Design Document

MojoDodo

Team #A08: Jason Quesenberry (Team Lead), Jason Hwu, Jeffrey Yu, Duc Nguyen

Date: 11/13/2022

Table of Contents

Revision Record

Date	**Author**	**Comments**
Oct 16, 2022	Team	Document updated for CDR
Nov 13, 2022	Team	Updated Conclusion/Integration for final

Project Description

We have designed a dodo bird under the theme “Buzz in Wonderland.” The dodo can flap its wings at different speeds, open its beak, and output squawks, all of which may occur simultaneously.

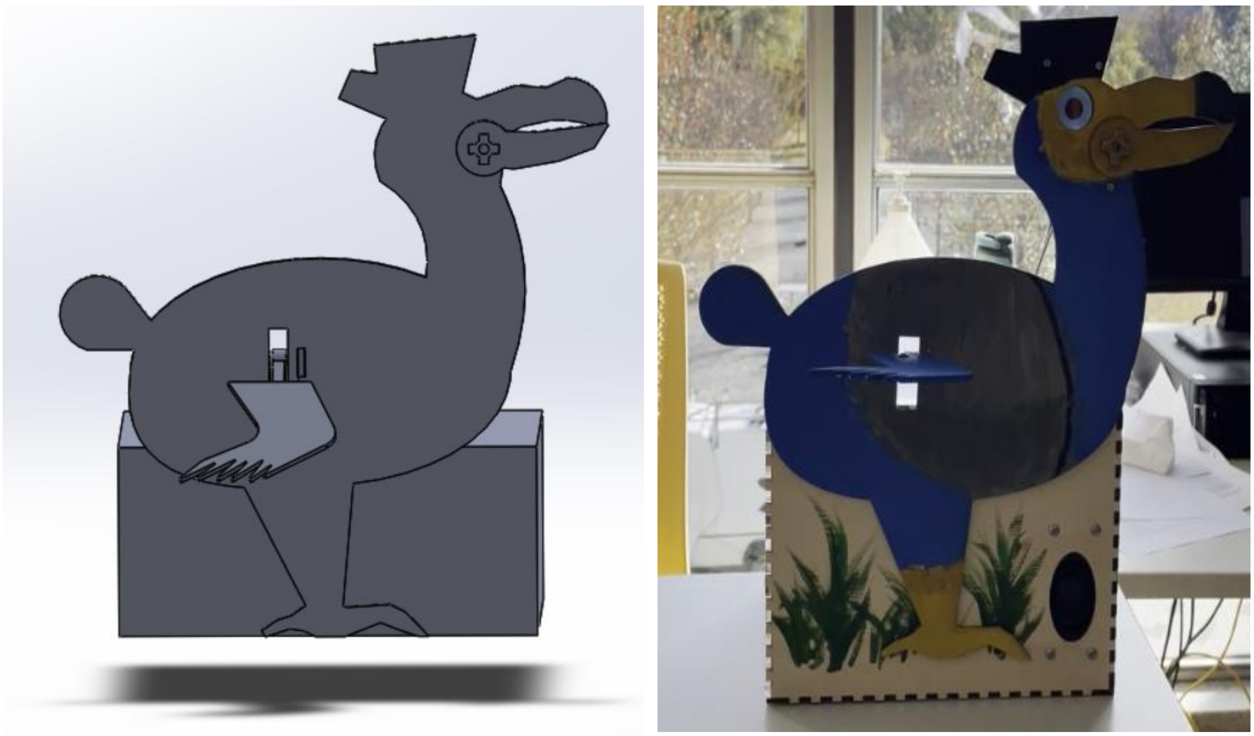

Figure 1: Our initial rendering of the dodo bird and our completed MojoDodo.

System Design

The inputs of the system include 120V AC input power, Wi-Fi signals from a director, and user inputs via 5 push buttons and a potentiometer. The outputs of this system include a speaker, an LCD, LEDs, and the movement produced by the figurine driven by motors. The system consists of 6 sub-systems including the user input, audio, mechanical, power, control, and lighting subsystems. The user interface consists of 5 buttons: shine, squawk, friendly, angry, and fun. The processor is an mBED LPC1768 using an Adafruit Huzzah for Wi-Fi capabilities. The audio subsystem includes a 2W speaker and a Class D audio amplifier. The motion subsystem includes two motor drivers: two servos. The power subsystem includes a 120V AC to 5V DC transformer, a user input switch, and a fuse.

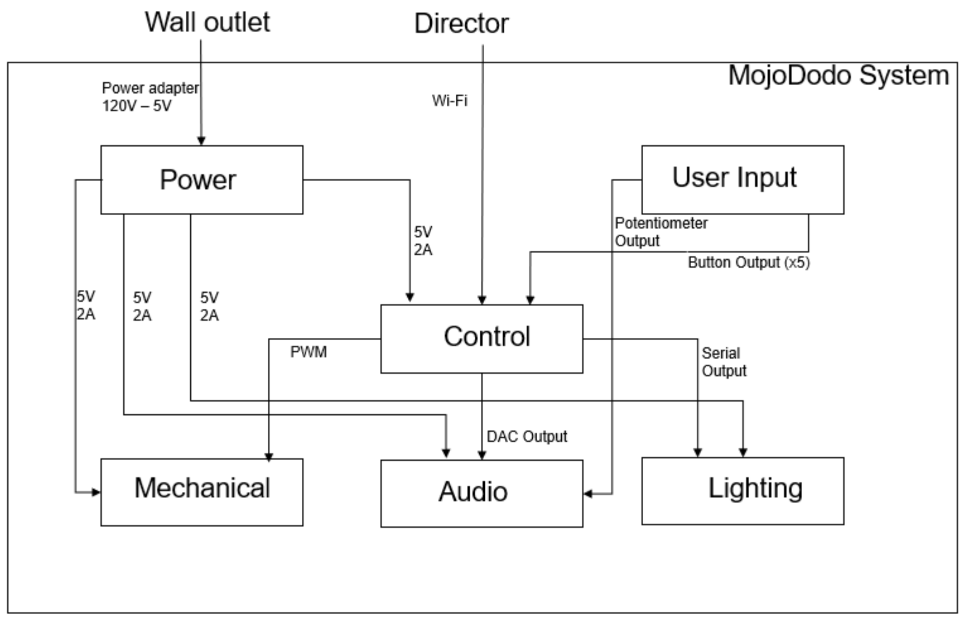

Figure 2: System Block Diagram

Interface	Source	Destination	Description
120V	System Input	Power	120V AC will be input into the system and stepped down to 5V, 2A
Voltage	Power	Control	5V, 2A
	Power	Mechanical	5V, 2A
	Power	Audio	5V, 2A
	Power	Lighting	5V, 2A
GPIO	Control	Lighting	Serial output control for LCD screen
	Control	Audio	DAC output signal to speaker
	Control	Mechanical	PWM signal to each servo
User I/O	User Input	Analog	User turns a 2500 Ohm potentiometer to adjust volume
	User Input	Control	User pushes 5 push buttons to command different operating modes
	User Input	Power	User flips a power switch to turn the system ON/OFF
Wifi	Director	Control	Wi-Fi signal transmits information from the Director to be processed by the system

Table 1: System Interface

Sub-System Designs

Our design incorporates 6 sub-systems: user input, power, control, mechanical, audio, and lighting. These 6 sub-systems will operate and perform based on their descriptions below.

User Inputs Sub-System

The user inputs will determine the operating mode of the system based on the push button pushed by the user. It will also determine the volume of the speaker based on the position of the volume knob (a 2.5 kilo Ohm potentiometer) chosen by the user.

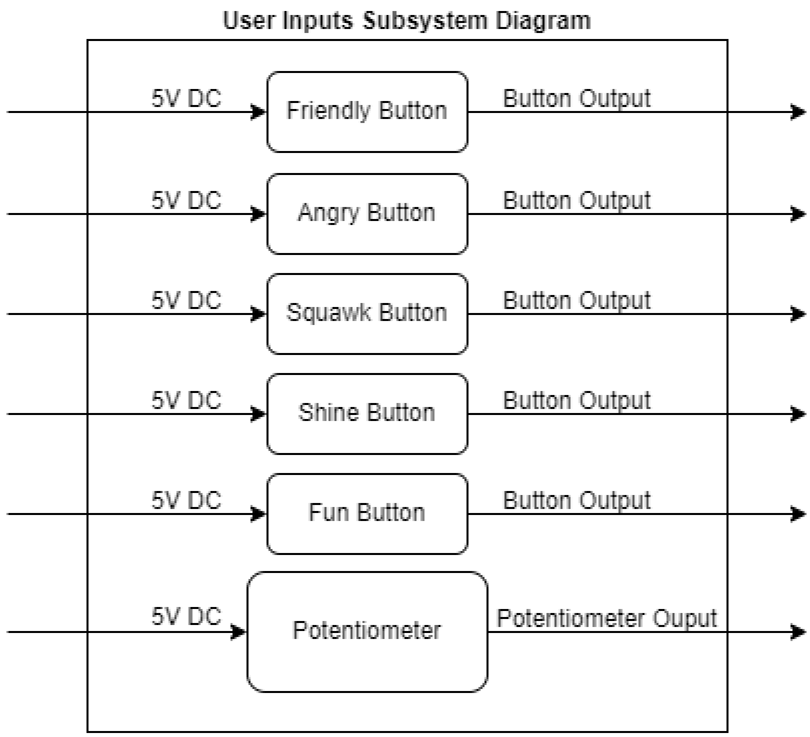

Figure 3: User Inputs Subsystem

Audio Sub-System

The audio subsystem will take 5 Volts DC and receive an analog signal from the control subsystem, then amplify it in the class D amplifier and finally play it through the speaker.

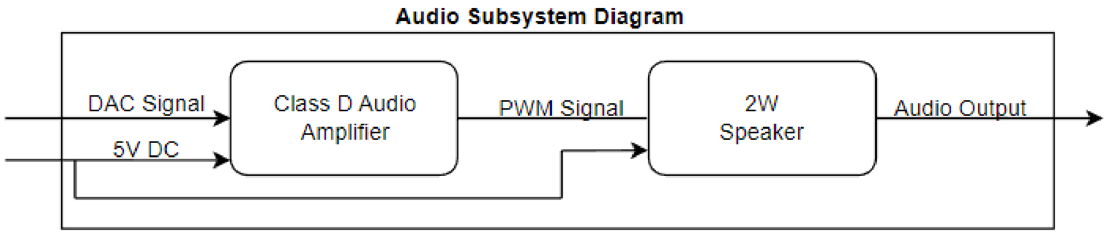

Figure 4: Audio Subsystem

Mechanical Sub-System

The mechanical subsystem consists of two servo motors that drive two points of actuation on the dodo bird. One point of actuation is the beak and the other is the wing. Each servo motor requires a 5 Volts DC input for power and PWM signals for controlling direction.

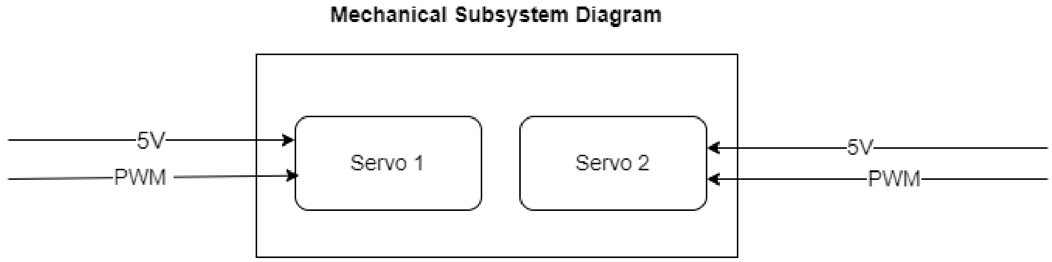

Figure 5: Mechanical Subsystem

Power Sub-System

The power subsystem will take 120 Volts AC as an input and output 5 Volts DC. It will route power through a 10-Watt wall adapter, then through a user input switch and a fuse before distributing it to the other sub-systems.

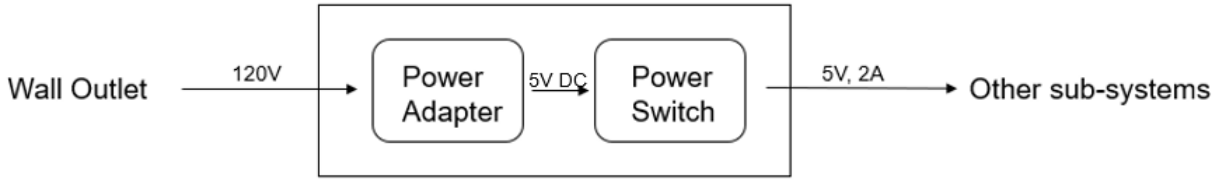

Figure 6: Power Subsystem

Control Sub-System

The control subsystem will have a 5V DC input and will output all necessary signals to the other subsystems. This includes a serial output to the lighting subsystem, a DAC output to the sound subsystem, and PWM signals to the mechanical subsystem. The Wi-Fi module will communicate with the Director and send any information to the mBED, which will then produce those output signals.

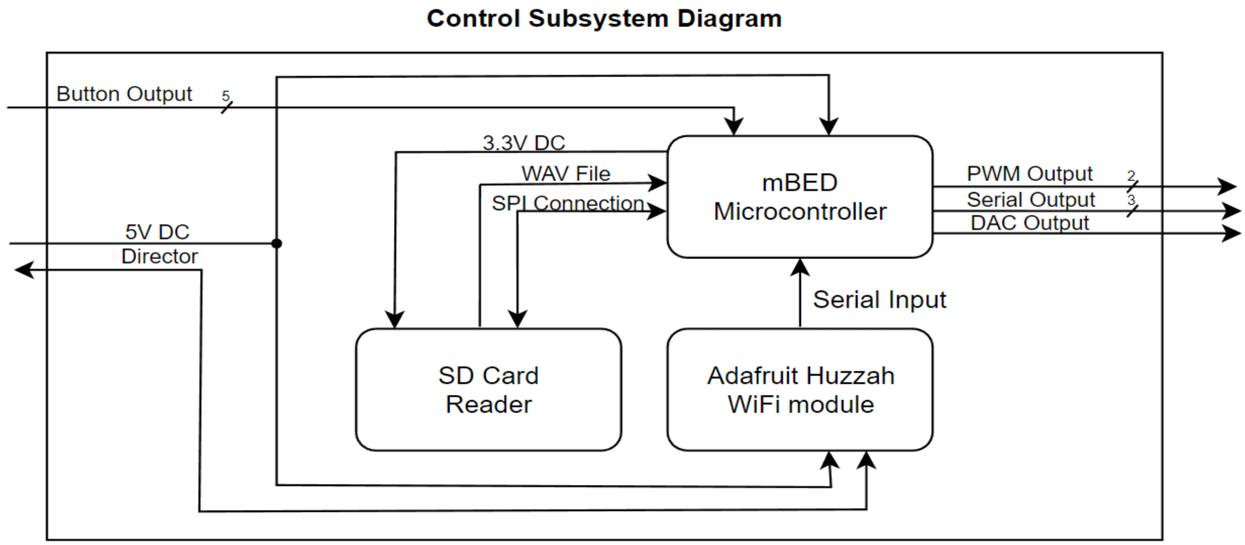

Figure 7: Control Subsystem

Lighting Sub-System

The lighting subsystem will take 5V DC and receive a serial connection from the control subsystem to then display static Green LEDs (5) and output specific pictures/animations to an LCD screen.

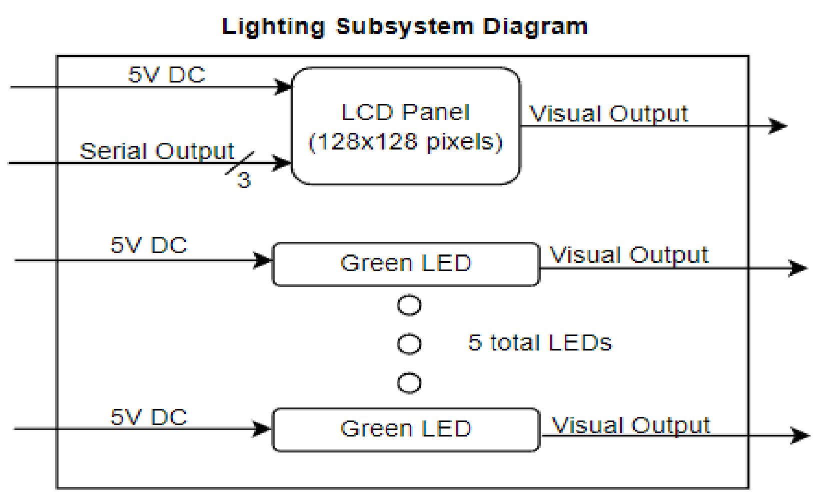

Figure 8: Lighting Subsystem

Constraints, Alternatives, and Trade-Offs

In our analysis of which components to use for our motors, we considered utilizing a solenoid, stepper motor, or servo motors. We decided not to use the stepper motor due to the total cost of our materials increasing significantly, and we moved away from using a solenoid because of the limitations associated with rotational movement. In the end, we decided to incorporate two servos for the beak and the wing movement, which simplified cost and movement restrictions.

One constraint identified with our control subsystem was that the chosen mBED LPC1768 processor did not have a built-in Wi-Fi module. As such, we had to resort to using an external Wi-Fi module: the Adafruit Huzzah.

An alternative to the mBED processor could be a Raspberry Pi processor, however budget concerns and the difficulty of integrating a Pi with motors and other peripherals were believed to outweigh the benefit of having built-in Wi-Fi and Python capabilities. Additional trade-offs that occurred with the mBED include not having a display output and limited support for external libraries. That said, we believed the benefit of having established C++ classes for all of our outputs outweighed these issues.

Electronic Design

Our electrical design is centered around an mBED microprocessor, a custom-built printed circuit board (PCB) and other components to be soldered onto the PCB. The figures below show a high-level block diagram of the system, a detailed schematic that includes information related to component pin connections to the mBED and a schematic of our PCB layout which details where the component headers are to be located.

The high-level component block diagram includes all inputs such as push buttons, a potentiometer, and a power supply. Each of the inputs leads to the microcontroller, and produce outputs to either the LCD screen, LEDs or the speaker.

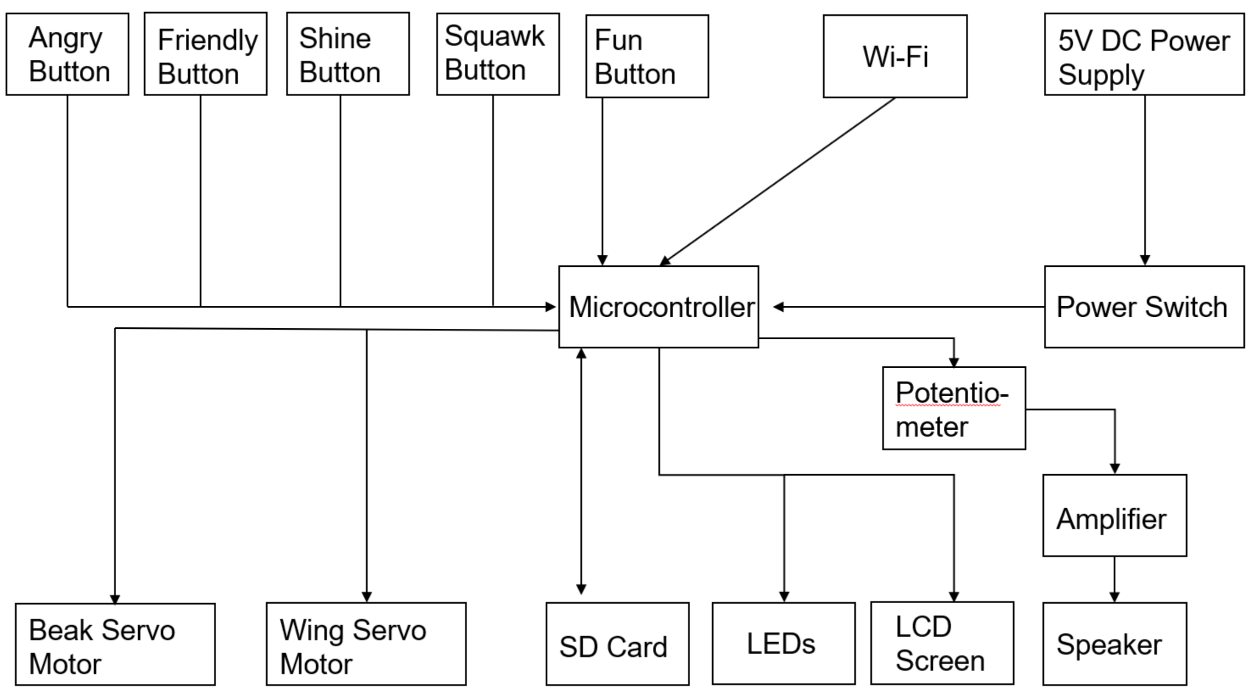

Figure 9: Electronic Component Block Diagram

Created in Eagle, our schematic translates the high-level component block diagram into a connected diagram that details a pin connection for each component.

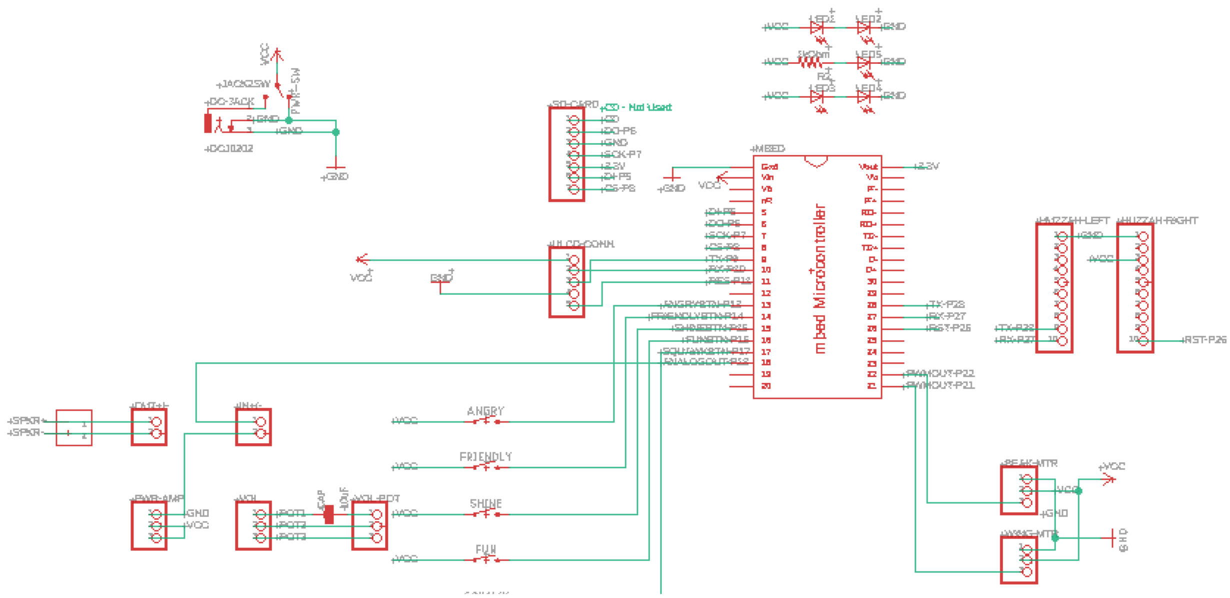

Figure 10: PCB schematic created in Eagle.

From the schematic that we created in Eagle, we obtained a board view of the PCB with appropriate traces and connections. This is a two-layer PCB design with our team's name and class section printed on the silkscreen.

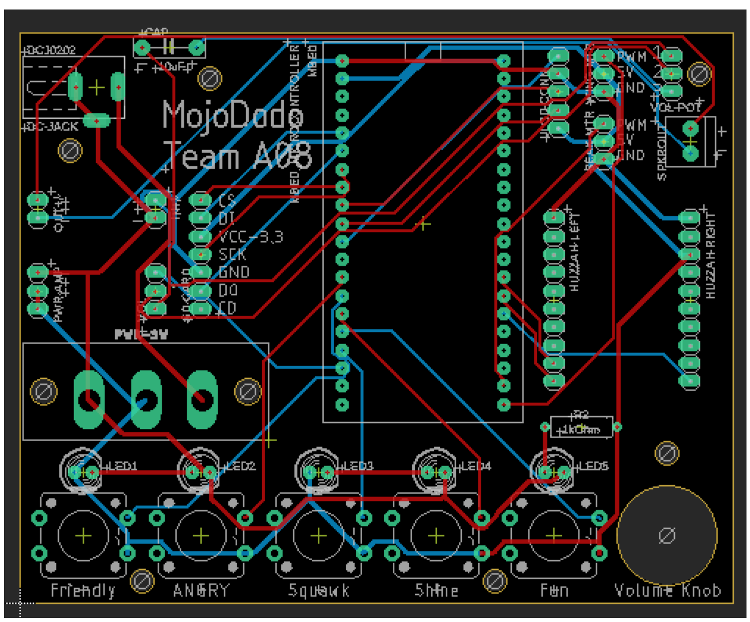

Figure 11: PCB board design created in Eagle.

Mechanical Design

The mechanical design of the project consists of a dodo bird that can actuate its wing and beak. The dodo bird is mounted in front of a box that will house all the electronics. The main mechanical features are a servo motor to actuate the wing flapping and another servo motor that will open and close the beak.

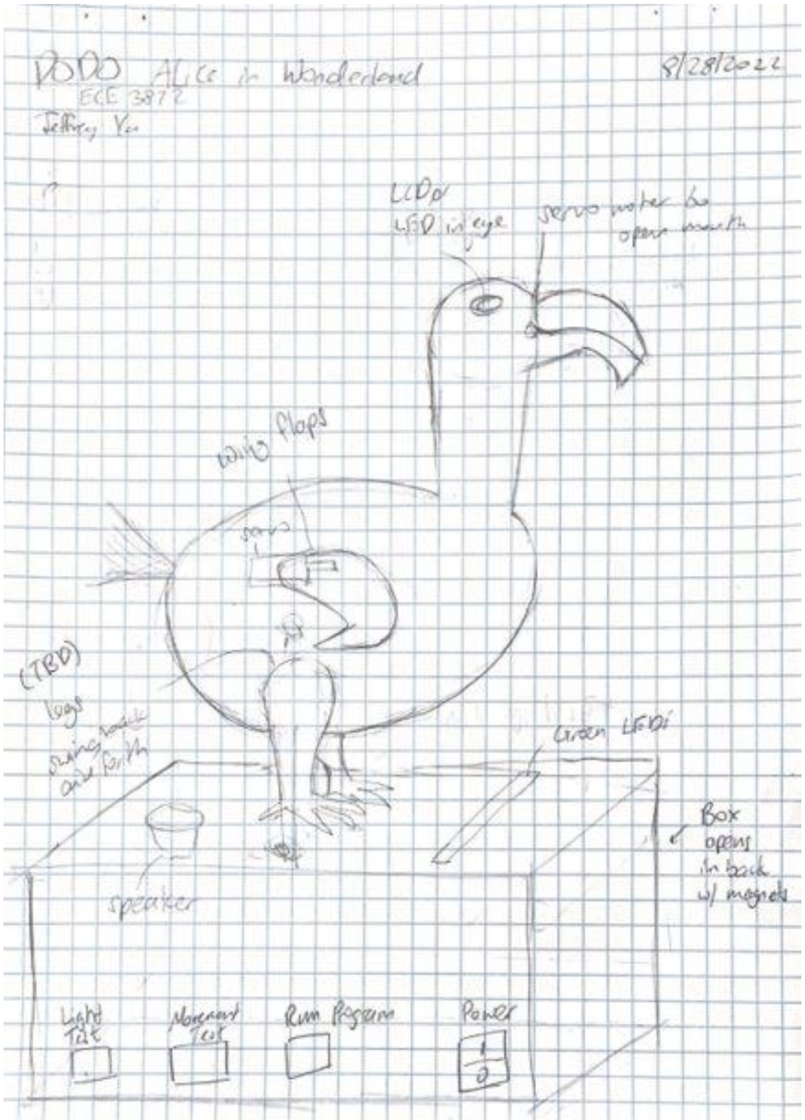

Figure 12: Initial sketches of the dodo design.

The design then transitioned to how to actuate the wings and the beak. After simulating the servos with weights and testing the ranges of motion of the servos, they were deemed to be best suited to perform the tasks of actuating the wing and beak. Two custom mounts will be used to attach the beak and wing to the servos.

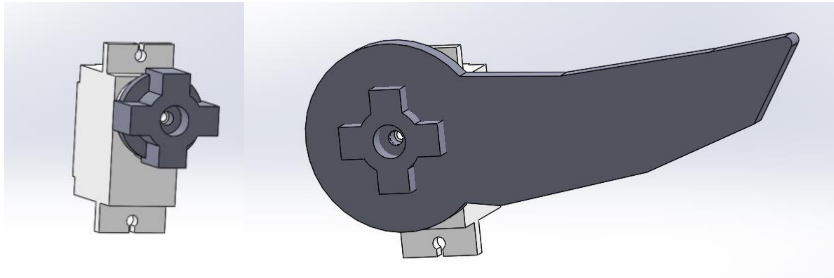

Figure 13: Servo Beak Mount

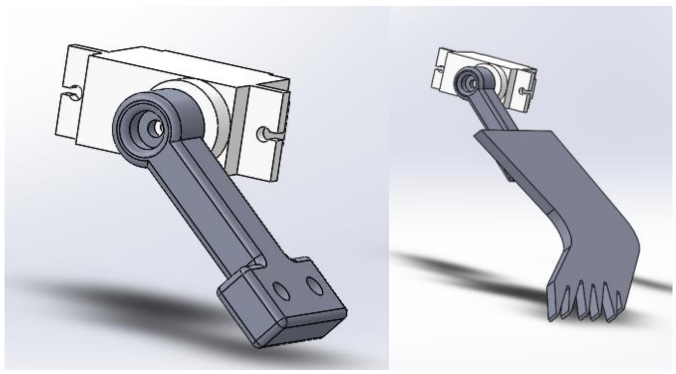

Figure 14: Servo Wing Mount

The assembly of the dodo was done in Solidworks to help visualize the size and workings of the components. The material of the bird and most of the box will be 1/8" wood that will be laser cut. To convert the Solidworks components to laser cutting files, a Solidworks drawing was made that was then converted to an Adobe Illustrator file, which the laser cutter uses. There will be a clear acrylic side on the box that has the PCB mounted onto it to showcase it. The final dimensions of the box will be 20cm wide (lengthwise with dodo), 10cm high, and 15cm deep. This is adequate to fitting the PCB and the speaker. Below is the full CAD of the bird and the box.

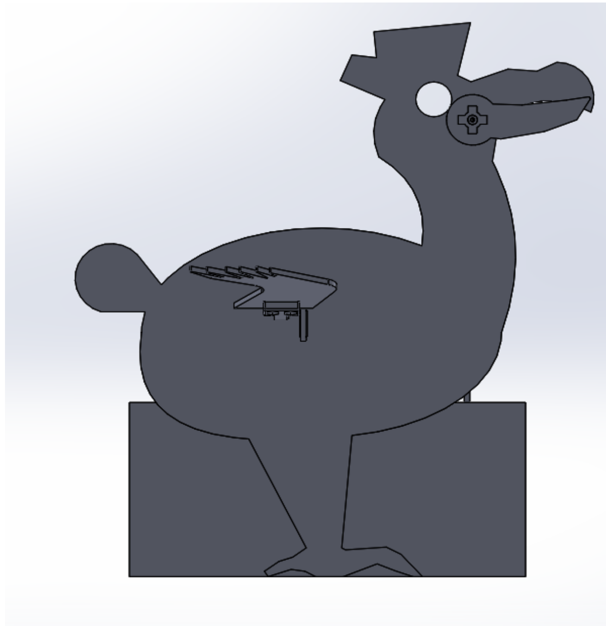

Figure 15: Front view of whole MojoDodo

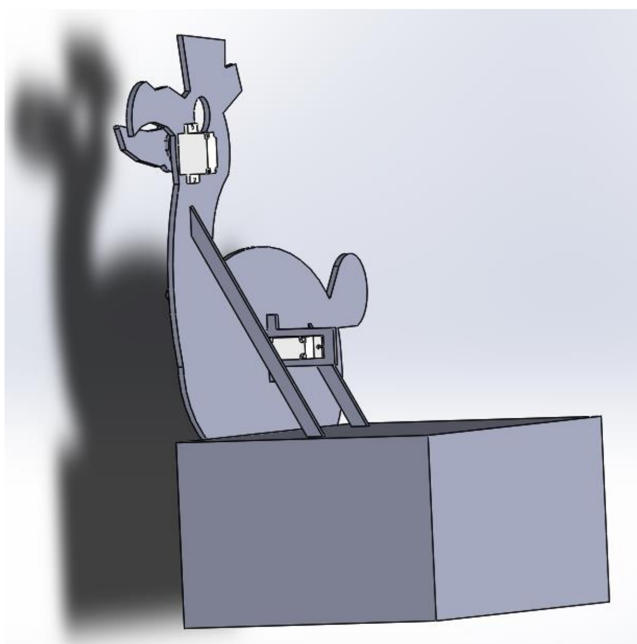

Figure 16: Angled view of whole MojoDodo

Figure 17: Rear-view of cardboard mock-up of MojoDodo.

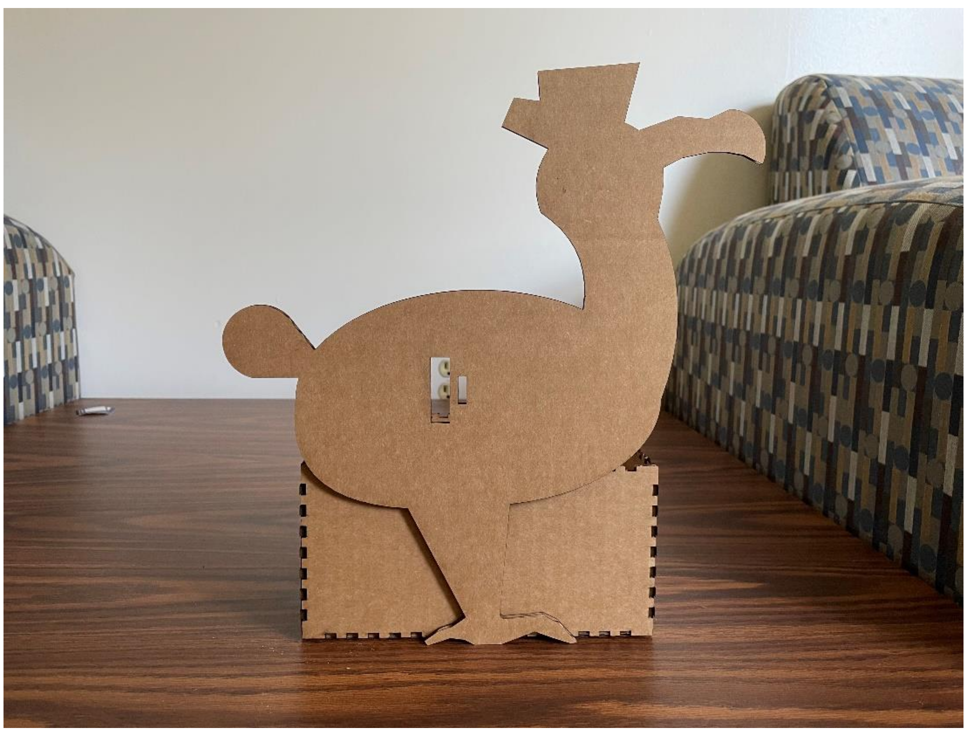

Figure 18: Front-view of cardboard mock-up of MojoDodo

Software Design

Our software design includes 7 different states—off, idle, test, connect, receive, decode, and play as shown in Figure 19. The testing/user input state is divided into five buttons that perform different functions. Using these states and our state diagram, we were able to design the software architecture in Figure 20.

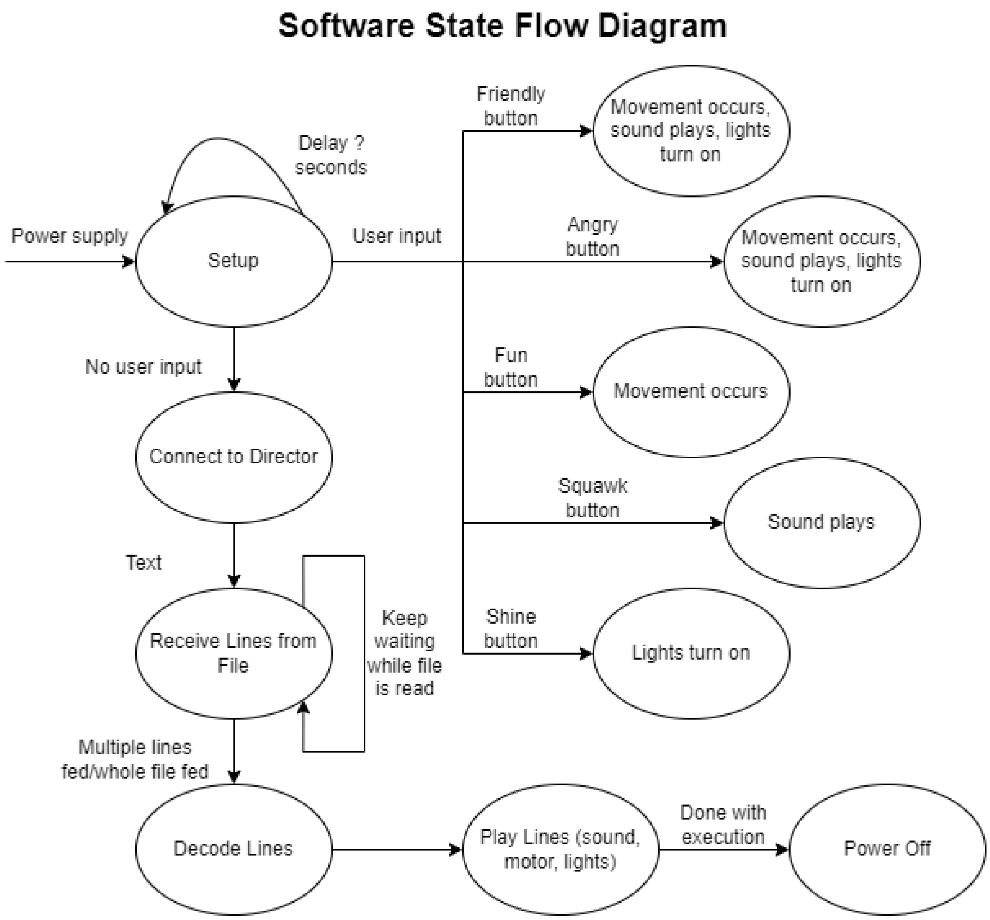

Figure 19: Software state flow diagram.

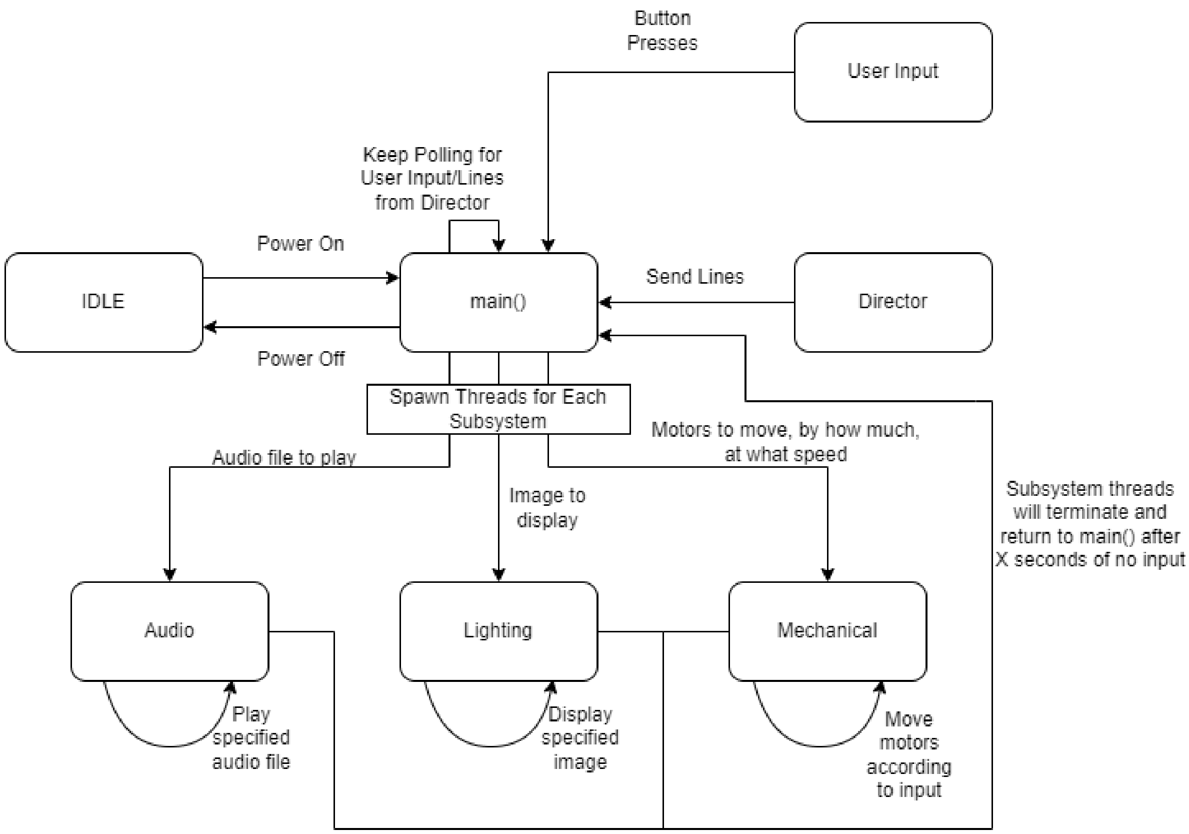

Figure 20: Flowchart detailing the software architecture that includes the threads that dictate the robot's movement and audio/visual outputs.

Schedule

The schedule for completion of the project is shown in Table 2 where the task lead is indicated on each task.

Table 2: Schedule to complete MojoDodo

	Week Number										
Task	**1**	**2**	**3**	**4 (9/12)**	**5 (9/19)**	**6 (9/26)**	**7 (10/3)**	**8 (10/10)**	**9 (10/17)**	**10 (10/24)**	**11 (10/31)**
Brainstorm	Jason Q.										
Finalize Top Level Design	Jason Q.										
Proposal											
Design Power (some simulation ready by PDR)				Tom							
Define power subsystem block and interface diagram											
Research power input techniques											
Acquire parts for both Mechanical and Electrical											
Design User Inputs (some simulation ready by PDR)				Tom							
Define user input subsystem block and interface diagram											
Determine type of input devices to use											
Test acquired parts for input readability											
Design Controller (some simulation ready by PDR)				Jason H.							
Research microcontrollers to use			Jason H.								
Develop code framework				Jason H.							
Breadboard and test microcontroller					Jason H.						
Design Mechanical (some simulation ready by PDR)				Jeffrey							
Research methods of articulating motion											
Determine type of motors to use											
Test mock wing assembly											
Design Audio (some simulation ready by PDR)				Jason H.							
Research speaker types and driver circuits											
Test speaker output in breadboard circuit											
Design Lighting (some simulation ready by PDR)				Jason Q.							
Determine type of LED/LCD to implement											
Test lighting elements on breadboard											
PDR											
Design Power (100% simulation ready by CDR)							Tom				
Test power delivery and current output											
Test power subsystem integration to other subsystems											
Produce printable pcb files for power subsystem											
Design User Inputs (100% simulation ready by CDR)							Tom				
Test input devices' integration with Controller subsystem											
Dry fit components to ensure fitment											
Produce printable files for user input subsystem											
Design Controller (100% simulation ready by CDR)							Jason H.				
Verify I/O to microprocessor											
Dry fit components to ensure they will fit in the Dodo											
Produce printable pcb file for controller subsystem											
Design Mechanical (100% simulation ready by CDR)							Jeffrey				
Design and simulate mouth											
Design overall body and skeleton											
Design controller and support box											
Design Audio (100% simulation ready by CDR)							Jason H.				
Simulate speaker playing a sound file											
Ensure power usage is within specifications											
Produce printable											
Design Lighting (100% simulation ready by CDR)							Jason Q.				
Verify visual output											
Test acceptable input range											
Produce printable lighting subsystem files											
CDR											
Build Power								Tom			
Print and test Power pcb											
Test full range of power output to other subsystems											
Build User Inputs (expand to Component-level is needed)								Tom			
Print and test User Input pcb											
Test full range of user inputs to other subsystems											
Build Controller (expand to Component-level is needed)								Jason H.			
Print and test Controller pcb											
Test code is running properly on microcontroller											
Build Mechanical (expand to Component-level is needed)								Jeffrey			
Laser cut components											
Print and test mechanical limbs of the system											
Test range of movement and speed of movement											
Build Audio (expand to Component-level is needed)								Jason H.			
Print and test Audio pcb											
Test range of audio output and volume control of speaker											
Build Lighting (expand to Component-level is needed)								Jason Q.			
Print and test Lighting pcb											
Test range of colors and brightness of the LEDs/LCDs											
Test Power subsystem as a whole									Tom		
Test User Inputs subsystem as a whole									Tom		
Test Controller subsystem as a whole									Jason H.		
Test Mechanical as a whole									Jeffrey		
Test Audio as a whole									Jason H.		
Test Lighting as a whole									Jason Q.		
System Integration - Ensure all subsytems fit and communicate properly										Jason Q.	
Final System Test - Ensure entire system meets all requirements after integration and subsystems' behaviors have not changed.											Jason Q.
Final Inspection and Demonstration											

Integration

Our plan for integration was as follows: (1) begin by designing software for both the mechanical and electrical subsystems, (2) build a breadboard prototype using electrical components, which can be done simultaneously while step one is occurring, (3) test the software with the breadboarded parts and the mechanical components in isolation of each other, (4) and combine the electrical and mechanical subsystems together with the software.

Throughout each step, we aimed to continuously debug and go back and forth between steps to isolate problems as they occurred. When we experienced issues with step 4, we failed over to step 3 to determine the source of the error was power related. While we had a procedure, we did not strictly adhere to it. Flexibility was introduced by initially writing software for the audio and the lighting subsystems and simulating tests for them, after which we added software for user input controls and then simulated all of them together.

We began by writing software in C/C++ using mBED's online compiler, which we then used to test the capabilities of our servo motors as well as to test our audio files playing through a speaker. We proceeded to incorporate software for our LEDs and LCD screen and tested those individually. Applying multithreading, we added our usage of push buttons to the system and combined the user input, lighting, and audio subsystems to perform at the same time. The Wi-Fi module was later introduced to give our system the ability to communicate with the Director, and we successfully succeeded in feeding commands to the system. When we approach the building of our system, we will combine our successfully tested subsystems with the mechanical subsystem and our custom PCB.

Our largest integration problem was with the 3.3V SD Card reader. Initially during breadboarding, the card reader was powered using the VOut pin of the mBED processor. After breadboarding, the debugging micro USB cable was no longer connected to the mBED. Unbeknownst to our team, without the micro USB cable the VOut pin did not provide sufficient current to power the card reader. As such, when only running on external power (DC jack) on the PCB, the SD card would consistently fail to initialize. A workaround was implemented to alleviate this issue: a shunt resistor was put in place to bridge VDD (5V) to the SD Card reader's Vin pin.

Conclusion

Over the course of 12 weeks, our team was able to successfully design, build, and integrate the MojoDodo. Working through dividing the project into smaller sections and tasks made the project much more probable to finish. Through starting software earlier than the other subsystems, the path to completion was smoothed out compared to other strategies. Mechanically, the dodo was not too challenging to implement once our team learned how to cut the materials with the tools provided in the makerspace. Once the servos had a mount and point of articulation set, the dodo was very easy to manipulate mechanically through software. Through breadboard testing before integration, many small errors were caught far before they would become schedule blockers.

In the next iteration of a similar project, our team would make some changes to improve quality of life. This includes creating a fuller schedule earlier in the project, and putting more research into the capabilities of the microprocessor chosen for the project. For example, if more research was put into the mBED microprocessor, the team would have known about the VOut pin limitation that was described in the Integration section – and would have saved hours of headache. Overall, our team was pleased with the quality of the final project, and the team was ecstatic that all requirements were met as they were set out in the decomposition stage.