Motorized Throttle Quadrant for Flight Simulation

By

Team 45 — Wendi Fu, Ziang Guo, Yuqi Xue

{wendifu2, ziangg2, yuqixue2}@illinois.edu

ECE 445 Design Document

TA: Chaitanya Sindagi

March 5, 2021

Glossary

Auto Pilot (A/P) is the main flight automation system governing the attitude of the aircraft consisting of multiple automation subsystems (including A/T).

Auto Throttle (A/T) Part of the flight automation system that takes full control of the thrust demand of the aircraft once engaged. The pilot only need to specify a desired performance figure and the A/T system will automatically adjust engine outputs to achieve said performance.

Control Column is the device for controlling aircraft's pitch and roll placed in front of the pilots on most aircraft models.

Flight Simulator is a hardware device, a software program, or a combination of both that artificially re-creates aircraft flight and the environment in which it flies, for pilot training, design, or other purposes.

FMC/CDU short for Flight Management Computer/Central Display Unit, is the input device for flight information including route, aircraft performance figures, and performance limits. Must be correctly configured for flight automation to function.

Main Control Panel (MCP) is the control panel allowing the pilots to select operating modes of the flight automation system and command automated attitude changes.

Master Caution is the secondary warning for critical non-emergency events such as A/T disengage.

Multi-Function Display is the inboard displays capable of displaying multiple system status including system recall and engine status.

Primary Flight Display (PFD) is the primary flight instrument in a digitized cockpit, displaying crucial information including flight automation status and aircraft attitude.

Recall is the log-keeping functionality of the FMC capable of summarizing events occurring with the aircraft for improved crew awareness.

Speedbrake is mechanised devices on the top-surface of the wings that induces extra drag when deployed for deceleration. Deployment of the speedbrake is controlled either manually with a lever on the throttle quadrant or automatically when the aircraft lands.

Throttle Quadrant refers to the unit housing all essential thrust control inputs (buttons and levers). Often placed in between two pilots in the center pedestal for ease of access.

Thrust Lever is a lever-like input device with a defined travel range. The position of the thrust lever is mapped to engine thrust demands.

TO/GA (Take Off/Go Around) is the maximum rated thrust the engines can generate.

1 Introduction

1.1 Objective

Modern airliners are equipped with multiple flight automation systems to alleviate the workload of pilots during long flights. One of these automations commonly fitted is the Auto Throttle (A/T) system. A/T is able to fully control the engine thrust demand of the airplane in order to achieve constant airspeed cruise and provide crucial safety functionalities. To improve the situational awareness of the pilots, Boeing, alongside many major airplane manufacturers, fitted their airplanes with motorized throttle quadrants. When A/T sends thrust settings to the airplane's engines, the thrust levers will move accordingly to the positions that reflect the real-time engine thrust outputs. Such system allows the pilots to regain control of the throttle at any moment with full awareness of the current thrust settings. This is a crucial control characteristic of these airplanes. Without this consistency between thrust lever positions and actual engine thrust outputs, when A/T is disengaged, the airplane's engines will immediately try to adjust to the current thrust lever positions, which may cause a sudden and unexpected power increase or decrease of the engines.

Current main-stream consumer-level flight simulator peripherals use spring tension to simulate the weight felt by the pilot when operating the control column. However, no effort is made to simulate the synchronous movement of the throttle quadrant with A/T commands. Home-based simulation systems often have the pilots to manually adjust the thrust levers to match the actual thrust setting inside the simulator software program, hence deteriorating the experience. Being able to recreate the motorized throttle will bring the realism of the flight simulation experience to a whole new level.

With this project we aim to design and create a motorized throttle quadrant for casual flight enthusiasts. The throttle quadrant can interface with mainstream simulation software (Microsoft FSX [1] or Lockheed Martin Prepar3D [2]) via universal protocol (e.g., USB 3.0) and synchronize thrust levers' positions with A/T commands. Figure 1 is an illustration of such process in action which vaguely represents the final product's main functionality.

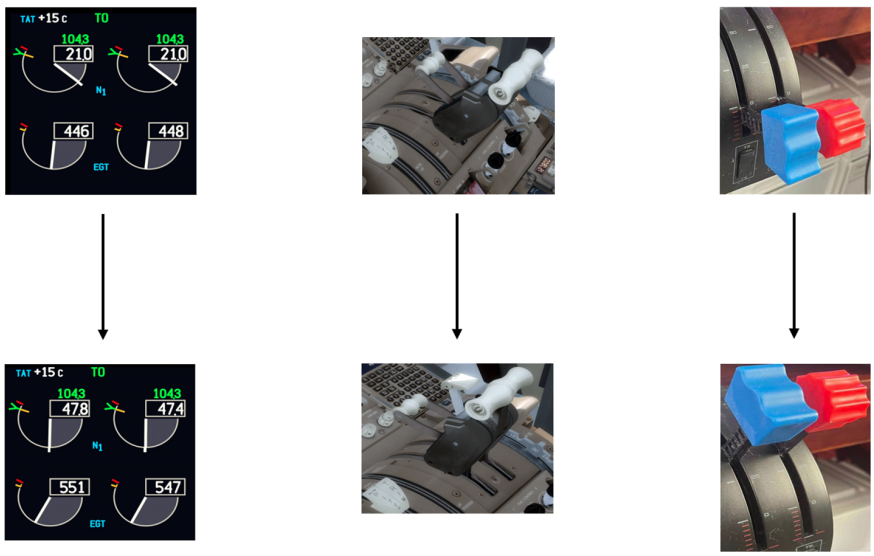

Figure 1: Illustration of device operation. Multi-function display (MFD) of the aircraft (left). Real looking of the throttle quadrant (middle). Flight simulator throttle quadrant (right). The MFD, real throttle quadrant, and simulator throttle quadrant on the same row represent the same the engine thrust setting.

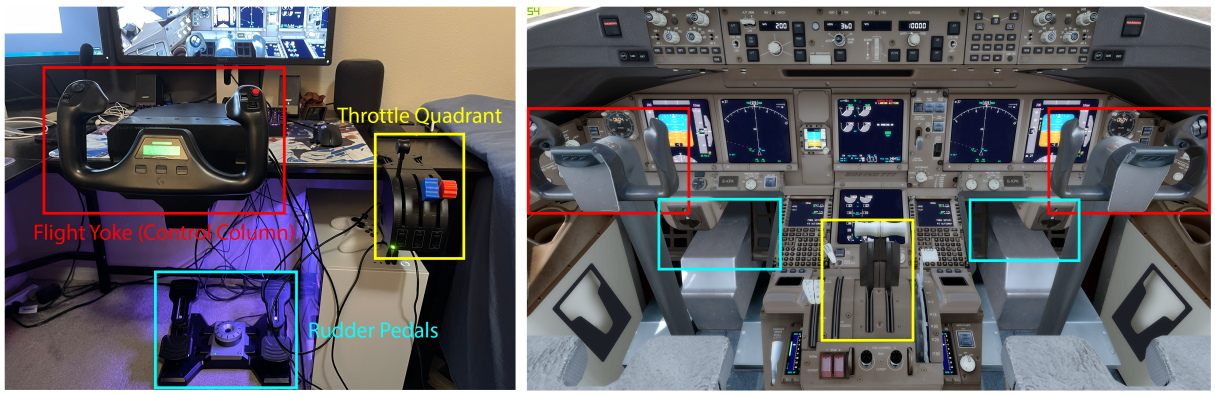

Figure 2: Comparison of a casual in-home flight simulator (left) and the real cockpit view in Prepar3Dv4 (right). Corresponding components are labelled by the same color (for example, red boxes mark the control columns in both pictures.

1.2 Background

Despite commercial full-scale flight deck simulators for professional pilot training purpose, a common choice for casual flight enthusiasts is Logitech Flight Yoke and Throttle Quadrant [3, 4], often purchased together with a Rudder Pedal System, as shown in Figure 2. While Logitech's solution is affordable and popular, it does not have a motorized throttle quadrant, thus deteriorating the realism of flight simulation.

In the domain of driving simulators, wheel and pedal systems such as Logitech G920 [5] already support force feedback, and existing solutions are affordable to casual players. These simulators use a motorized wheel to simulate the actual behavior of a car's wheel. Our project aims to build motorized throttle levers for flight simulators.

Flight Illusion has motorized flight yokes on sale [6], but their solutions cost more than $1200 and are too expensive for casual players. No ready-to-use motorized throttle quadrant under $1000 is available on the market as far as we have discovered.

1.3 High-Level Requirements

To better evaluate the outcome of the project, three major requirements must be met:

- The throttle quadrant must be recognized by both the OS (i.e., be listed in Device Manager in Windows 10) and flight simulator software (i.e., be listed in the corresponding settings menu) as a valid input source. We will be targeting Windows 10 and Prepar3Dv4. Supporting more OSes and simulators is possible but out of the scope of this work. The flight simulator must also be able to calibrate the throttle quadrant input according to the documentation provided by the flight simulator software [2].
- If A/T is engaged and commands a thrust change, the throttle quadrant must reflect the movement (speed and direction of position change) of the thrust levers shown in the simulator's virtual cockpit with a latency less than 100ms, which is an unnoticeable delay comparable to that of a wireless game console controller [7]. The thrust levers must stop within 2% of the commanded position. We assume that the simulator software is reliable for sending appropriate A/T commands so that the speed of the levers under A/T can be realized by the motors. The pilot is allowed to move the levers at a faster speed than A/T does.

- If the pilot overrides A/T thrust settings by moving the thrust levers while A/T is engaged, the throttle quadrant must set free the thrust levers in less than 50ms to hand over the control of the levers to the pilot. The extra force needed to initiate the override must be less than 1 N. The movement of the thrust levers from then on must be unhindered until A/T is engaged again.

2 Design

Figure 3 is an overall view of our design. Our design is divided into two parts that will work in tandem to achieve the requirements listed in Section 1.3. The software part includes an add-in for selected flight simulation software and a USB driver. The add-in will gather airplane status information from APIs provided by the flight simulation software. The USB driver will allow bidirectional communication between the A/T controller inside the flight simulator and the throttle quadrant we build. The hardware part includes a custom control board carrying a microcontroller and a stepper-motor driver chip. The control board will translate between thrust lever position and throttle input. Stepper motors with feedback enabled will provide accurate movement of the thrust levers and slip detection for override detection. In addition to software and hardware components, we will also include a specification of the host/device communication protocol we use.

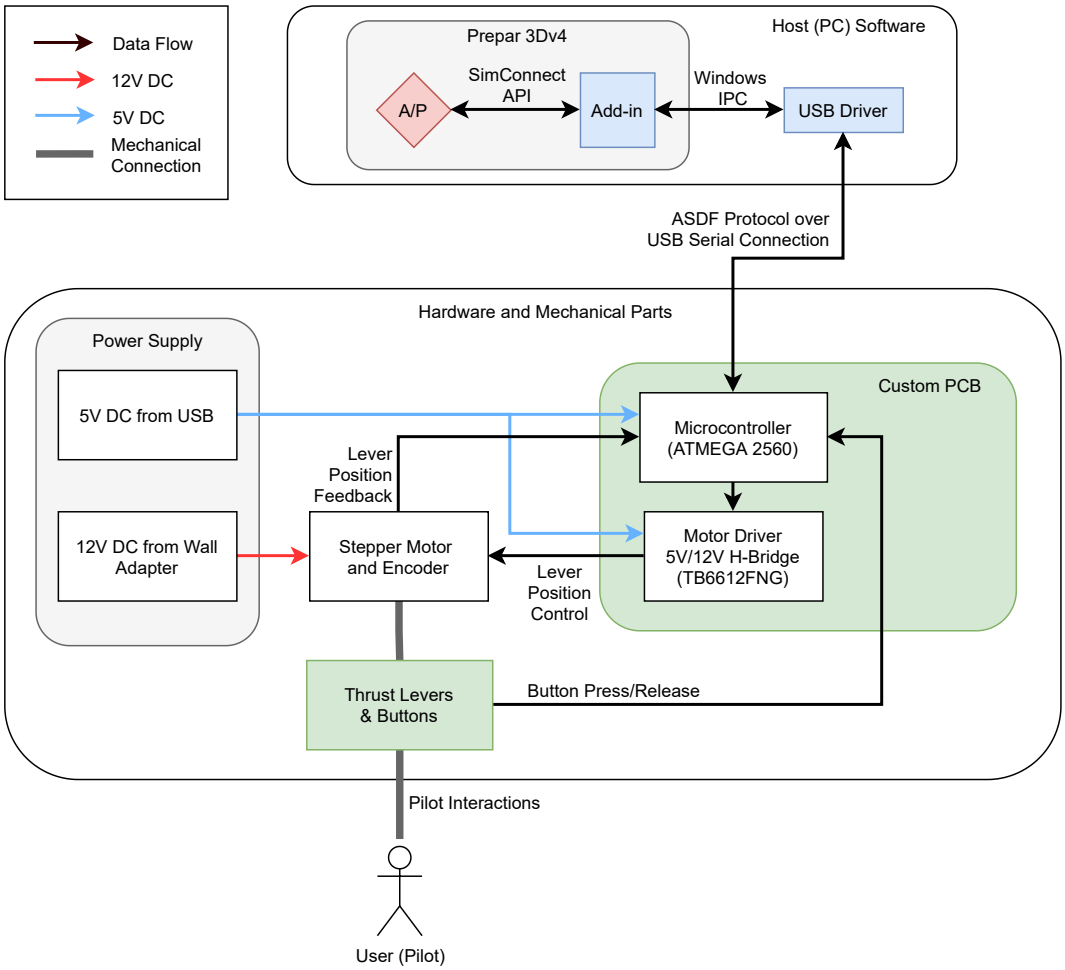

Figure 3: Block diagram of the design.

2.1 Host Software

The software subsystem running on the host computer consists of the simulation software's Auto Pilot functionalities, an add-in for data gathering and virtual inputs, and a USB driver for device registration and communication over serial protocol on the USB bus. The software subsystem must be capable of constant polling from both the simulation software and the throttle peripheral to ensure continuous and immediate response once any change is made. The R&Vs of the host software are listed in Table 1.

2.1.1 Auto/Pilot

The A/P system is provided by the flight simulation software with well-documented APIs allowing access to aircraft status data and control inputs. This block will be the software endpoint of our system.

2.1.2 Flight Simulator Add-in

The add-in block will be written to be compatible with select flight simulation software. It will be responsible for gathering flight simulation data from the A/P system API as well as transmitting pilot input into the flight simulation software for interpretation. It must be able to achieve continuous, real-time bidirectional communication.

2.1.3 Windows USB Driver

The USB driver will allow the Windows 10 operating system to recognize our hardware as a valid peripheral connected to the USB bus. All data and signal traffic to and from the software add-in will be handled by the USB driver. Once installed on the host PC, the USB driver should enable PnP for the throttle peripheral.

Requirement	Verification
The FS Add-in can read flight data from the simulator software via its API calls.	1. Start the simulator program with the add-in. 2. The debug terminal of the add-in should appear automatically or be opened manually. 3. Load a scene in the simulator program, bring the aircraft into sky, manually adjust thrust demand for both engines and extend the speedbrake. 4. Verify that the debug terminal of the add-in prints correct data that agree with the flight instruments in the virtual cockpit.

The FS Add-in can write data via the simulator API to control the airplane with $<$ 10ms delay.	1. Start the simulator program with the add-in and the debug terminal. 2. Load a scene in the simulator program. 3. Send thrust demand change commands via the debug terminal and start the timer. 4. Confirm requested flight control changes corrects takes effect and end the timer. 5. The time between command sent and flight control response should be less than 10 ms.
The FS Add-in registers the throttle quadrant as a valid controller in the simulator software so that it can be calibrated and used as a controller input source.	1. Connect the throttle quadrant to the host computer. 2. Start the simulator program with the add-in. 3. Go to Input Calibration in Settings menu in the simulator program. 4. The throttle quadrant must be listed as a input source and all input channels (3 levers and 2 buttons) are ready to be assigned to simulator functionalities. 5. Start input calibration process of the simulator program, and verify that the simulator program correctly adjust control mappings to the input range of the device.
The USB Driver registers the throttle quadrant as a valid USB controller in the Windows OS and the OS can identify the new hardware once the throttle quadrant is plugged into the host via USB 3.0.	1. When the Windows OS operates normally, start the USB Driver and plug in the throttle quadrant to an operating USB 3.0 port. 2. Observe the "new device" notification from the Windows OS, and check that the throttle quadrant is recognized in the Device Manager.

The FS Add-in can read (write) data from (to) the throttle quadrant microcontroller via USB Serial Protocol with < 100ms delay in each direction.	Note: This test procedure assumes that the throttle quadrant microcontroller correctly handles USB Serial Communication with the host. 1. Start the FS Add-in debug terminal. 2. Connect the throttle quadrant to an operating USB 3.0 port. 3. The data read from the throttle quadrant will be automatically printed onto the debug terminal every 250 ms. 4. Open the ATMega serial debug terminal and connect it to the throttle quadrant. 5. Send thrust demand change commands via the debug terminal to the throttle quadrant and start a timer 6. Verify that correct information is printed in the ATMega debug terminal. 7. Observe the thrust levers move automatically to the demanded thrust position and end the timer 8. The delay between command issuance and thrust set should be no more than 100 ms.

Table 1: R&Vs for Host Software.

2.2 Electrical & Electronic Parts

The hardware subsystem consists of a microcontroller, a stepper motor driver unit, three stepper motors, and the user interfaces (levers and buttons). The hardware subsystem should provide the pilot with a firm but effortless operational feel as close to that of real airplanes as possible. The levers must be able to move at a constant rate smoothly whenever A/T commands a thrust adjustment. The levers must stop exactly and firmly at the designated position without slipping or drifting. The motors should never impose opposing torque to pilot inputs and must report any override to the software subsystem. The R&Vs of the EE hardware are listed in Table 2.

2.2.1 Microcontroller

The microcontroller will be the processing center on the peripheral side. Connected to the host PC via USB 3.0 and powered by USB, the microcontroller will interface with the host USB driver to deliver and collect necessary data. The microcontroller will also be responsible for instructing movements of the motors, responding to button presses, and decoding thrust lever positions. This will likely to be an ATMega microcontroller running custom codes.

2.2.2 Motor Driver

The motor driver manages power and control signals for the stepper motors. Special stepper motor drivers are necessary due to the special construction of stepper motors and the unique requirements to run them. This will likely to be an aftermarket component.

2.2.3 Stepper Motors

Three stepper motors will be used to physically move the levers. Each stepper motor will be responsible for only one lever. The stepper motors will be powered externally by a power adapter and controlled by the stepper motor driver. They will remain unpowered unless an instruction was received to move. The stepper motors will have closed feedback loops for accurate determination of the motors' positions.

2.2.4 Power Supply

The power supply will be the source of energy for the 3 stepper motors. The power supply must be able to rectify, convert, and filter the input voltage to 12 V DC required by the stepper motors to function. The power supply should output 12±0.5 V DC at 2 A under any load condition without active cooling.

Requirement	Verification
The microcontroller firmware can decode and encode data in the correct format.	1. Start the simulator program with the add-in and the debug terminal. 2. Connect and set up the throttle quadrant with ATMega serial debug terminal monitoring. 3. Send thrust demand change and speedbrake deployment commands to the throttle quadrant. 4. Verify the successful decoding of the test pattern in the serial debug terminal. 5. Manually apply test signals to the microcontroller via debug ports. (bypassing hardware encoder/motor feedback) 6. Verify the reception of test signals in the Add-in debug terminal.
The motor driver can advance the stepper motor without slipping and stop within 1.8° of assigned position.	1. Power up the throttle quadrant with 12 V DC. 2. Manually send thrust lever position change commands to the motor driver circuit (bypassing the microcontroller). 3. Verify all 3 steppers can start without slipping and stop within 1 step (1.8°) of assigned location after continuous (more than 2 steps) motion.
The stepper motors can move all levers smoothly. Individual levers can move without affecting its neighbors. The motors sustain long operations without reach above 60°C.	1. Fully power the throttle quadrant (5 V rail and 12 V rail). 2. Open ATMega debug terminal and send stress test command (cycle all motors at full power continously). 3. Verify the movements of the levers are steady at 12.6° /s turn rate (7 seconds to complete idle-TO/GA travel). 4. Leave test running for at least one hour. 5. Confirm with infrared camera that the stepper motor packages are below 60°C.

The microcontroller can accurately respond to virtual thrust adjustments and adjust the thrust levers accordingly with delay of < 100ms.	Note: This test procedure assumes that the software host is functional. 1. Start the simulator program. 2. Connect the throttle quadrant to an operating USB 3.0 port. 3. Load compatible scenario. 4. Verify the levers are moved to standby positions when the aircraft system is initialized. 5. Engage A/T inside the simulation software and command a thrust change via the MCP and start a timer. 6. Verify the movement of the thrust levers follows that of the thrust levers in the virtual cockpit and stop the timer 7. The delay should be no more than 100 ms. 8. Extend the speedbrake inside the virtual cockpit and start a new timer. 9. Verify the speedbrake lever moves to match its position in the virtual cockpit and stop the timer. 10. The delay should be no more than 100 ms.
The resisting torque from the steppers when the lever is stationary prevents thrust changes from ambient interference but does not require more than 1 N applied to be overriden.	Note: This test procedure assumes that the software host is functional and a compatible scenario is loaded. 1. Measure with newton meter the force required to move the thrust lever at least one step. 2. Engage A/T inside the virtual cockpit. 3. Lightly touch the thrust levers. 4. Confirm the levers are not moved and no thrust change is registered. 5. Attempt to operate the levers like in normal manual flight. 6. Confirm the levers can travel within the 90° movement range with the same 2% report accuracy. 7. Measure with newton meter the force required to move the thrust lever at least one step. 8. The extra force required should be no more than 1 N.

The power supply is capable of motorizing all steppers at 12V 0.5A each constantly without reaching $>$ 60°C on the unit package.	1. Power up the throttle quadrant 12 V rail with the power supply. 2. Send power stress test command via the ATMega debug terminal (run all motors at 12 V 0.5 A continuously). 3. Confirm with an oscilloscope that the output of the power supply is at 12$\pm$ 0.5 V DC. 4. Keep the test sequence running for 1 hour 5. Confirm with infrared camera the temperature of the exterior of the power supply unit is $<$ 60°C

Table 2: R&Vs for Electronic Hardware.

2.3 Mechanical Parts

The R&Vs of the mechanical hardware are listed in Table 3.

2.3.1 Levers and Buttons

The thrusts levers and buttons are the main user interface. They will be fixed on one end with a hard-limited travel range. Rotation of the thrust levers about the fixed end will correspond to adjustments to the thrust setting. The travel range and thrust mapping will be as close to a Boeing 737-800 as possible for realism. Buttons will be attached to the top of both thrust levers and one on the left thrust lever. These button presses will be handled by the microcontroller and be mapped to corresponding functionalities found on the Boeing 737-800. Figure 4 is an illustration of the physical look of our product.

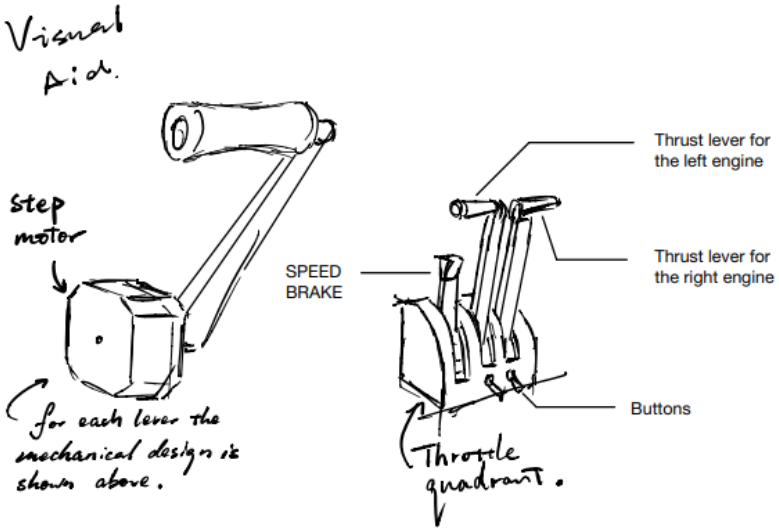

Figure 4: Visual aid for physical design.

Requirement	Verification
The levers can be mapped to corresponding functionalities in the simulation software.	Note: This test procedure assumes that the software host and the EE hardware are functional. 1. Open the simulator settings and select throttle/speedbrake lever mappings 2. Map the physical levers to corresponding controls and initiate calibration. 3. Confirm the levers can be mapped to full control range (determined automatically by the simulation program) with at least 1:1 resolution.
The TO/GA button and both A/T disengage buttons are functional with $<$ 100ms input latency.	Note: This test procedure assumes that the software host and the EE hardware are functional. 1. Start the simulator program. 2. Load a compatible scenario. 3. Setup FMC/CDU for takeoff. 4. Press the TO/GA button and start a timer. 5. Confirm on the PFD that A/T mode has changed to TO/GA and stop the timer. 6. The delay should be no more than 100 ms. 7. Press A/T disengage button and start a new timer. 8. The A/T ARM switch on the MCP flicks to off position. The Master Caution is lit up. A/T DISENGAGE is shown on the MFD recall section and stop the timer. 9. Verify that the delay is no more than 100 ms.

Table 3: R&Vs for Mechanical Parts.

2.4 Host/Device Communication Protocol

The host program and the device microcontroller will communicate over a USB Serial connection, which provides a streaming interface for bidirectional reads and writes. We use existing USB Serial libraries for the Windows OS and the ATMega chip to implement our communication protocol.

We present the packet-based communication protocol, the Aviation Simulator Data Format (ASDF) Protocol, for the host to poll data from the thrust quadrant and send commands to set thrust position. The packet sent by the host consists of a command opcode and optionally several bytes of data. The opcode is 1 byte with the most significant bit set to 1. Each data byte will have the most significant bit set to 0 to be distinguished from opcode bytes. This scheme limits the number of opcodes and the maximum value of one data byte to $2^7 = 128$, which is more than enough for our purposes. The details of each command are specified in Table 4.

Command	Opcode	Function
CMD_RESET	0x80	Reset the throttle quadrant device.
CMD_POLL	0x81	Get current status of the thrust levers and the buttons.
CMD_LVR_SET	0x{8-F}2	Lock and set lever positions. This command is actually a combination of 8 commands. Bit 6 to bit 4 form a bitmask indicating which thrust levers are to be set. The opcode is then followed by the position values in the order of the bitmask. If none of the bits are set, this command does nothing, and no following value bytes are required.
CMD_LVR_RELS	0x83	Release the thrust lever (and might let the flight simulator disengage A/P).
CMD_ASDF	0xFF	Reserved for debug purposes.

Table 4: Host ommands for the ASDF Protocol.

The response packet sent by the thrust quadrant microcontroller contains one or more bytes depending on the command received from the host. The response formats are listed in Table 5.

Response	Format	Function
ASDF_ERROR	0xFF	Response when the device encounters an error.
ASDF_ACK	0x00	A generic response when the device successfully completes a command.
ASDF_RESET	0x01	Generated when the device finishes initialization, which might be the result of a CMD_REST command or the normal bootup.
ASDF_POLL_OK	0x02	First byte of the response to a CMD_POLL command. This byte will be followed by the data bytes asked by the command.
ASDF_LVR_RELS	0x{0,8}3	Generated when the thrust lever is released to pilot's control. If bit 7 is set, then this response is the result of a CMD_THR_RELS command. Otherwise, this indicates that the pilot wants to forcibly disengage A/P by moving the thrust lever.

Table 5: Device responses for the ASDF Protocol.

2.5 Tolerance Analysis

The most essential component in our design would be the stepper motors that moves the thrust levers. We have identified three important parameters that impact the final product the most and performed analytical calculations to establish performance targets.

- When the TO/GA button is pressed on a Boeing 777-300ER while both engines are at idle, it will take exactly 7 seconds for the thrust levers to advance to the TO/GA position. Our intended stepper motor model, the 17HS4401S, is capable of 1.8° steps. To closely simulate the motion of the thrust levers on a real plane, our motor must have a maximum turn rate of at least

$$\frac{90^\circ \text{ / } 7 \text{ second}}{1.8^\circ \text{ / step}} \approx 7 \text{ step/second.}$$

  The target turn rate will also affect the maximum running torque of the motor during operation.

- For simulation accuracy as well as user experiences, the thrust levers must be able to react to controller signals with as little delay as possible. It is also crucial that the thrust levers are capable of remaining stationary under minor external influences such as table vibration or device movements. The 17HS4401S offers 2.2 N· cm maximum detent torque and 40 N·cm minimum holding torque. We have the choice of either leaving the motor wings de-energized until motion request or have them energized all the time and de-energize when the pilot overrides. Having constantly energized wings will maximize the "locking" effect of the thrust levers and filter any unintended inputs. The response delay of the motors will also be reduced. However, the pilot will need to exert extra force to move the lever by at least one step to have the override be detected. Assuming a thrust lever of length 14 cm (measured from Logitech Saitek throttle quadrant), the extra force required is

$$F_{\text{extra}} = \frac{40 \text{ N} \cdot \text{cm}}{14 \text{ cm}} \approx 2.86 \text{ N.}$$

  Such amount of force will be noticeable to the pilot.

  De-energizeing the wings until movements are requested will provide lower locking force holding the thrust lever position and the pilot can move the levers with little extra force. But the motor will take longer to reach desired angular velocity once requested to move and external noises (vibrations, small nudges, etc) cannot be filtered as effectively.

- In complicated flight scenarios (crosswind approach, wind shear, damaged airplane, etc.), the A/T will make minute adjustments at very high rates. We expect all thrust lever motors to be operating at full torque output to keep up with the rapid thrust changes. With a rated voltage of 12 V and a maximum current of 1.7 A, the maximum power of the motor one motor is

$$P_{\text{max}} = 1.7 \text{ A} \times 12 \text{ V} = 20.4 \text{ W.}$$

  Then the two thrust motors combined will consume more than 40 watts of heat. For the depicted worse-case-scenario, most of the power will be dissipated as heat when the motors need to quickly change direction. Apparently, removing 40 watts of heat passively from the small throttle quadrant package will be challenging. The two possible solutions are reducing current at the cost of lower running torque or including active cooling systems which will add to the complexity and cost of the project.

After close analysis and considerations over the three identified problems, we have made the final design choice that will minimize the compromise made to user experience while keeping the design simple and the

cost reasonable. Our final design choice is a current limit enforced by the stepper motor driver circuit. The ATmega-2560 microcontroller will contain firmware capable of detecting abnormal torque surges and cutoff motor power should a stall is detected. A resistor-based current limiter will be the fail-safe backup should the microcontroller fail. The limit is currently discussed to be 0.5 A. With the new limit in place, the new holding torque would be

$$T_{\text{new}} = 40 \text{ N} \cdot \text{cm} \times \frac{0.5 \text{ A}}{1.7 \text{ A}} = 11.8 \text{ N} \cdot \text{cm}.$$

With a 17 cm thrust lever (accurate to Boeing 777 specification), the required user input force to override the stepper motor is

$$F_{\text{extra}}^{'} = \frac{11.8 \text{ N} \cdot \text{cm}}{17 \text{ cm}} = 0.69 \text{ N}.$$

This significantly reduces the chance that the pilot will need to exert noticeably high (>1 N) of force to perform emergency overrides. The reduced current will also lower the thermal output of the steppers. The new anticipated thermal capacity is

$$P_{\text{max}}^{'} = 0.5 \text{ A} \times 12 \text{ V} = 6 \text{ W}.$$

The two thrust lever steppers combined will have a maximum thermal output of 12 watts. Such thermal output can be easily managed with externally-mounted heat-sinks such as the DA-T268-301E-TR with 7W of thermal dissipation at 76°C. Last but not least, we confirm that our new current limit can still satisfy the minimum RPM requirement:

$$t_{\text{step, min}} = \frac{2LI_{\text{max}}}{V} = \frac{2 \times 2.8 \text{ mH} \times 0.5 \text{ A}}{12 \text{ V}} = 0.23 \text{ ms}.$$

This is lower than the required minimum of 143 ms per step and hence will still satisfy the required RPM for accurate simulation.

3 Costs

We estimate the costs for each component of our prototype and bulk product in Table 6.

The total labor cost for this project is estimated to be 100 hours for each of us, we also estimate that the ECE machine shop will take 6 hours to complete our design. Considering that the average hourly income for Illinois Electrical Engineers is $30 to $40, we take the average and assume the total labor cost is

$$[(100 \times 3 + 6) \text{ hours}] \times [\$35 \text{ per hour}] \times 2.5 = \$26775.$$

Our intended competitor product is the Logitech G Pro Flight Throttle Quadrant with 557 customer ratings on Amazon US [8]. Assume a typical 15% sales-to-feedback ratio [9], our anticipated sale is approximately 4000 units. With the assembly process optimized, the time estimated for an experienced worker to assemble a unit from bare components is 2 hours, including basic quality control and testing. Then the total retail unit cost is

$$\$35/\text{hour} \times 2 \text{ hours} + \$33.2 = \$103.2.$$

Splitting the development costs to 4000 units and apply a revenue of 5%, the final retail price before tax would be

$$\left(\frac{\$26775 + \$94.06}{4000} + \$103.2\right) \times 105\% = \$115.41.$$

This is slightly higher than the non-motorized Logitech G at $87 before tax. However, non-motorized offers from Thrustmaster is more expensive than our product at $152.44 before tax.

Part	Description	Qty	Cost (Protptype)	Cost (Bulk)
17HS4401S	Stepper Motor	3	$29.91	$14.97
ATmega-2560	Microcontroller	1	$3.82	$3.82
TB6612FNG	5V/12V H-bridge	3	$14.85	$1.65
485-1503	Switch Button	2	$1.90	$1.90
292303-1	USB Female Connector	1	$1.11	$1.11
JX-12069	12V DC 2A Power Supply	1	$7.95	$3.65
B002KKZRYM	USB A to A Cable	1	$4.52	$1.00
PCB	Cost of customized PCB	1	$10.0	$0.10
Mechanical Parts	Cost of Levers and the Outer Shell	1	$20.0	$5.0
Total			**$94.06**	**$33.2**

Table 6: Component Costs Per Unit.

4 Schedule

Our schedule is listed in Table 7.

Week	Wendi	Yuqi	Ziang
02/22/21	Research on motor control; communicate with machine shop	Research Prepar3Dv4 SDK and Windows 10 USB Serial API	Familiarize with PCB design workflow and layout in AutoCAD Eagle with Wendi
03/01/21	Research on customized PCB design and microcontroller; validate and finalize mechanical design	Design Host/Device communication protocol; begin host software development (Prepar3Dv4 Add-in and Windows OS driver)	Complete first PCB schematic and layout draft; Source testing equipment locally for asynchronous testing and validation
03/08/21	Complete mechanical parts with machine shop; finish purchase all required parts; draw the PCB for the microcontroller	Continue host software development, including debug terminal command line interface (CLI) and automatic software test suite	Source components locally and perform conceptual test on bread boards; Contact local PCB prototyping service and print first PCB design
03/15/21	Test the effectiveness of the mechanical design; finalizing mechanical design; begin implementing control unit	Finalize development of debug terminal CLI	Perform validation of first PCB version; Report any problem to Wendi and discuss possible fixes
03/22/21	Finalize control logic and control unit; test the PCB and make necessary improvements	Finalize Add-in and in-simulator A/P communication as well as corresponding software test suite	Validate improved PCB design; Contact local PCB prototyping service for second revision
03/29/21	Implement and finalize the hardware and test; integrate the hardware and software	Finalize Windows OS driver, including device registration in Device Manager and and in simulator program as well as corresponding software test suite	Improve PCB design based on Wendi's test results for third PCB revision; Contact local PCB prototyping service for third revision; Validate third PCB revision
04/05/21	Test and validate requirements with real device	Test and profile host software with real device	Finalize PCB design in time for last PCBway order
04/12/21	Performance analysis and optimization of microcontroller and device effectiveness; finalize the project	Performance analysis and optimization of host software and host/device communication; finalize software stack, including both host software and microcontroller program	Perform stress tests and extended flight simulation sessions for debugging and optimization purposes; Confirm all performances meet R&V requirements
04/19/21	Reserved for schedule slip, or begin preparing final report	Reserved for schedule slip, or begin preparing final report	Reserved for schedule slip, or begin preparing final report
04/26/21	Final demo	Final demo	Final demo
05/03/21	Final presentation and final report	Final presentation and final report	Final presentation and final report

Table 7: Weekly project schedule.

5 Ethics and Safety

Our throttle quadrant aims to provide an actual and smooth experience to the audience of flight simulations, we believe that our product can bring a more authentic and affordable option to the publicity, which facilitates the wide audience of flight enthusiasts to access actual flight experience, and may serve as training equipment before they step up to planes. Thus our design aligns with the IEEE Code of Ethics Section 7.8.2: "to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems" [10].

Since our design-related extensively to mechanical and electrical parts, the safety concerns related to our physical design during our session of design, experiment and use need to be addressed. Such safety concerns could be potential safety threats towards users, also the misuse of our design may cause undesired consequences to the user and the peripheral itself. To optimally be following the IEEE Code of Ethics Section 7.8.9, striving to reduce potential harm to the user and the property, we make efforts to optimize our design [10].

Our design involves 5V/12V power supplies to drive our microcontrollers and the stepper motors to perform actions. Although the voltage does not exceed the safety voltage of 22V, power leakage may also lead to uncomfortable experience and safety threats to the users. To address the concern regarding the electric shock and power leakage, the handle of our design will be made from or covered by insulating materials to isolate the user from the electrical components as well as to prevent external items from entering the throttle quadrant and short the internal components.

The lever moving during A/T mode is controlled by the simulator program, thus may possess a possibility that the lever may cause physical damage toward the user. There is a possibility which the motor stuck the thrust lever to one end of the designated track on the frame and finally cause the motor failure. To address the safety concerns towards these mechanical issues, we will control and limit the angle of our thrust lever through a feedback loop, and the motor will disengage as soon as manual inputs involve. Also, as the Auto Throttle reflects the actual thrust level, the lever will not exceed the angle limit which we measure and set up in the microcontroller.

References

[1] "Microsoft Flight Simulator: Standard." [Online]. Available: https://www.microsoft.com/en-us/p/microsoft-flight-simulator-standard/9nxn8gf8n9ht. [Accessed: 28 Feb 2021].

[2] "Prepar3D." [Online]. Available: https://www.prepar3d.com/. [Accessed: 28 Feb 2021].

[3] "Logitech G Flight Simulator Yoke with Throttle Quadrant." [Online]. Available: https://www.logitechg.com/en-us/products/driving/driving-force-racing-wheel.html. [Accessed: 28 Feb 2021].

[4] "The Best Flight Simulator Hardware and Software for Pilots in 2021." [Online]. Available: https://hangar.flights/guides/flight-simulator-student-pilots/. [Accessed: 3 Mar 2021].

[5] "Logitech G920 & G29 Driving Force Steering Wheels & Pedals." [Online]. Available: https://www.logitechg.com/en-us/products/driving/driving-force-racing-wheel.html. [Accessed: 28 Feb 2021].

[6] "HW-FFB-CSN - Flight Illusion." [Online]. Available: https://www.flightillusion.com/product/hw-ffb-csn/. [Accessed: 28 Feb 2021].

[7] "Console Latency: Exploring Video Game Input Lag." [Online]. Available: https://displaylag.com/console-latency-exploring-video-game-input-lag/. [Accessed: 4 Mar 2021].

[8] "Logitech G Pro Flight Throttle Quadrant." [Online]. Available: https://www.amazon.com/Logitech-Pro-Flight-Throttle-Quadrant/dp/B01M00UHE3. [Accessed: 4 Mar 2021].

[9] "Amazon Feedback to Sales Ratio." [Online]. Available: https://sellercentral.amazon.com/forums/t/feedback-to-sales-ratio/279703/7. [Accessed: 4 Mar 2021].

[10] IEEE, "IEEE Code of Ethics," 2020, IEEE. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 18 Feb 2021].