything that would be required of me. Unfortunately, Python was the only coding language I was familiar with, and I soon figured out I would need to quickly learn C++ to communicate with the Arduino Zero microcontrollers that Nathaniel and I had chosen to be our onboard flight computers. After roughly a week of teaching myself C++, I felt comfortable enough to begin working with the Arduinos. The first program that I created was a systems testing protocol {1}. This allowed us to make sure that the altimeter, barometer, orientation sensor, data storage chip, and servos controlling the gimbal were working properly before each flight. Before testing this program, I first designed a schematic and wired all of the proper components together. This wiring schematic is shown below.



Next, I began working on the auto-stabilization program. For this, I chose to use a proportional-integral-derivative (PID) algorithm. This type of algorithm requires an input to stabilize the output value around a chosen target value. In our case, the input is the rocket’s orientation, the output is the gimbal angle to control the motor’s thrust, and the target value is 0 degrees (straight up with no declination). In addition, the equation allows for adjusting three variable values or “gains” Kp, Ki, and Kd that bias past, current, and future calculations respectively. Equation 1 showcases the entire PID algorithm equation.

(1)

To further understand how this complex equation worked, I first created a dummy program that used the equation, but in a much simpler way than the way I intended to do with the rocket. Once I felt comfortable with the math, I created a flight simulation program {2} that used the accurate information about our rocket to aid with tuning the PID gains without having to waste time or money on motors to tune them manually with no help. This program would output graphs that were visually helpful for understanding how changing the different PID gains affected how the simulated rocket behaved. One of these graphs is pictured below.



Next, I transferred the useful code from this simulation to the Arduino coding software and modified it to use the actual data from the orientation sensor to output an angle for the servos on our rocket’s gimbal to move to {3}. As a side project I also created a program that helped aid in calibrating the servos when first attached to our gimbal {5}. Around the time I finished this program we received our memory chips in the mail, so I created a program to read the data captured during the flights {6}, and a program to delete the memory once we had saved it elsewhere {7}. Lastly, it was time to create a complete flight program that contained everything required for an entire launch {8}.  A rocket’s flight can be broken down into 6 stages including ground idle, powered flight, unpowered flight, ballistic descent, chute descent, and landing. I outlined this final program into 6 major sections, each corresponding to one of these flight stages. Luckily, I was able to reuse the systems testing protocol program for the majority of the ground idle stage and the PID program for the powered flight stage, but I had to create the other stages from scratch. It took some time, but I finished right around when Nathaniel had finished creating the last iterations of the hardware.

As of the writing of this report. Nathaniel and I have completed successful PID tuning and hardware ground tests using rocket motors with a 2.5 second burn time. Successful flight stage tests were completed using an elevator, as well as a parachute deployment test in the Science Center’s atrium. After three attempts, we fortunately were able to get the PID algorithm to execute properly during an unaided flight. By the end of this week, or possibly later this fall, we hope to conduct more launches using more powerful and longer-burning motors to further demonstrate the capabilities of our rocket.

This project was one of the greatest opportunities I’ve ever had to learn and solve problems. Being able to apply so much that I was taught in prior classes was great. It was very, very frustrating at times when I felt lost or stuck, but being able to sit down, work things out on my own, and see the results at the end is very rewarding. In addition, I learned and created sophisticated programs in a programming language I had no prior experience with, learned more about topics such as control theory, and improved my communication skills as a collaborative partner while working with Nathaniel. These are all things that I know will prepare me for my future career. I look forward to sharing even more of our successes, challenges, and discoveries at the Elkin Isaac student research symposium next spring.

All the programs and relevant files that I created during this project are linked below.

{1} - Onboard Systems Testing Protocol Program <https://github.com/DM-FURSCA-2021/Onboard-Systems-Testing-Protocol/blob/main/Combined_Sensor_Test.ino>

{2} - PID Simulation Program <https://github.com/DM-FURSCA-2021/PID-Algorithm/blob/main/CPPPID.cpp>

{3} - Arduino PID Program

<https://github.com/DM-FURSCA-2021/PID-Algorithm/blob/main/ArduinoPID.ino>

{4} - Arduino PID Program Header File

<https://github.com/DM-FURSCA-2021/PID-Algorithm/blob/main/ArduinoPID.h>

{5} - Servo Alignment Program <https://github.com/DM-FURSCA-2021/Servo-Alignment-Program/blob/main/Servo%20Alignment%20Program.ino>

{6} - FRAM Chip Memory Reading Program <https://github.com/DM-FURSCA-2021/i2c_FRAM_Chip_Files/blob/main/Adafruit_i2c_FRAM_chip_memory_read.ino>

{7} - FRAM Chip Memory Deleting Program <https://github.com/DM-FURSCA-2021/i2c_FRAM_Chip_Files/blob/main/Adafruit_i2c_FRAM_chip_memory_delete.ino>

{8} - Full Flight Program <https://github.com/DM-FURSCA-2021/Full_Flight_Program/blob/main/Full_Flight_Program.ino> -

{9} - Full Flight Program Header File <https://github.com/DM-FURSCA-2021/Full_Flight_Program/blob/main/Full_Flight_Program.h> -