SAN DIEGO STATE UNIVERSITY

ECE Firmware Readiness Report

EE/COMPE 496B Senior Design

Team 14, Project 25: Rapid Deployment Runway Closure System

The Blockage Brigade

ME Team: Alyssa Elkins, Timothy Turner, Nicolas Wolford, Ala Zeidan

ECE Team: Sean Connolly, Khalid Nunow, Jomari Paguia, Marc Tanwangco, Bianca Yousif

Table of Contents

Executive Summary - Page 2

System Process Description/Diagram - Page 3

State Diagram/Flow Chart - Page 6

Protocol Descriptions - Page 10

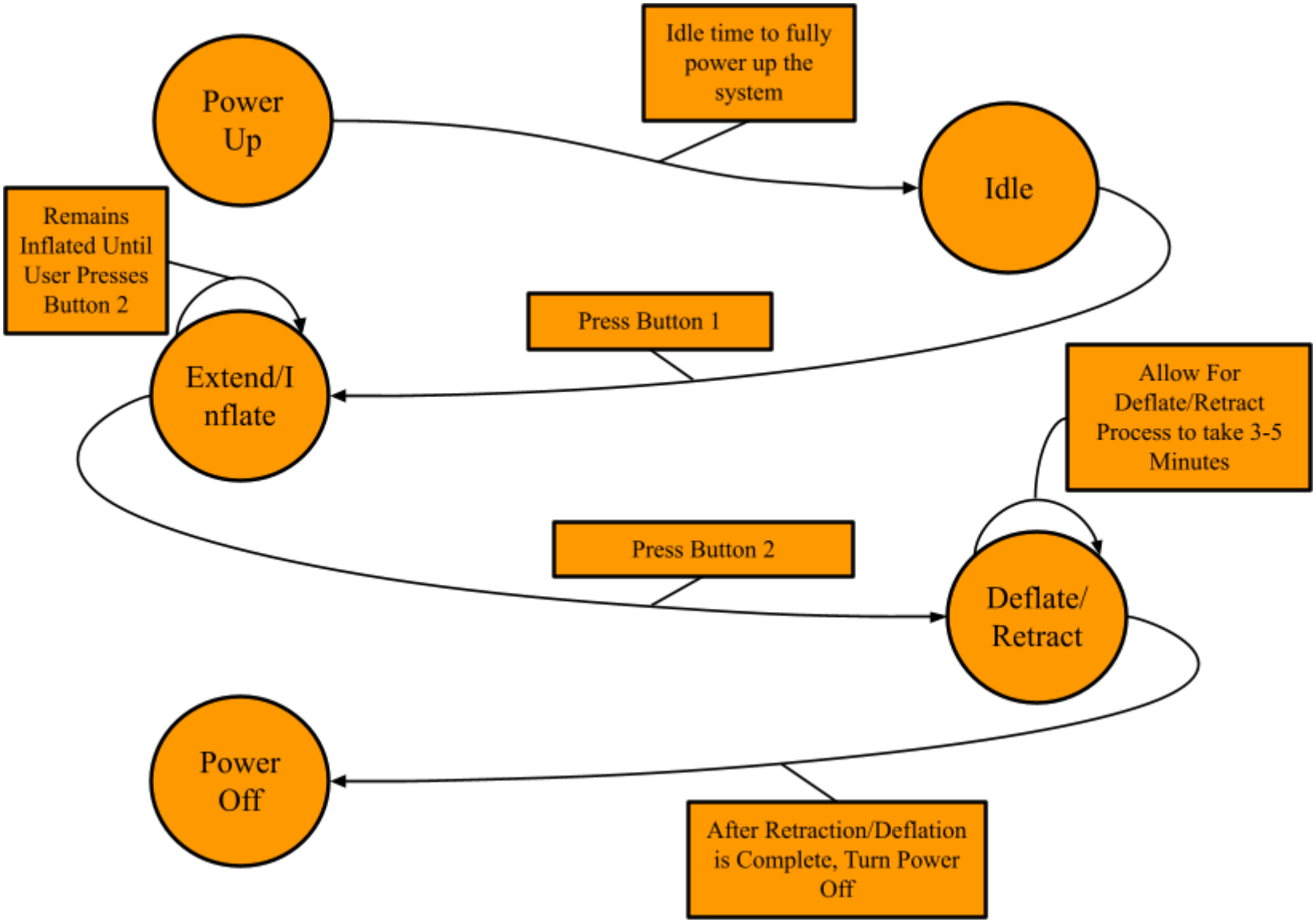
Validation Results - Page 11

Executive Summary

In this firmware readiness report, the team has put a detailed firmware design for the Rapid Deployment Runway Closure System. It will start off with a detailed description of how the system realizes its intended functionality using firmware. In the PCB that was previously designed, there will be a power switch included that will turn the system on and off and run the declared motor. To get a better understanding, a diagram has been included. After this, a state diagram was created to show the program flow of the firmware. The state diagram shows only the key algorithms and processings for the firmware of the system which include the “Extend Model,” the “Inflate Model,” and the “Retract/Deflate Model.” To put it in simpler terms, the section of the “System Process Description/Diagram” focuses on what the software does for the user while the “State Diagram/Flow Chart” focuses on how it works. There will also be a few protocol descriptions such as an external power switch and the N-channel MOSFET, since the firmware communicates with other devices.

In this report, it has been shown that the team has written and tested the firmware to the greatest extent possible without the project-specific hardware. A motor driven circuit, bidirectional motor driving circuit and a fan driving circuit had been created and tested in order to actively sink and source the current by using the Arduino Uno directly.

System Process Description/Diagram



Shown above in Figure 1 is a diagram showcasing what the software does for the Rapid Deployment Runway Closure System. The entire project code resides in the Arduino Uno and how it would interact with each component in the system.

In the PCB, there will be a power switch that allows the user to turn the system on and off. When the user wants to use the system to prevent an aircraft from landing on the runway, they would turn on the power using the switch. Button one represents the extension and inflation of the inflatable. This is one of two buttons that are on the systems that the user has and when this button is pushed, an LED will be lit up and the button is telling the software to begin the extending process for the inflatable. When the user pushes the button, it allows for Motor 1 to be set on HIGH so that it could begin unraveling the inflatable from the spool. The system will be able to drive across the runway, however, if the wheel system does not drive straight as intended, the user could drag the inflatable out and stick a peg onto the end of the inflatable to prevent it from moving around from the desert-like environments. After the user sees that the inflatable is fully extended and has pegged the inflatable to the ground, the user will now see that the fans will be turned on after a delay from the motor extending the inflatable. There would be another delay so that the fans could be powered up in maximum speed to blow up the inflatable and a secondary delay to have at most just one fan to run constantly to keep the inflatable inflated.

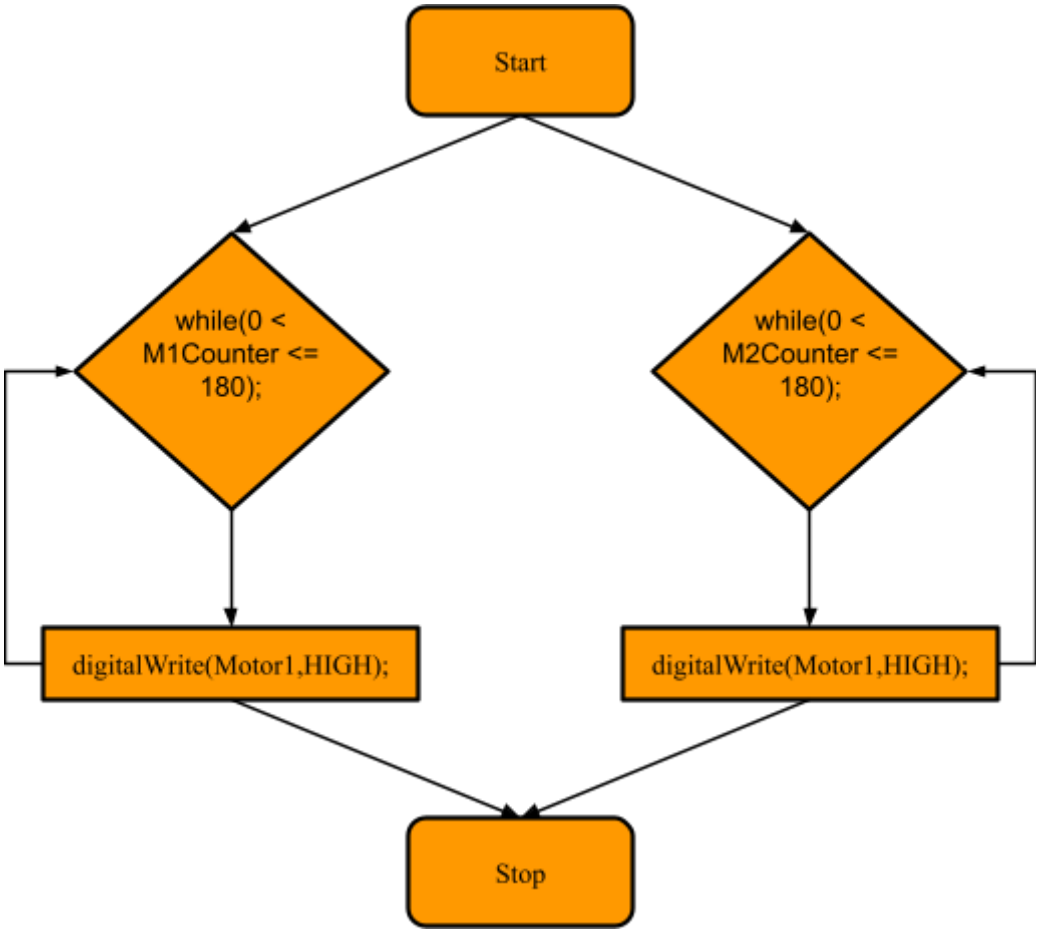
Once the system is fully operational and the user sees that they have accomplished what they needed with the system, they would then press the second button for the deflate and retract process. Within this process, the second LED would be lit up to signify that the system is in the deflate/retract process and through the deflate process, the user would see that all fans that are used to deflate would be turned on HIGH to deflate the inflatable as quickly as possible, but would first see a delay to make sure the fans are up to maximum speed. Then, they would notice

that the inflatable is getting deflated and once fully deflated, all the fans would be set on LOW. The user would then see motors one and two be set to high so that the retraction process would begin and see that the inflatable is being put back into the system. Within 3-5 minutes, the inflatable should be fully back into its system.

Once everything is back into the system and there were no issues with how the inflatable was retracted back into the system, the user would then shut off the system and then store it back into the aircraft carrier.

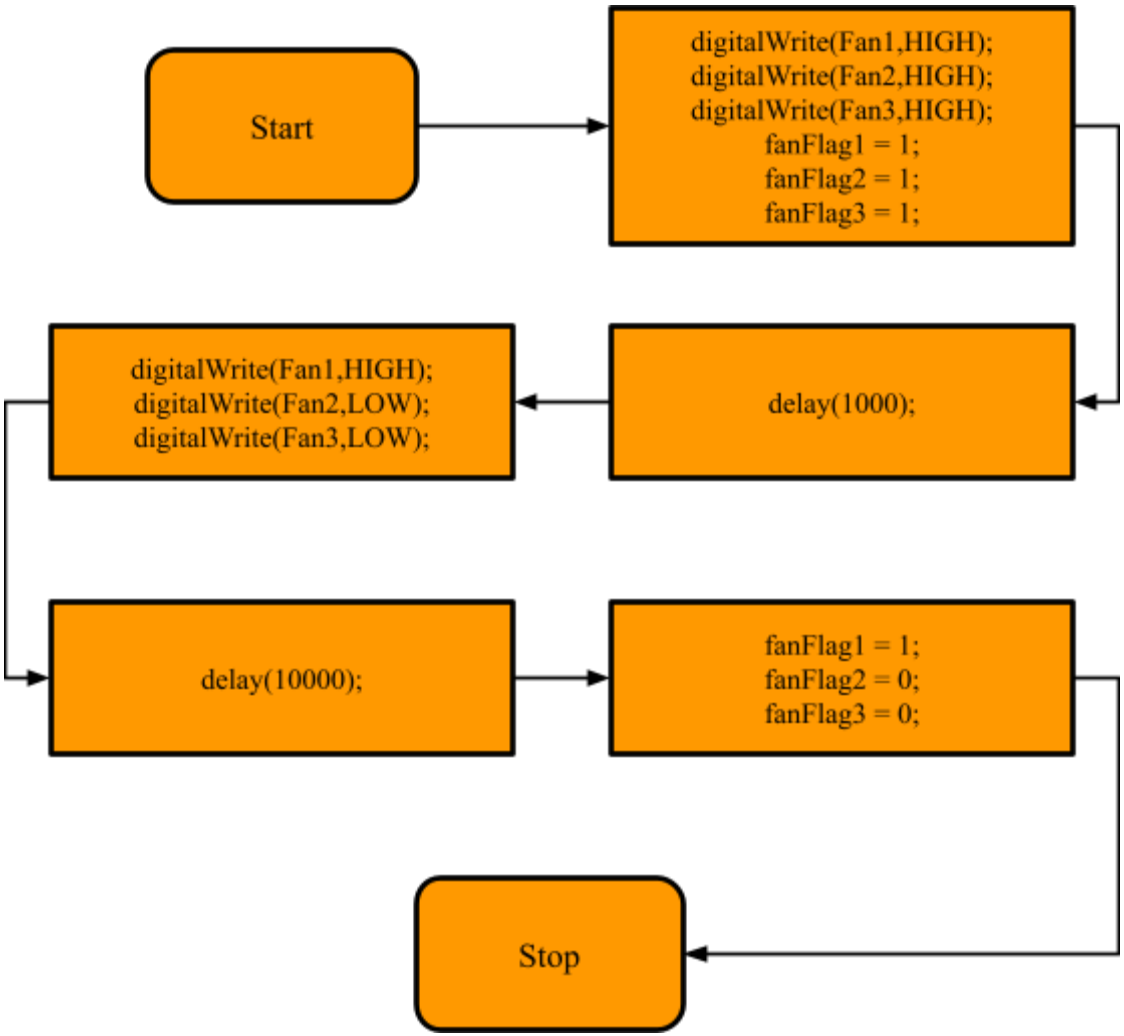
State Diagram/Flow Chart

Extend Module:



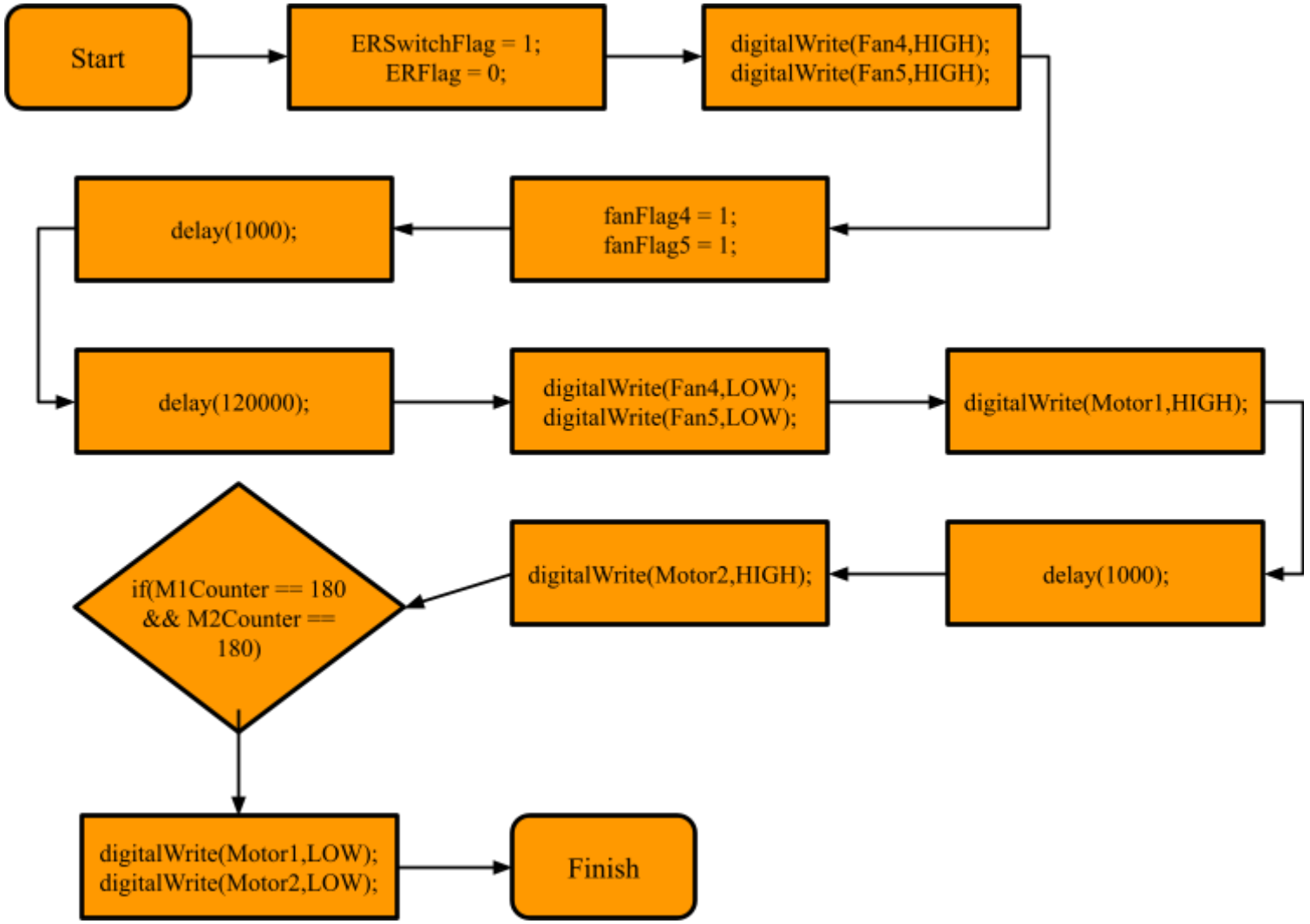
For this module, it showcases how the inflatable will be extended. From the start, there would be two while loops that determine if the M1Counter or M2Counter are between and that represents the two motors that we are using for the system. Within the while loop, it is necessary that the counters be between 0 and 180 because it is crucial that there is enough pulses for 70 feet, which is about the length of the runway. If the counter satisfies the while loop, it would constantly run motor 1 on high until it does not satisfy the while loop. Once it does not satisfy the while loop, it would stop and go to the inflation module.

Inflate Module:



In the inflate module, it would start where the retract module left off and have fans one, two, and three set on HIGH so that it could be turned on and have the three fan flags be set to 1. Then, there would be 1 second delay to allow for the fans to get up to speed. There is another delay to allow for the fans to continue to run for a period of time, however, it is only necessary to have one fan to remain on, so the other two fans are set to low and the flags to zero. This would continue to run until the user decides to start the deflation and retractions process.

Deflate/Retract Module:



In the deflate/retract process, the Extend/Retract Switch Flag is set to 1 and the Extend/Retract Flag to 0 to signify that the system is in the retract phase. Then, fans four and five are set to HIGH as well as the flags for the fans to 1 to begin the deflating process. This allows for at least 2 minutes of deflation time. Once fully deflated, the fans four and five are set to LOW and begin the retraction process. Motors 1 and 2 are set to high with a delay of 1 second to allow for the motors to start up. Next, there is an if statement for the two counters so that if it is equal to 180, then it would set both motors to LOW.

Protocol Descriptions

All code in the project is programmed for use on the Arduino Uno. The user will interact with buttons which are the main function input to control the whole system. There is an external power switch, which when turned on, will allow the 12 volt battery to begin delivering power to the Arduino as well as powering up the fans and motor so they will be ready to use. When system power is turned on the firmware is already coded into the Arduino and when a button is pressed, it will activate the part of the system that it controls without any additional assistance. Each motor driver is connected by 2 pins on the Arduino. When the button is pushed, it sets the motor to high which turns on the motor to begin the process of unraveling the spool which contains the inflatable device.

A IRL520N N-channel MOSFET can be driven directly from the Arduino Uno due to its 2-4V gate threshold voltage, and 4.5 amp current capability at 5V gate charge. Switches S1 and S2 have one end connected to ground and the other end connected to header J17. These are pins 1 and 2 of a small connector of J6 so when switches are pushed, they will pull it to ground or to 0 volts in order for the Arduino to have the internal pull up resistor set to high. The firmware sets mosfet as output for the fans, and has A5 set for the fan. Fans connected to the PCB and set on HIGH on the Arduino, will activate the fan chosen. Relay output 1 is set as A4 as the output and Relay output 2 is set as A3 when these are set to High. First, when Motor 1 (A4) is set to high, this will turn on motor in direction 1 which will have a delay for 5 seconds and then be set to low which will turn off the motor. As well as turn the motor in direction 2 for 6 seconds then turn it off at which point it will turn off and the fans will then turn on. Since it is not controlled by a GUI or SPI/I2C, it is fully explained above.

Validation Results

Test	Description and Coverage	Procedure	Results
1	In Figure 5, it shows a 12V battery to power on an Arduino Uno as well as a motor and to do this, a motor drive circuit has to be created in order to turn on and off the motor. The motor is similar to what will be used for the system.	To create the motor drive circuit, a 1.5 kohm resistor was used that goes into a npn transistor through the base and would be able to turn the motor on and off. Then the Senior Design's multimeters would be used to measure the voltages and currents of the circuit and Arduino. The test setup could be seen in Figure 6. Shown in Figure 7 is the test code that was created for the Arduino Uno to be able to turn on and off the motor.	During this test, the voltages and currents from the Arduino Uno were observed and found that there was an overload spike in between the toggling of the on and off. The voltage was measured to be at 12.57V and the current at 2.8mA.
2	A bidirectional motor driving circuit was also created that was similar to the bidirectional fan control that uses the Arduino Uno, 12V battery, and a similar gearmotor that would be used in the system as shown in Figure 7 and Figure 9. The purpose of this test was to see how the motor would run in either direction and measure the voltages and currents between the Arduino Uno and transistors.	In order to create the bidirectional motor driving circuit, there would be 2, 680 ohm resistors connected to a 2N2222a transistor as well as having the motor connected to a couple of diodes and relays. All of these components would then be powered by the 12V battery. In Figure 11, firmware code was created to test the motor to turn in one direction	Within this test, the fan was able to run at its maximum speed. Along with that, Arduino's 5V output was used to drive the low gate voltage directly through a 470 ohm current limiting resistor for 10.6mA peak. Lastly, the voltage across the MOSFET's drain and source pins were measured and came out to be 0.06V with a current of 1.3A

Table 1: Validation Test that Showcases the Three Tests and has the Description/Coverage of Each Test, Procedure, and Results such as Measurements and Findings from the Tests.

		and another.	and a total power dissipation of 0.078W with an RDSON of 0.046 ohms.
3	A fan driving circuit was created that utilizes an Arduino Uno, motor, 12V battery and MOSFET as shown in Figure 8. The purpose of this test was to run the fan at its maximum speed and measure any voltages and currents through the Arduino Uno and transistors.	To create the fan driving circuit, there was a 470 ohm resistor connected to the PWM pin of the Arduino and to the Gate of the IRF520N MOSFET. There was also a 12V, 1.3A motor connected in parallel with a 1N4007 diode. To power all the components, a similar 12V battery was used that will be used for the system. Also, Figure 11 is the firmware code for the fan and to test it could run at maximum capacity when turned on and off.	In this test, the charge and discharge time for the 5V gate-to-source was calculated and it came out to be 1.2 microseconds. The base current was also increased on the 2N2222a transistor to 6mA due to a previous test where there was a long duration current spike. Therefore, it is clear to actively sink and source the current by using the Arduino directly. However, it can be limited in its current handling capability.

Table 1: Validation Test that Showcases the Three Tests and has the Description/Coverage of Each Test, Procedure, and Results such as Measurements and Findings from the Tests.

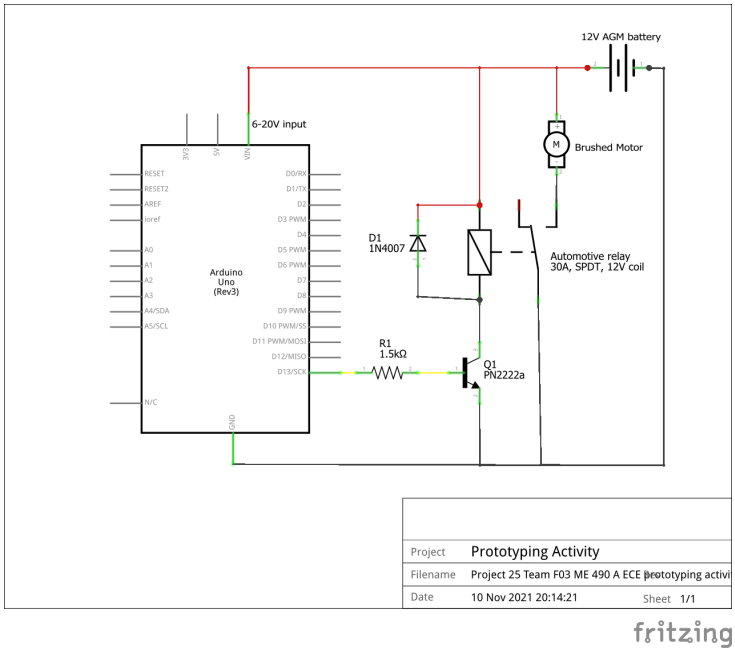


Figure 5: Schematic of the First Test That Showcases the Use of an Arduino Uno, 1.5kohm Resistor, Relay, 12V Battery, Brushed Motor, PN2222a Transistor, and an 1N4007 Diode

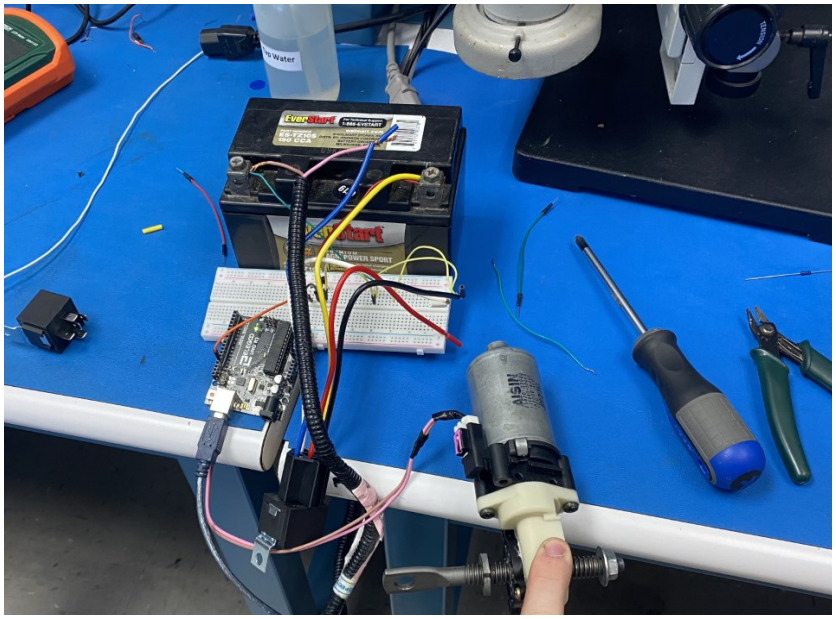


Figure 6: Test Setup of the First Test That Shows All Components Connected Together Ready for Measurements

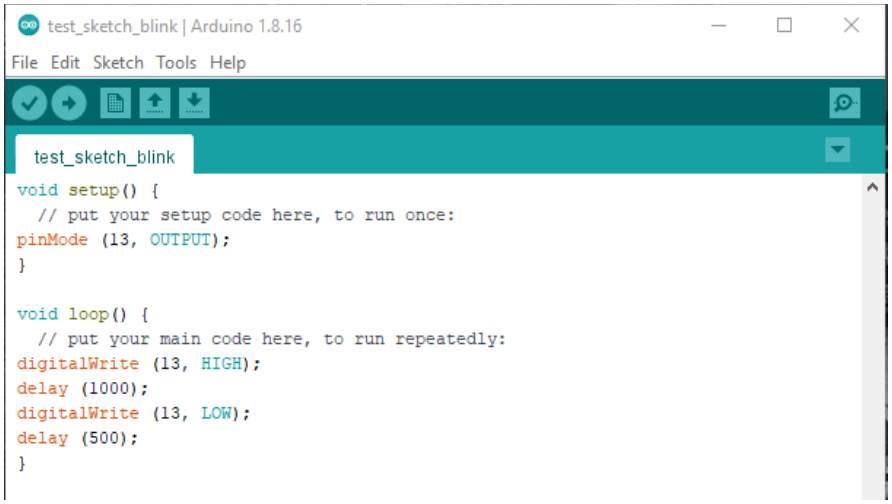


Figure 7: Test Code that was Used in Test 1 That Allows for the Motor to be Turned On and Off

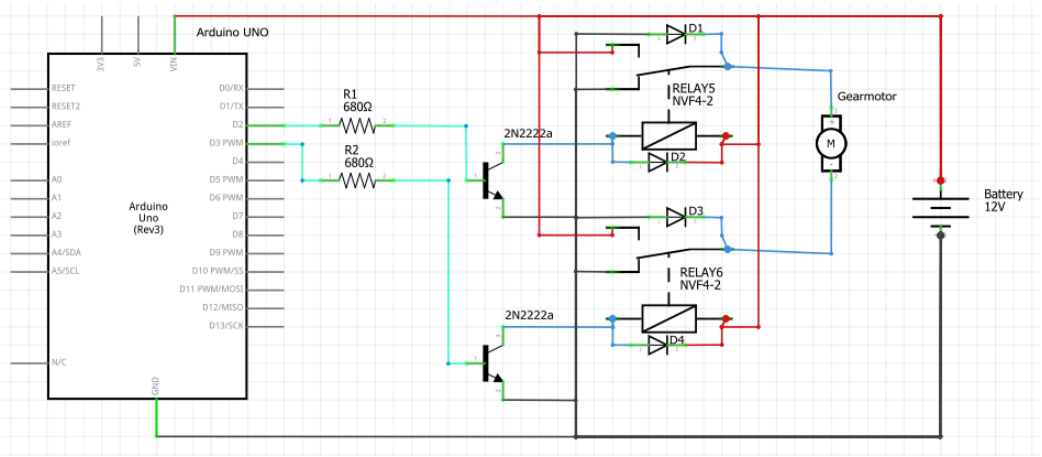


Figure 8: Schematic of the Second Test That Showcases the Use of an Arduino Uno, two 6.8kohm Resistors, two 2N2222a Transistors, four Diodes, 2 NVF4-2 Relays Gearmotor, and a 12V Battery

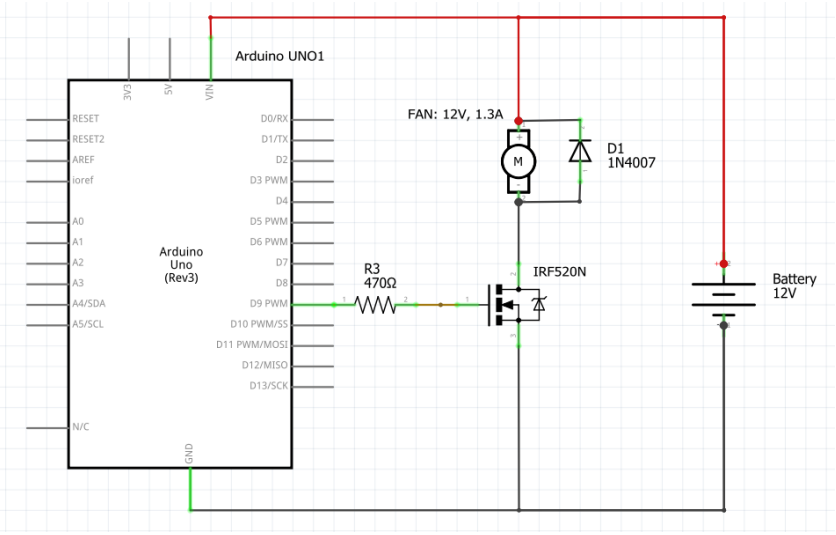


Figure 9: Schematic of Third Test That Showcases the Use of an Arduino Uno, 470ohm Resistor, IRF520N MOSFET, 1N4007 Diode, Fan, and 12V Battery

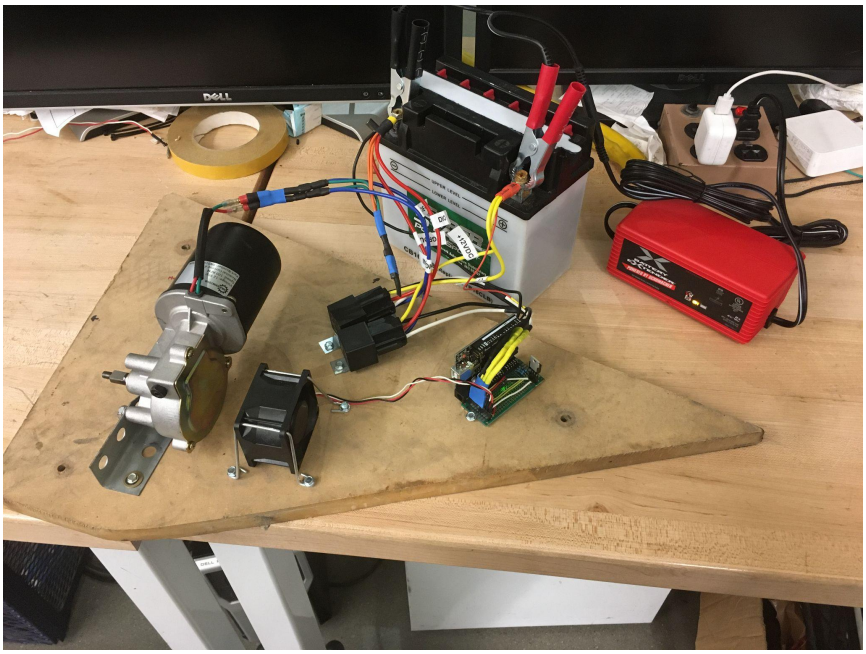


Figure 10: Test Setup of the Second and Third Tests Together Ready for Measurements as Well as for the Incubator Test

Figure 11: Test Code that was Used for Tests 2 and 3 That Allows for the Motor to be Turned On and OFF, Turn in One Direction and Turn in Another, and Allow for the Fan to be Turned On