Teaching an Arduino to Fly

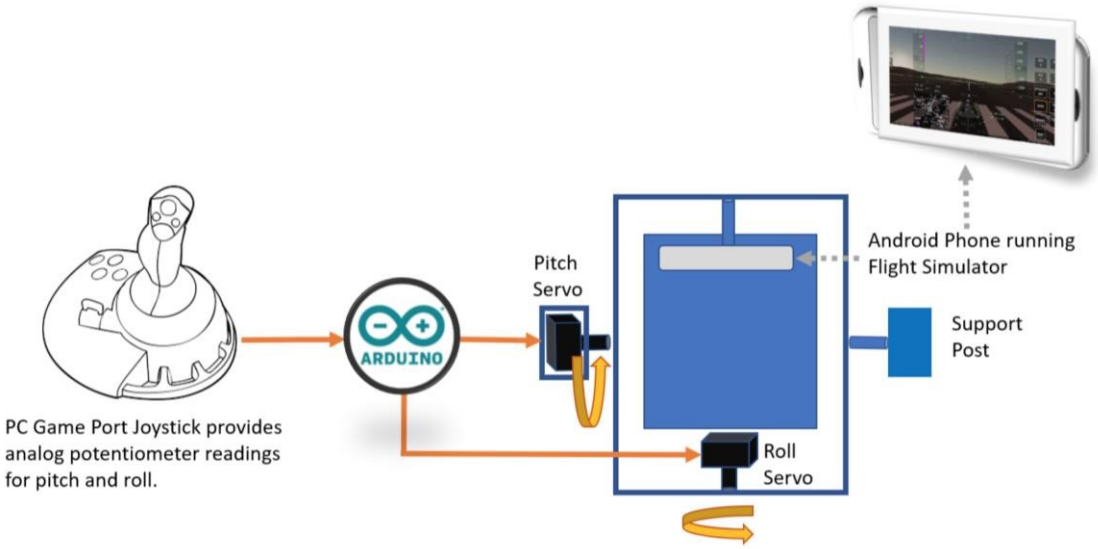
AN ARDUINO-CONTROLLED MODEL FLIGHT SIMULATOR

Leo Salemann| HCDE 539 Physical Computing | Dec 11, 2018

Description

There's lots of model airplanes, but what about a model flight simulator? This Project combines an Arduino, a joystick, a Flight Simulator running on an Android phone, and a plastic airplane model with a Lego pilot. The Arduino reads axis angle from the joystick to drive servos which change the orientation of a small motion platform. The platform has Android Phone running Infinite Flight, a flight simulator designed for mobile devices. As the platform pitches and rolls, the flight simulator responds as if the user were holding the phone. Components include Adafruit Metro (Arduino Uno clone) which drives drive 2 heavy-duty servos, capable of "lifting" the platform. The joystick is an old PC-game port type, providing analog pins that can be read like any potentiometer.

System Diagram

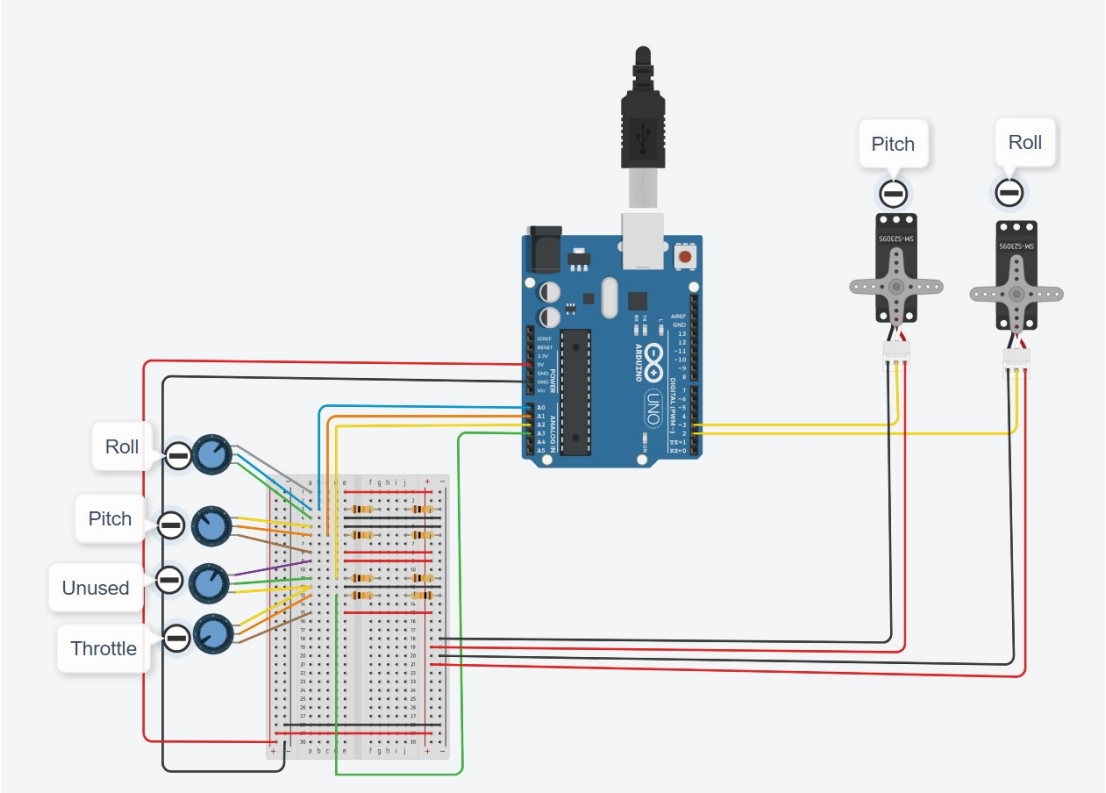


Implementation

The PC Game port joystick provides analog pins with potentiometer readings for pitch (up/down) roll (lean left/lean right) and throttle settings. The Arduino reads these through analog its analog inputs, then does some processing to ensure they range from 0 to 1023 with a mid-point at 512. Joystick values are then mapped to servo angles, which can range from 0 to 180 degrees, but can truncated by “Platform motion limit” variables to something like 45 to 135. The motion platform holds a smart phone (Android in this case by iPhone would work) running a mobile flight simulator called Infinite Flight. Normally, the user holds their phone like a flight yoke, rocking it back and forth or left and right to fly. The motion platform provides the same motions, while giving the user a physical flight yoke to control. A plastic aircraft model is placed on the platform as well, mostly for artistic purposes.

Infinite flight can be controlled by Bluetooth game controllers as well as the phones accelerometer. An attempt to route the flight yoke analog throttle signal via Arduino and Bluefruit proved unsuccessful, so a store-bought game controller was pressed into service to provide throttle, flaps, spoilers, and landing gear control.

Circuit Diagram



Available online at https://www.tinkercad.com/things/eyIbdlANTBl-2dof-motion-platform

Software

Each component of the system was developed individually, with a single Arduino Sketch .

`Joysticktest.ino` was one of the first sketches, based on the temperature sensor demo at https://blogs.uw.edu/fizzlab/technology/examples accessed 11/21/2018. The intent was simply to connect the joystick potentiometer pins to the Arduino analog inputs, and view the data through serial monitor and plotter. Serial monitor quickly revealed problems with "centering" (neutral position for each axis was around 250-300 instead of 512) along with "low-end truncation" (joystick axes would go down to around 122 but not to zero). Function **`remapJoyAxis()`** was added to separately remap the "top part" and "bottom

part" of each joystick axis, to get a behavior closer to 0-1023 with 512 in the middle. Pin assignments and **`remapJoyAxis()`** were later integrated into the main simulator code.

`ServoTest.ino` was based on TinkerCAD circuit "arduino servo motor avec potonsiométre" by user Ounis.Brahim, accessed 11/17/18. The circuit and sketch allow an Arduino to control up to three servos, based on a reading of one potentiometer per servo. For this project, only two of the three servos were used, forming the "back half" of the flight simulator.

`JoystickServo.ino` is where joystick-read and servo-write come together. Values are read from joystick servos, processed, and sent to servos to control the motion platform. The same **`remapJoyAxis()`** from **`Joysticktest.ino`** is used here to ensure a full range of 0 to 1023 on the joystick side, with a 512 midpoint. These are then (re)remapped to servo target angles ranging from zero to +/- **`Platfrom_RollLimit`** and **`Platform_Pitchlimit,`** which are typically 90 to 45, based on how "wild" you want the platform to behave. The servos are read for their current angles, but are not given new targets unless the new-old delta exceeds the **`NULLZONE_roll`** or **`NULLZONE_pitch`** as appropriate. The **`NULLZONE`** variables were an attempt to suppress "jitter" in the platform, with mixed results. The original intent of **`JoystickServo.ino`** was to be another test harness focused on joystick/servo interaction, but it became the main simulator code once Bluefruit experiments proved unsuccessful.

`BlueTooth/factoryreset.ino` was taken straight from the Adafruit Bluefruit LE UART Friend page at https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-uart-friend/factory-reset to ensure the firmware was up to date. It seemed to work, after figuring out which of the three different ways to initialize the Bluefruit would work for my configuration. Software serial seemed to do the trick.

`BlueTooth/hidmouse.ino` was an attempt to emulate a Bluetooth mouse, in the hopes that Infinite Flight could interpret the joystick throttle axis as a mouse-move. Although proper Bluetooth AT commands were observed via Serial Monitor, Infinite flight would not recognize mouse motion as an acceptable input axis. Code was downloaded from https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/ble-services#at-plus-blehidmousemove-14-35 and used with minimal modification.

`BlueTooth/throttle.ino` was an attempt to emulate a Bluetooth game pad. It was found to only emulate discrete button presses as opposed to continuous axis motion. Code was based on **`hidmouse.ino,`** with the **`AT+BleHidMouseMove`** replaced with **`AT+BLEHIDGAMEPAD`**.

`FlightSimulator.ino` was an attempt to integrate **`JoystickServo.ino`** with **`BlueTooth/hidmouse.ino`**, yielding a program that would read joystick pitch, roll, *and* throttle axes, control the servos via pitch and roll and send throttle increment/decrement commands via Bluetooth gamepad buttons. Although Infinite Flight could recognize the

Bluefruit as a game pad and map gamepad buttons to throttle increment/decrement; the actual Arduino code was sending button-press events but never button releases. The issue could not be resolved by the project's due date, so in the project's master branch, the contents of **`FlightSimulator.ino`** was replaced with **`JoystickServo.ino.`** The SevoJoystickLashUp branch still contains the original **`FlightSimulator.ino`** with its Bluetooth experimental code.

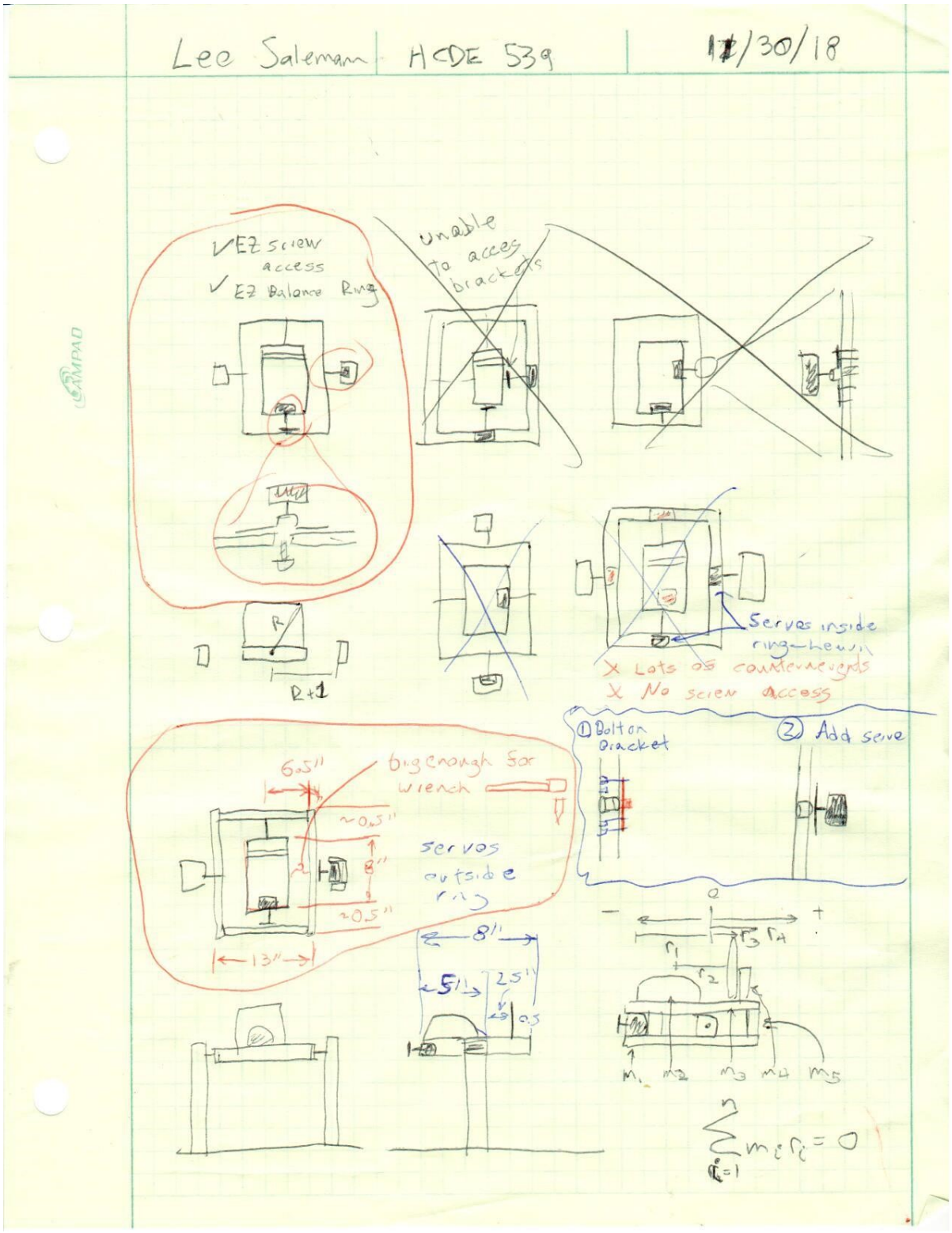
`3DOF_MotionPlatform.ino` is virtually a copy of **`JoystickServo.ino`**, with the addition of one more "throttle servo" for demonstration purposes on a TinkerCAD circuit.

Gallery

DESIGN

The transition from prototyping to production brought on a "measure twice; cut once" approach which then became "measure five times; cut three times." Even after converging onto a platform-within-ring design, there were still multiple alternatives to consider with regards to servo placement. The optimal solution placed the roll servo in the back of the central platform and the pitch servo in the right-side support post. This configuration kept the outer ring as light as possible, helped balance the weight of the mobile phone in front, and provided an overall better aesthetic than having the support posts fore and aft.

Lee Saleman | HCDE 539 | 11/30/18



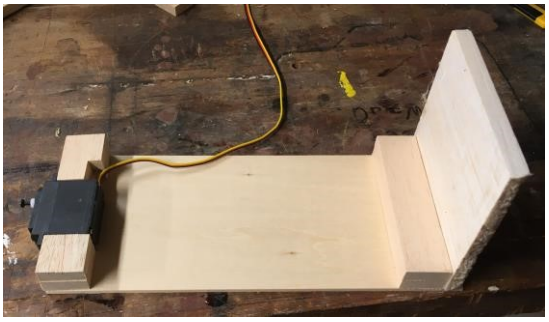
BUILD-OUT

Paper Prototype. Simple cardboard platform balanced on a pencil to assess the degree of platform motion required for a good flying experience. Also used to evaluate Flight Simulators and aircraft models. Infinite Flight's A-10 proved to be fast and responsive enough for good flying, but not so skittish as to be difficult for beginners. Full video available on YouTube.

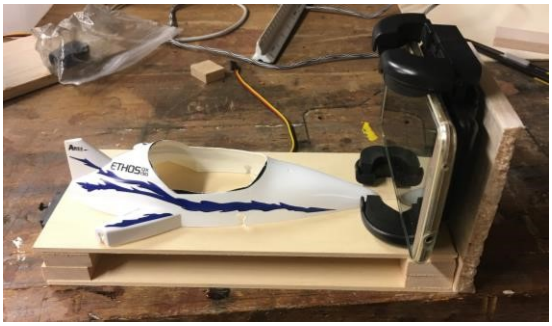
Joystick breakout. A pair of female-to-male jumper wire ribbons directly connect PC game port joystick pins to the adafruit breadboard. No physical changes to joystick required.

Single-Axis Test. Joystick, Adafruit Metro, one servo, and balsawood platform to validate that servo is strong enough to move platform holding mobile phone and counterweight for balance. Full video available on YouTube.

Bluetooth configuration. Infinite flight can be configured to read inputs from the phone accelerometer and external Bluetooth devices simultaneously. Each flight control axis (pitch, roll, …) can be configured separately, as can numerous other controls such as flaps and landing gear. Full configuration video available on YouTube.



Platform Construction. Final balsa wood platform with roll servo embedded in the back.

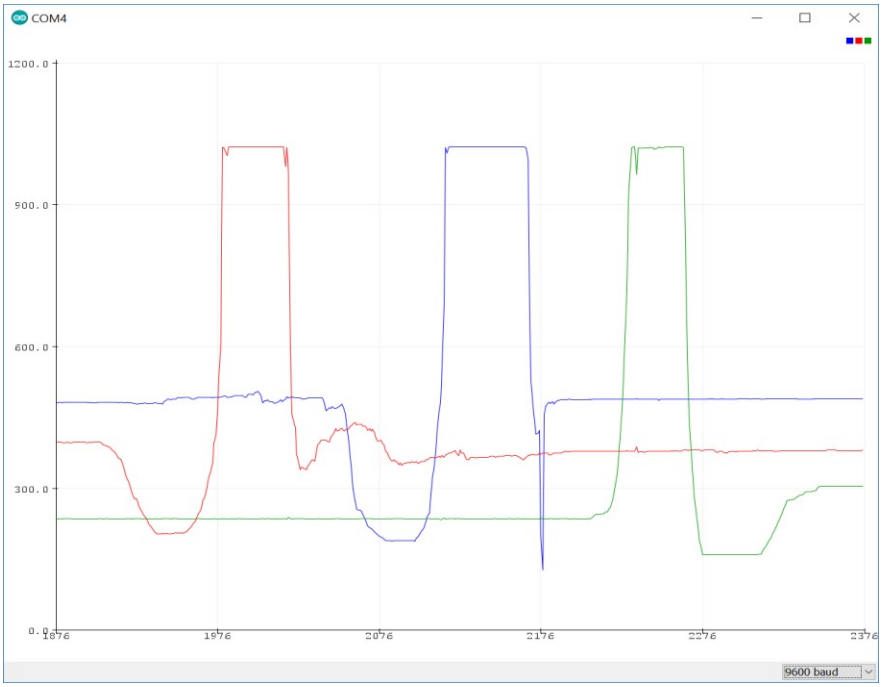


Platform Construction. Added top deck, aircraft model, and initial phone mount (later replaced).

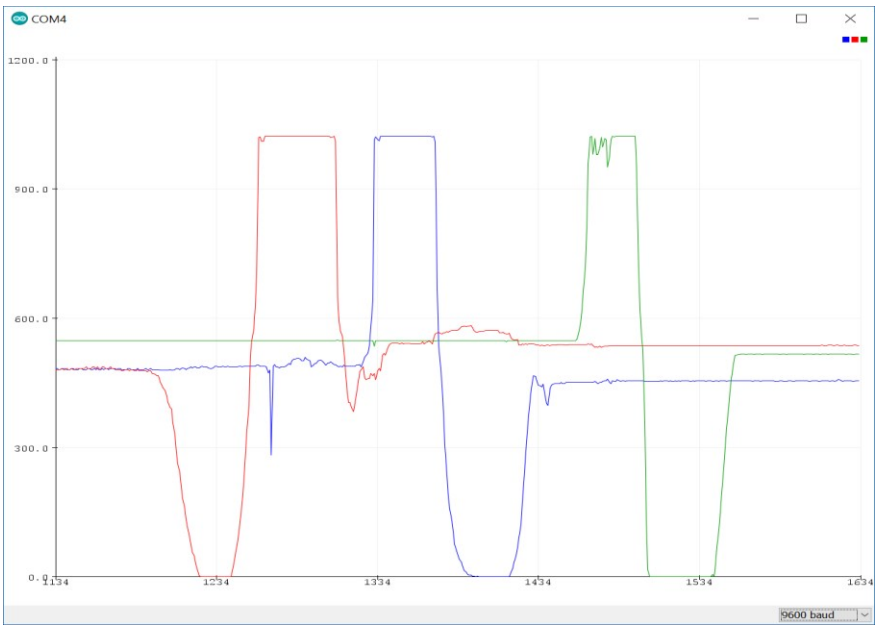
Gimbal Construction. Pitch servo embedded in right side support post (red circle). Horizontal box frame and platform pitch together; platform rolls within frame.

Test flight of Completed Product. White flight yoke sends pitch and roll commands to Adafruit Metro (Arduino Uno equivalent) mounted to right side of frame. Metro drives servos, moving platform. Black game controller controls throttle, flaps, landing gear through Bluetooth, direct to Android. Full video available on YouTube.

JOYSTICK CALIBRATION



Original Joystick Values. Serial Plotter readout of unaltered pitch (red), roll (blue), and throttle (green) potentiometer readings. Everything goes up to 1023, but nothing comes down to zero, and nothing has a midpoint of 512.



Remapped Joystick Values. Serial Plotter readout after re-mapping upper a and lower "halves" of each joystick axis. Midpoints are closer to 512, minimums closer to zero, and range of values on either side of 512 is about the same. Also note occasional "spikes" of noise.

VIDEOS

Paper Prototype https://www.youtube.com/watch?v=EByNaTV4gio

Bluetooth Test https://www.youtube.com/watch?v=Wycs4KfiRxY

Single Servo Test https://www.youtube.com/watch?v=BoMxHcFf9q4

Full flight simulator, no Bluetooth https://www.youtube.com/watch?v=s8qhwEc9Rwo

Full Flight simulator, with Bluetooth https://www.youtube.com/watch?v=M3D7SJ-Qdug

Bluetooth Configuration https://www.youtube.com/watch?v=6N5Xnvj19ag

Links

RESEARCH

Alternate mobile flight simulator, not as many free aircraft; no easily accessible Bluetooth interface. https://www.x-plane.com/mobile

Bluetooth BLE commands https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/ble-services

Bluetooth/Bluefruit factory reset https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-uart-friend/factory-reset

Bluetooth/Bluefruit HID Mousemove sketch https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/ble-services#at-plus-blehidmousemove-14-35

SOURCES

Mobile flight simulator used in this project. https://infiniteflight.com/

Thermometer Arduino sketch, the basis of JoystickTest.ino
https://blogs.uw.edu/fizzlab/technology/examples

arduino servo motor avec potonsiométer (Arduino servo motor with potentiometer by user Ounis.Brahim https://www.tinkercad.com/things/hrFMb9KYSr1-arduino-servo-motor-avec-potonsiometre- accessed 11/17/18

PC game port Joystick pin out
https://allpinouts.org/pinouts/connectors/input_device/joystick-pc-gameport

Conecting PC game port joystick to Arduino http://build-its.blogspot.com/2012/01/arduino-game-port-interface.html

WORK PRODUCTS

Azure DevOps project:
https://dev.azure.com/HCDE539leos/Arduino%20Model%20Flight%20Simulator

Project Wiki
https://dev.azure.com/HCDE539leos/Arduino%20Model%20Flight%20Simulator/_wiki/wikis/HCDE-Final-Project.wiki?wikiVersion=GBwikiMaster

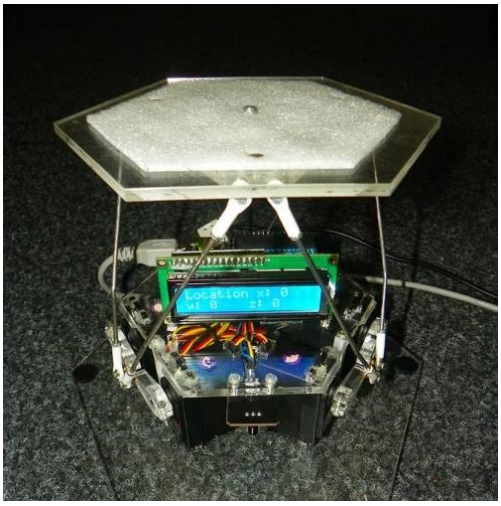
TinkerCAD circuit: https://www.tinkercad.com/things/eyIbdlANTBl-2dof-motion-platform

Process

From the course syllabus point of view, HCDE 539 struck a good balance of providing just-enough in-class instruction and homework assignments to get students familiar with Arduino-based physical computing, while allowing enough homework-free weeks to focus on projects. Framing a couple class sessions as studio time was an effective way to ensure everyone had access to instructors and TA's, regardless of their daytime commitments.

As for this project itself, it can only be claimed as a success because of multiple checkpoints and scale-backs along the way. The very first motion platform concept was based on a Steward Platform, a configuration used on actual full-motion flight simulators, as well as a few Arduino projects.

http://www.newworldencyclopedia.org/entry/Flight_simulator



https://www.instructables.com/id/Arduino-controlled-Rotary-Stewart-Platform/

Although this approach could have been close to a "scale model", it amounted to an all-or-nothing approach with no way to declare partial success. In contrast, the "one degree of freedom at a time" approach afforded numerous off-ramps in case technical or schedule risk rose above acceptable levels. The paper prototype technique of HCDE 518 helped identify a 3rd party flight simulator, an aircraft model, and an acceptable range of platform motion – all without writing a single line of code or placing a single element on a breadboard. The single-servo prototype could have been a minimal project itself, since it already incorporated input and output devices, and a flying experience that could have been framed as a simple takeoff and landing simulator. The simulator really came into its own with the addition of a second degree of freedom, allowing pitch and roll. This is where

joystick noise and calibration became a major issue, but there was enough schedule left to implement some mitigation code.

Bluetooth throttle, landing gear, and flaps control became a stretch goal. Integration of the joystick throttle slider with Arduino, Bluefruit, and the flight simulator itself would have afforded a full realization of all the input channels available on the original fight yoke, but schedule didn't allow. The Bluetooth game controller served as a reasonable substitute.

Another Process technique from this project was the first-time use of Azure DevOps for to manage code, task lists, and documentation. Although overkill for a one-person six-week project, using it for projects of this scale makes for a good training exercise in preparation of using it for enterprise-level projects. It was every convenient to have Trello-style task boards, wiki pages, and code repos all in one place. Github and Gitlab does this as well, though Azure DevOps is more feature-rich.

Looking to the future, I will probably keep this project in Azure DevOps, but sand up a mirror repo in GitHub where the rest of my portfolio code lives. It was great to give an obsolete flight yoke new life. At some point, I'd love to return to the Bluefruit work, to bring more sliders and switches to life.