EE 3330

Design Document

Matthew Smith & Jessica Reiff

07 Dec. 2025

Dr. Tayeb

TABLE OF CONTENTS

1.0 Description of Project

The project involves designing and implementing a custom flight controller for an Unmanned Aerial Vehicle (UAV). The controller will be built around an STM32H7 microcontroller, chosen for its performance, broad community support, and compatibility with established UAV firmware platforms such as CubeIDE. The Printed Circuit Board (PCB) will integrate key input sensors, including an Inertial Measurement Unit (IMU) with a gyroscope, accelerometer, and magnetometer, as well as a barometer for altitude estimation. Sensor data will be processed by the Microcontroller Unit (MCU) using sensor fusion techniques to generate real-time flight control outputs. These outputs will be sent to the Electronic Speed Controllers (ESCs) that drive the motors. While the ESCs are outside this project's main scope, their design may be explored if time allows, through a PWM driver.

2.0 Block Diagram

The block diagram outlines the power and data flow within the flight controller. The USB-C input provides 5V power to the voltage regulation stage, which generates regulated 5V, 3.3V, and 1.8V rails for the MCU and sensors. The MCU communicates with onboard sensors via I^2C and Universal Asynchronous Receiver-Transmitter (UART) through various input and output pins (I/Os), while sending PWM control signals to ESCs via timer outputs. There is also a 25MHz crystal oscillator to help the system with timing. This overview of the system is presented in Figure 1 below.

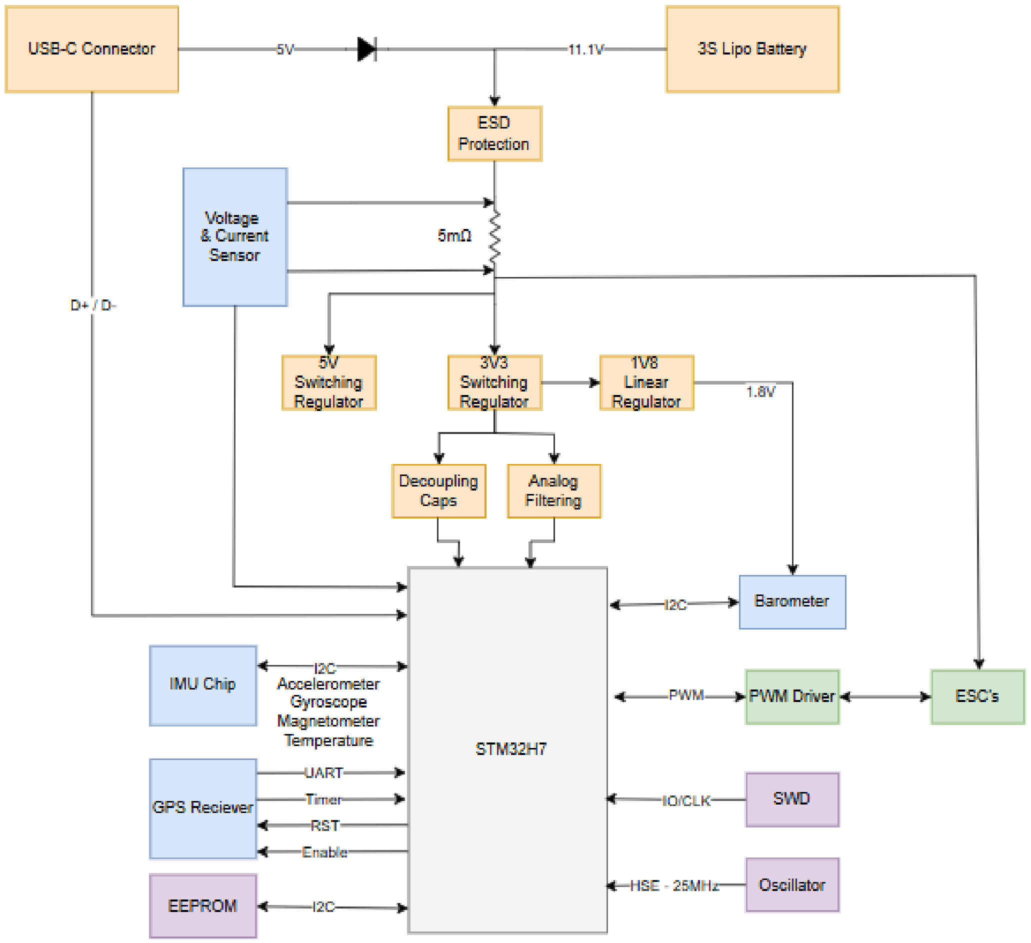


Figure 1: Block Diagram of the System.

3.0 Part Selections

The primary factor in selecting the parts for the board was cost. The team had a budget of $150, and it was essential to include as many peripherals as possible on this board to learn about the different components and how to connect them properly.

The first significant decision was the decision of the MCU. The team selected the STM32H753VIT6 microcontroller instead of the ESP32 recommended by the professor. This decision was based on higher processing power, more GPIOs, improved real-time performance, and a more developer-friendly interface. STM has its own IDE, which allows for more customization and advanced debug tools when programming, necessary for handling all the peripherals.

When looking for an IMU, Bosch Sensors were looked at due to their specifications and high ratings. A BMI088 and a BMI323 were the two IMUs mainly considered due to their popularity, as they are primarily used in drones and robotics. The BMI323 was chosen over the BMI088 because it offered more configuration options, a more detailed datasheet, and was also more cost-effective.

The regulators were chosen because they were also well-rated and had been used in alternative projects by the team, which made them an easy choice. When reviewing Digikey, no other options stood out because the datasheets for many of the regulators were inadequate, and the team did not feel confident in building a circuit for the regulator.

The other big choice was the Barometer, in which the team chose the ENS220. This decision was also an easy one for the team because Digikey did not have a wide variety of barometers. The only other options were the MPL3115A2ST1 and the KP253XTMA2, both of which were more expensive and larger in size. Given the budget and space constraints, the ENS220 was the clear choice.

4.0 Schematic Design Choices

4.1 Microcontroller

The schematic of the MCU is shown in Figure 2 below.

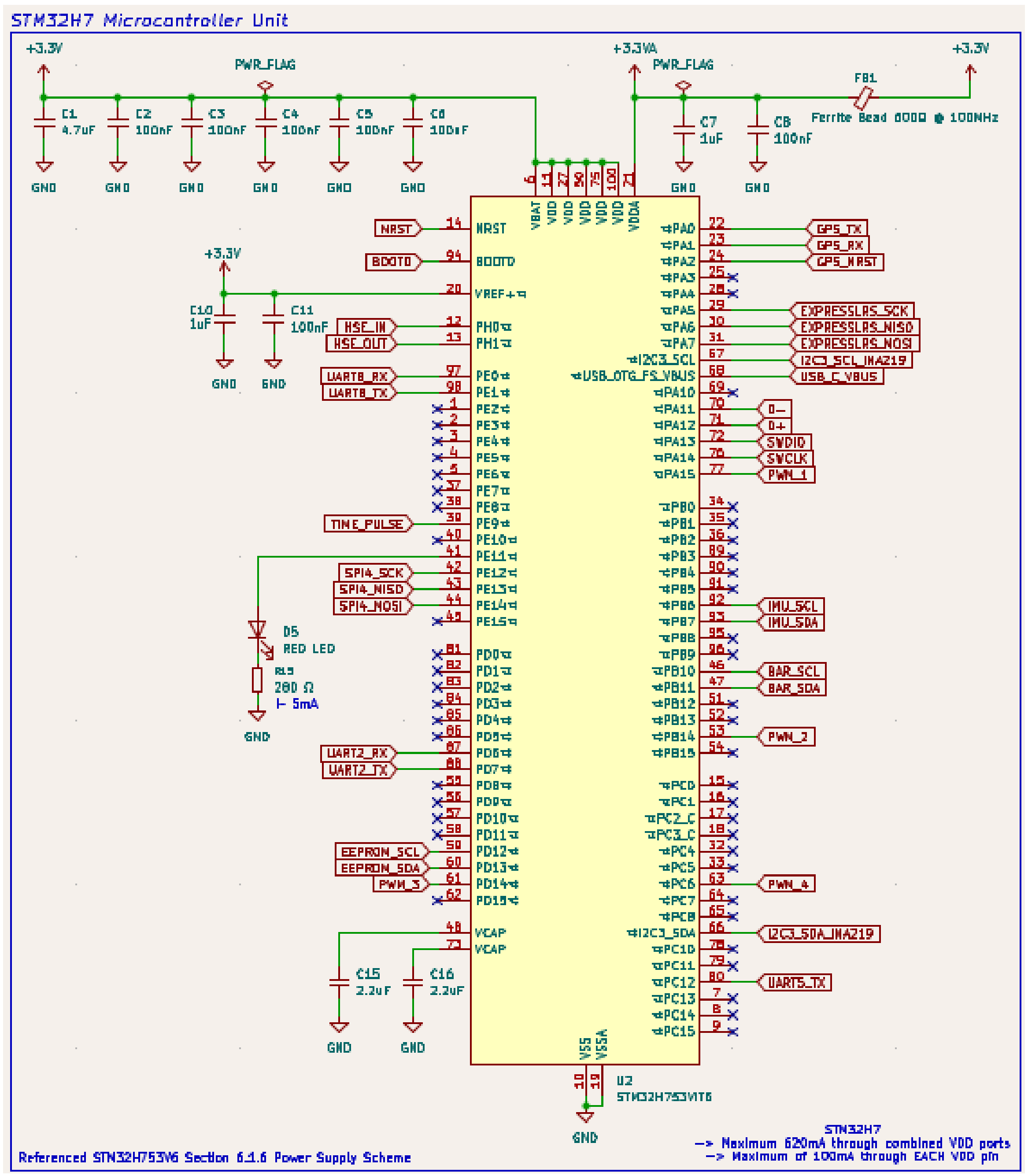


Figure 2: Schematic representation of the MCU

The decoupling network was designed according to Section 6.1.6, Figure 4 of the datasheet, to minimize noise, inductances, and prevent transient responses. A ferrite bead was added between the analog and digital supplies to filter noise on the analog voltage line. The NRST configuration follows Section 6.3.16 of the datasheet, as the pin includes an internal pull-up resistor, requiring only a capacitor to ground for reliable reset behavior. The circuit for the NRST pin is illustrated in Figure 3 below.

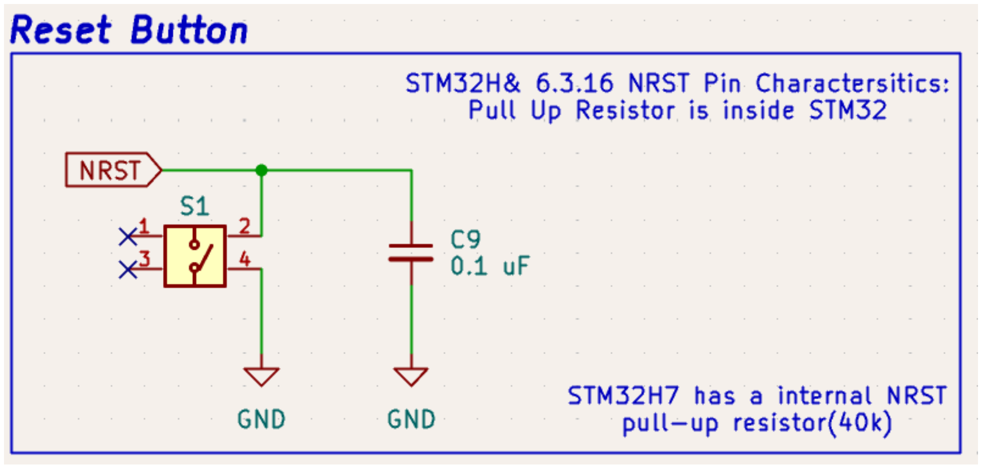

Reset Button
STM32H& 6.3.16 NRST Pin Charactersitics:
Pull Up Resistor is inside STM32
NRST
S1
1
2
3
4
C9
0.1 uF
GND
GND
STM32H7 has a internal NRST
pull—up resistor(40k)

Figure 3: Schematic Representation of NRST

For the BOOT0 pin, additional documentation from STMicroelectronics was consulted to confirm the correct configuration. The design replicates the connection diagram described in Section 4.2, Figure 4 of the STM application note, ensuring predictable startup behavior. The Boot0 circuit is illustrated in Figure 4 of the document below.

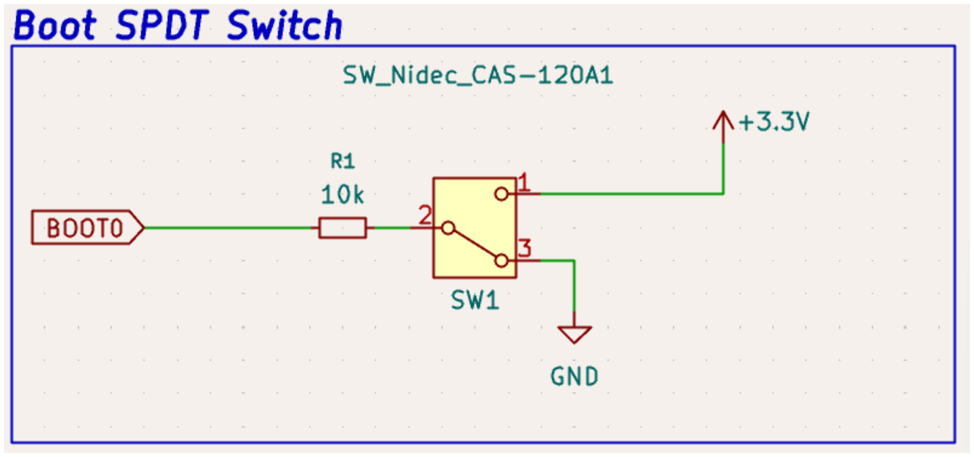


Figure 4: Schematic Representation of BOOT0

Additionally, an EEPROM circuit was included in the design to allow for non-volatile storage of data and configuration parameters, ensuring system reliability and facilitating easy reconfiguration. The EEPROM circuit is illustrated in Figure 5 of the documentation.

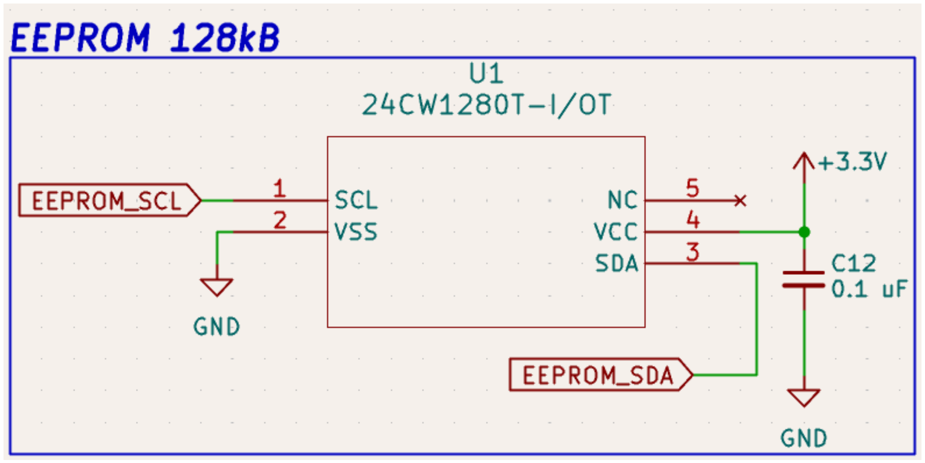


Figure 5: Schematic Representation of EEPROM

The MCU is driven by a 25 MHz crystal oscillator, providing a stable and accurate system clock. The oscillator circuit is illustrated in Figure 6 of the documentation.

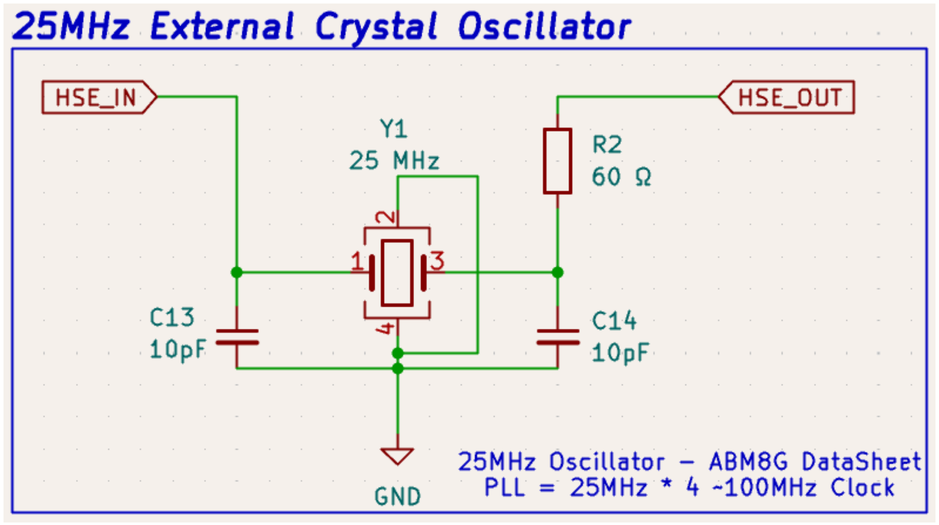

25MHz External Crystal Oscillator
HSE_IN
HSE_OUT
Y1
25 MHz
R2
60 Ω
1
2
3
4
C13
10pF
C14
10pF
GND
25MHz Oscillator - ABM8G DataSheet
PLL = 25MHz * 4 ~100MHz Clock

Figure 6: Schematic Representation of 25MHz Crystal Oscillator

4.2 I.M.U Chip

For this project, a BMI323 Chip was picked as the IMU. The main features of this chip include an accelerometer, a gyroscope, a magnetometer, and a temperature sensor. A design choice was made to use I²C data transfer. I²C was chosen because Bluetooth will be implemented in the future; SPI can conflict with Bluetooth in various instances, so I²C was deemed the safer option. The circuit was based on Section 8.1, Figure 41 in the datasheet. The team had to make the design choice on the size of the pull-up resistors. This decision was made by using the time constant equation. The maximum rise and fall times are 300 ns, and a 100 pF capacitive load was chosen; therefore, the maximum resistor value is three kΩ. The team downsized to 2200 Ohms to have some flexibility and a more common value. The final circuit for the IMU is shown in Figure 7 of the documentation.

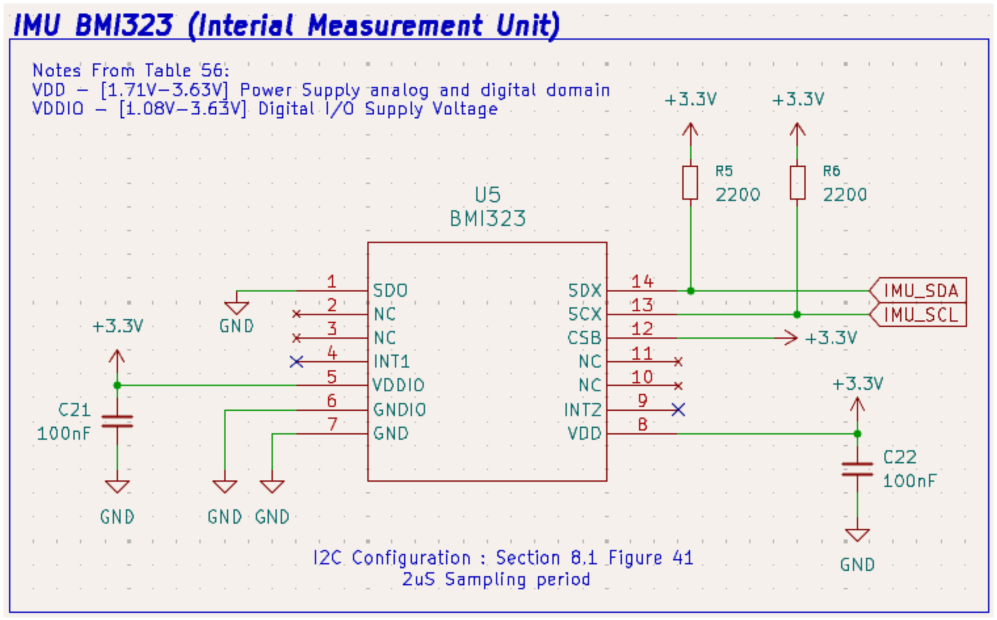

IMU BMI323 (Interial Measurement Unit)
Notes From Table 56:
VDD – [1.71V–3.63V] Power Supply analog and digital domain
VDDIO – [1.08V–3.63V] Digital I/O Supply Voltage
U5
BMI323
SDO
NC
NC
INT1
VDDIO
GNDIO
GND
SDX
SCX
CSB
NC
NC
INT2
VDD
R5 2200
R6 2200
IMU_SDA
IMU_SCL
+3.3V
C21 100nF
C22 100nF
GND
I2C Configuration : Section 8.1 Figure 41
2uS Sampling period

Figure 7: Schematic Representation of the IMU

4.3 Barometer

The team chose the ENS220 as a barometer because it seemed the most cost-effective. Again, I²C was selected to avoid SPI interference in case the team decides to add Bluetooth capabilities in the future. The barometer unit picked is powered by 1.8V, but the I2C lines can handle 3.3V capabilities. Since the barometer is the only device on the board requiring 1.8 V, a linear regulator was added to the power distribution network to step down from the standard 3.3 V rail. According to the datasheet, the ENS220 draws less than 0.1 µA of current, so there was no concern about current consumption or voltage drop. The circuit for the Barometer chip can be seen in Figure 8 of the documentation.

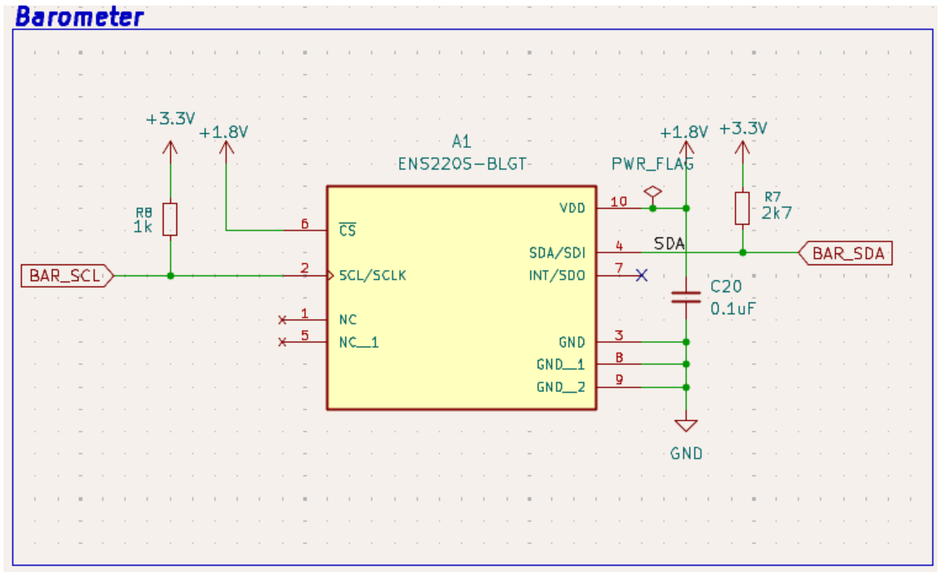


Figure 8: Schematic Representation of the Barometer

4.4 USB-C Receptacle

The team initially planned to use a battery to power the system and a USB-C for data transfer. However, due to the cost of components and the smaller scale of the board, the team decided to use USB-C for both power supply and data transfer. The final circuit for the USB-C receptacle is shown in Figure 9.

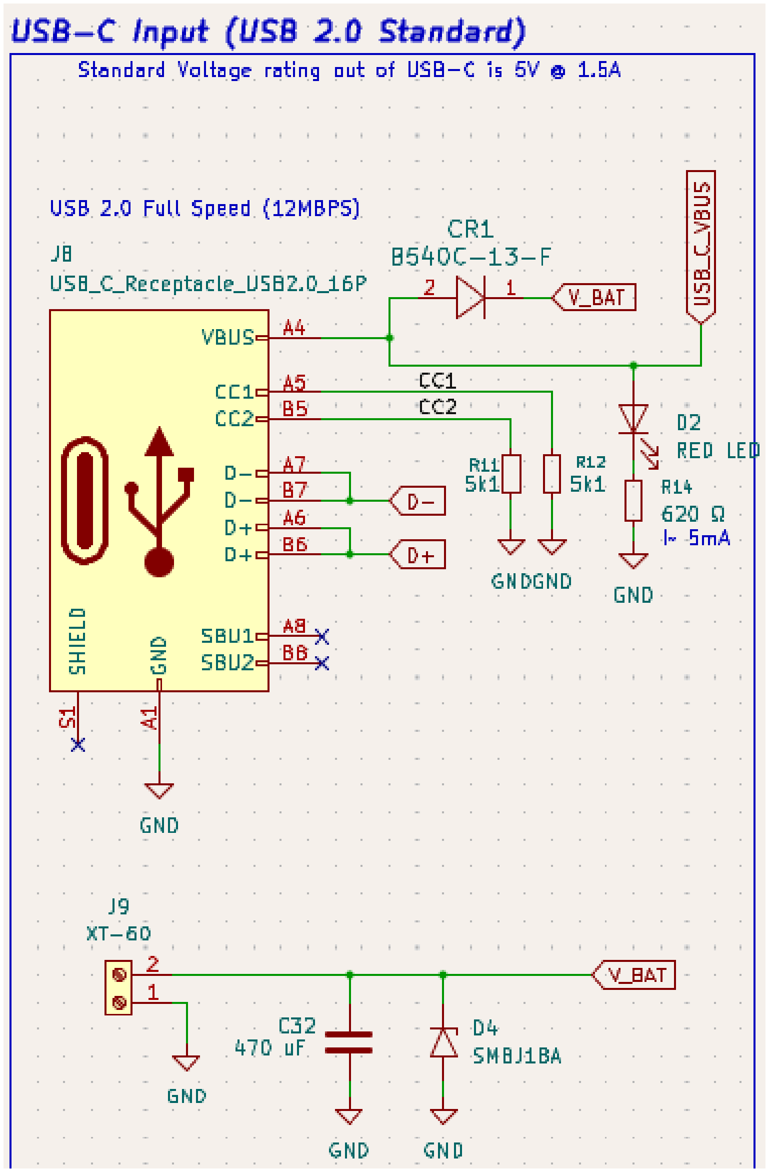


Figure 9: Schematic Representation of the USB-C

Along with the USB-C connector, ESD protection was needed since the area would be frequently connected to and disconnected from. The ESD protection circuit is illustrated in Figure 10 below.

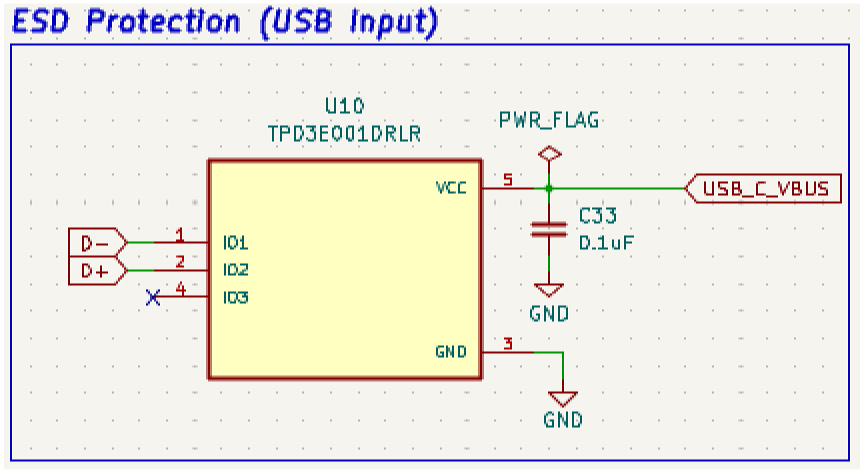
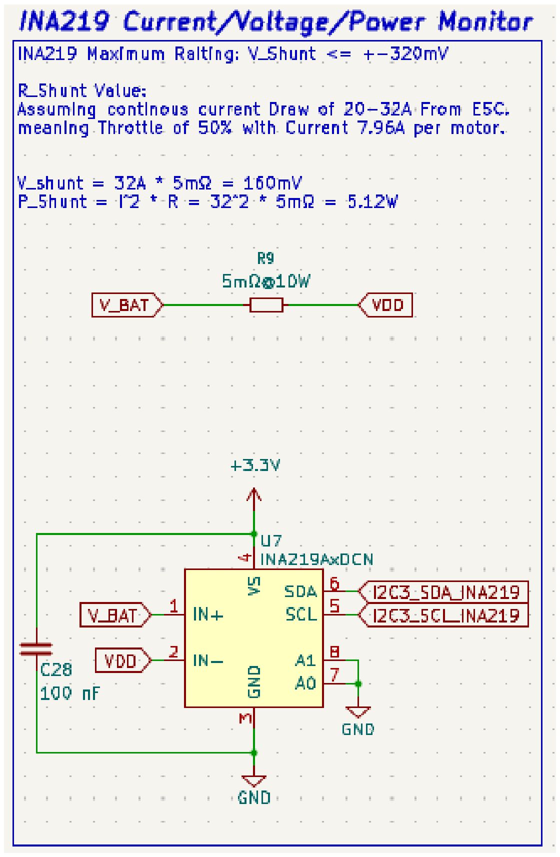


Figure 10: Schematic Representation of the ESD Protection

The team also decided to add a current/voltage/power monitor near the USB-C port. This can track the incoming current and voltage, which will be helpful in communicating battery levels to the drone's user. The circuit is illustrated in Figure 11 below.


INA219 Maximum Raiting: V_Shunt <= +-320mV

R_Shunt Value:
Assuming continous current Draw of 20-32A From ESC.
meaning Throttle of 50% with Current 7.96A per motor.

V_shunt = 32A * 5mΩ = 160mV
P_Shunt = I^2 * R = 32^2 * 5mΩ = 5.12W

Figure 11: Schematic Representation of Current/Voltage/Power Monitor

4.4 Power Distribution

All components operate at 3.3 V except for the barometer. To provide a stable supply for the system, a switching regulator (AP63203WU) was implemented to step down the 5 V USB-C input to a regulated 3.3 V, which powers the MCU and all peripherals besides the barometer. Since the barometer operates at 1.8 V, a linear regulator (MIC5365-1.8YC5-TR) was used to regulate 1.8 V from the 3.3 V supply. Decoupling capacitors were placed at supply pins to minimize voltage ripple, reduce noise, and ensure stable operation across varying load conditions. Both regulators are depicted in Figure 11 of the documentation.

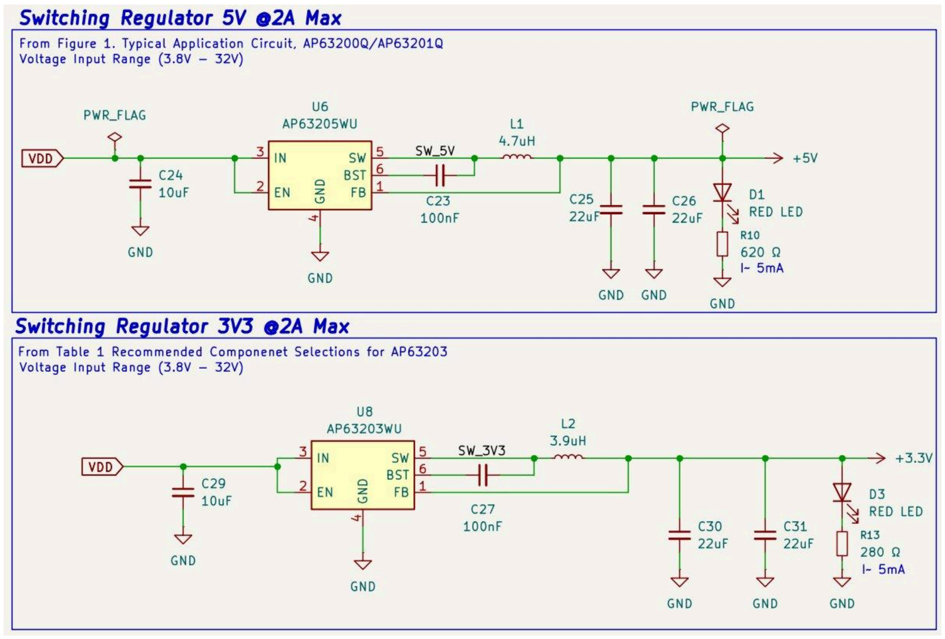

Switching Regulator 5V @2A Max
From Figure 1. Typical Application Circuit, AP63200Q/AP63201Q
Voltage Input Range (3.8V – 32V)
PWR_FLAG
VDD
U6
AP63205WU
IN
EN
GND
SW
BST
FB
SW_5V
L1
4.7uH
C23
100nF
C24
10uF
C25
22uF
C26
22uF
D1
RED LED
R10
620 Ω
I- 5mA
+5V
GND
Switching Regulator 3V3 @2A Max
From Table 1 Recommended Componenet Selections for AP63203
Voltage Input Range (3.8V – 32V)
U8
AP63203WU
SW_3V3
L2
3.9uH
C27
100nF
C29
10uF
C30
22uF
C31
22uF
D3
RED LED
R13
280 Ω
I- 5mA
+3.3V

Figure #: Schematic Representation of the Switching 5V and 3.3V Regulator

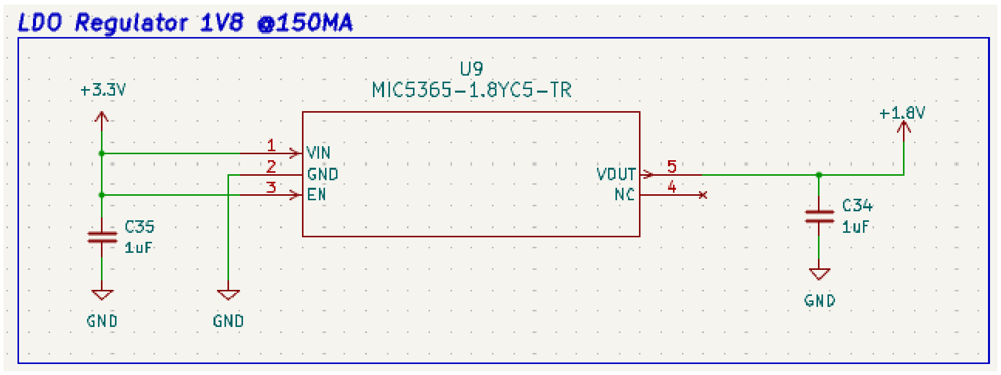


Figure 11: Schematic Representation of the LDO 1.8V Regulator

4.5 Test and Debug Points

All I²C and UART communication lines connected to onboard sensors include dedicated test points for each signal line. The VBUS from the USB-C connector and the outputs of each voltage regulator also have test points for power verification during debugging. The SWDIO and SWCLK pins were routed to male header pins, allowing for easy connection to a debugger for programming and firmware testing. Several plug-and-play connections were added through male headers, including three UART pairs and two SPI pairs to support future system expansion. Four timer outputs were routed to headers for connection to Electronic Speed Controllers (ESCs). These various plug-and-play pins are illustrated in Figure 12. These provisions allow for flexibility for testing, troubleshooting, and future feature implementation within the timeline.

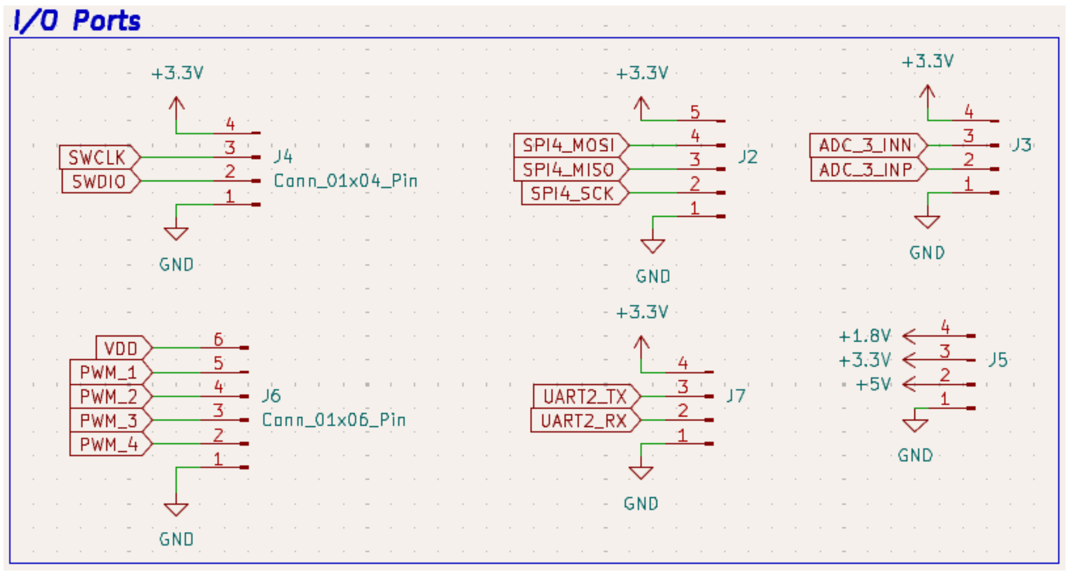


Figure 12: Schematic Representation of the I/O Ports

5.0 Layout

The goal was to design the PCB as compact as possible, since the flight controller will mount directly onto the drone, where surface area is limited. The layout process began with several rough concepts about key layout rules. Decoupling capacitors were placed as close to each MCU power pin to reduce transient voltage spikes. Test points were positioned near their associated chips to minimize the length of the signal lines. The 25 MHz crystal oscillator was placed close to the MCU, maintaining a suitable spacing from other high-speed frequency components. With a GPS and other high-frequency components, the oscillator needed to stay away from potential interferences. The first layout was inferior due to the time constraint the group encountered, as well as the design decision to have a 4-layer board. For the second layout, the team used a 6-layer board, which allowed for components to be placed on the underside of the board. By using both sides of the board, the size of the board was reduced. The layout for Revision 2 is illustrated in Figure 13 below.

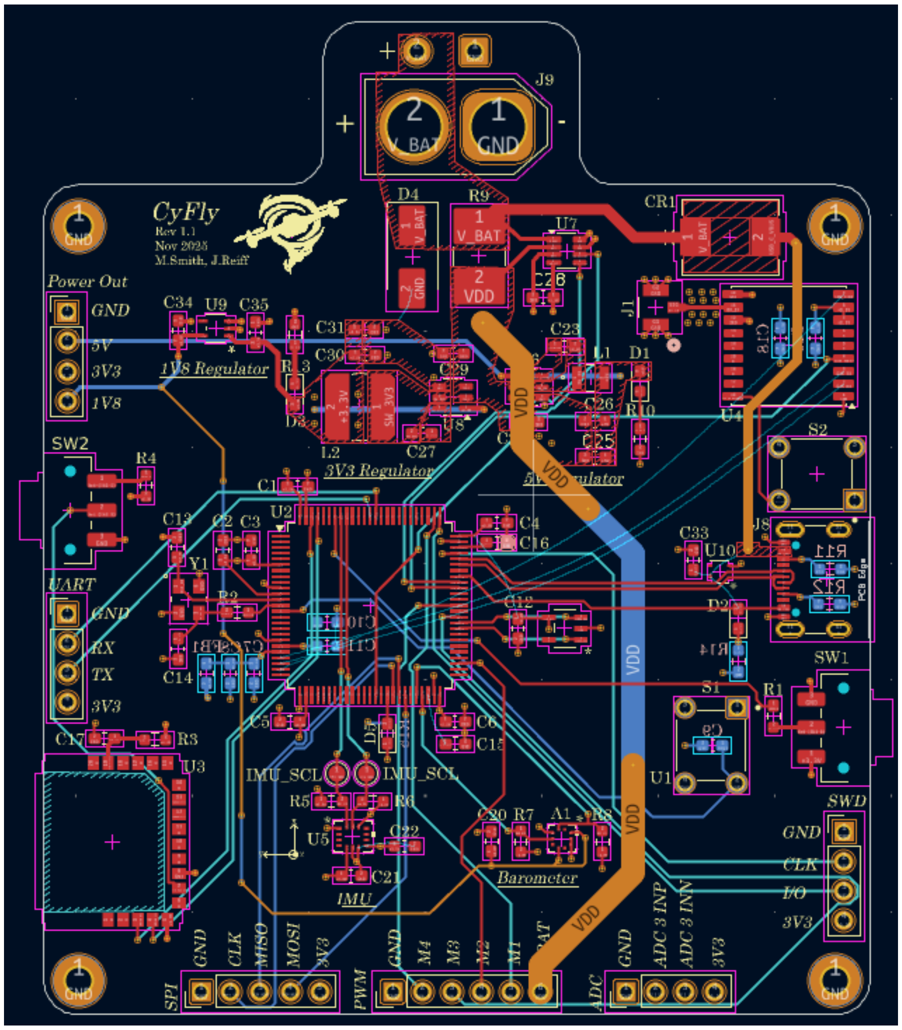


Figure 13: Layout of Revision 2

5.1 Prototype vs. Production Layout Considerations

The layout choices for this board were catered towards a prototype build, so debugging and experimentation were prioritized. If this board were to be put into production, many design choices would need to be changed. For example, the design included many header pins to facilitate easy testing and connection of multiple peripherals. Having all those pinouts would not be cost-effective or necessary. In production, the board would have a more definitive purpose and not need all the plug-and-play options, but rather just test pads for checking the functionality of peripherals. Additionally, the board could potentially be shrunk down even further to work more effectively on top of a drone. One way this could be optimized is by moving the power distribution circuit to a separate board with the ESCs, which is a typical configuration for a drone flight controller.

6.0 Revision 1 Lessons

Throughout this project, it was to be expected that there would be bumps along the way. Revision 1 provided the team with valuable information to improve Revision 2. A key finding of Revision 1 was the improper use of an inductor on the 3.3V switching regulator. A 4.7uH instead of a 3.3uH inductor was used. The 3.3V regulator powered the MCU, so without it, the entire board was unusable. The soldered revision one is shown below in Figure 14. Additionally layout was rushed in this revision, shown with the disorganization with our pinouts and surface area used.

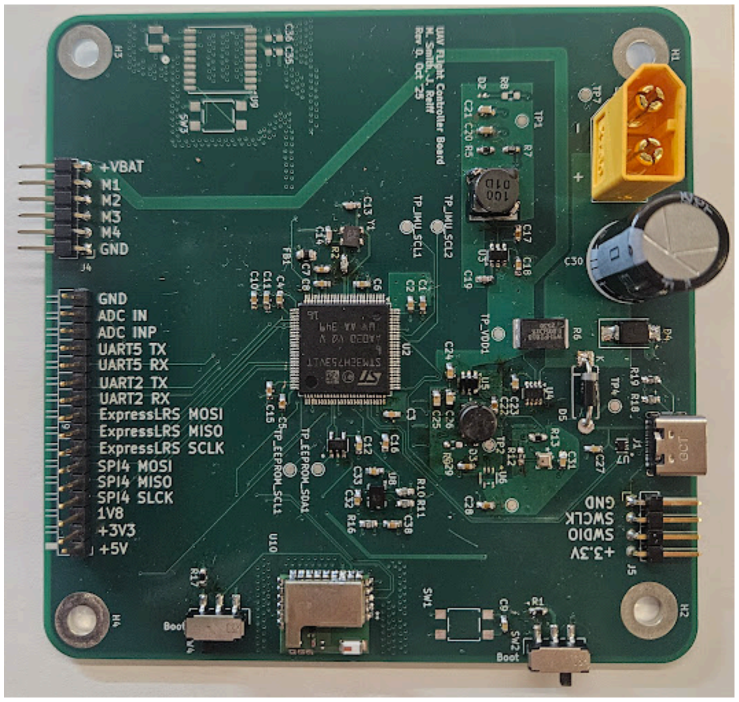


Figure 14: Revision 1 of the board built out

7.0 Revision 2 Lessons

Revision 2 brought its own set of learning opportunities. Fortunately, all the regulators worked right away on this board. However, the team was unable to establish a connection with the MCU. Unfortunately, this resulted in a lengthy debugging process. Initially, the team noticed that a 3.3V pin and a ground pin on the MCU were bridged together, which led to the conclusion that the MCU might be shorted. The team removed the MCU, just to find that it was not the problem. Before installing a new MCU, the problem needed to be identified and tracked down. A thermal test was conducted in order to locate the short. The voltage source was connected with a current output limit, and the team checked to see which components were overheating. The IMU was found to be quite warm, so the team removed it and replaced it with a new IMU. Once the IMU was replaced, the MCU was also replaced, and the board functioned as expected with

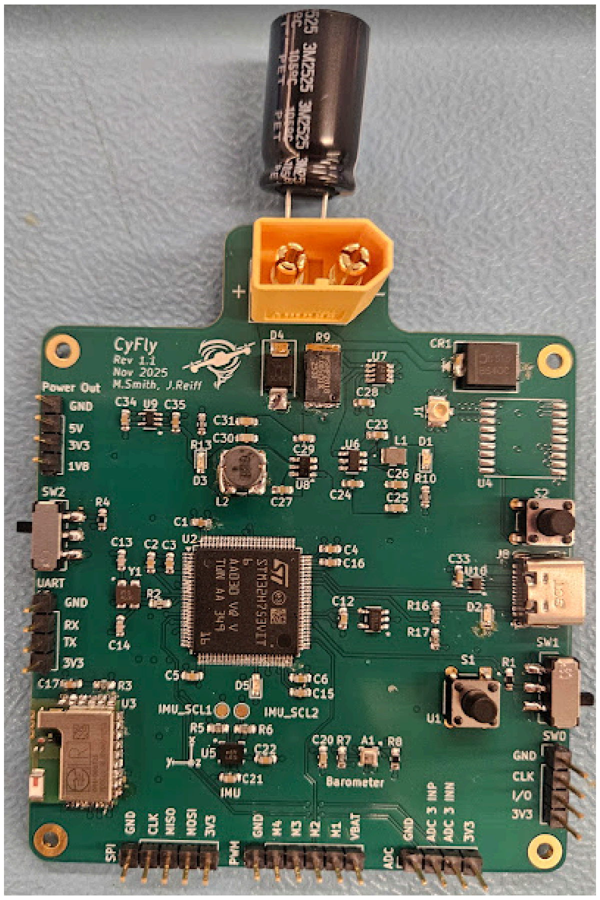


Figure 15: Revision 2 of the board built out

8.0 Testing the Board

The board testing procedure was conducted in a systematic manner to verify proper functionality and prevent damage to components. Testing began with validation of the onboard voltage regulators, as failure in any regulator could damage other components on the board. Test points connected to header pins were provided for each regulator, enabling easy verification of functionality. After confirming correct regulator operation, communication with the MCU was established, followed by testing of peripheral devices. Each step of the testing process is described in detail in the following subsections.

8.1 Testing Regulators

Testing began with the voltage regulators. The bench power supply was used to supply the board with 5V at the battery terminal, with a current limit of 0.5 Amps. Limiting the current keeps shorts from pulling too much current and damaging the board or components. A picture of the setup is shown below in Figure 16. With revision 2, all regulators performed as expected, and photos of the outputs are shown in Figure 17. With all the regulators working, the team was able to move on to establishing a connection with the MCU.

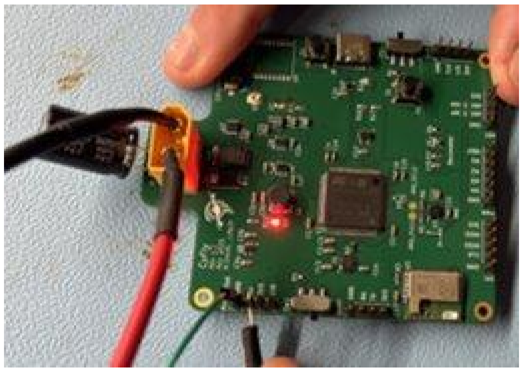


Figure 16: Testing setup for the voltage regulators.

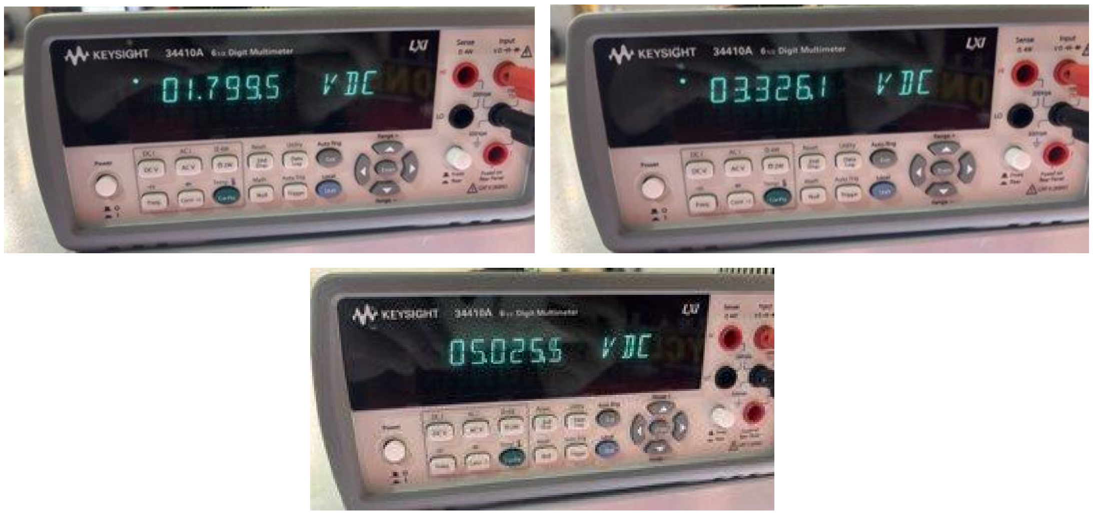
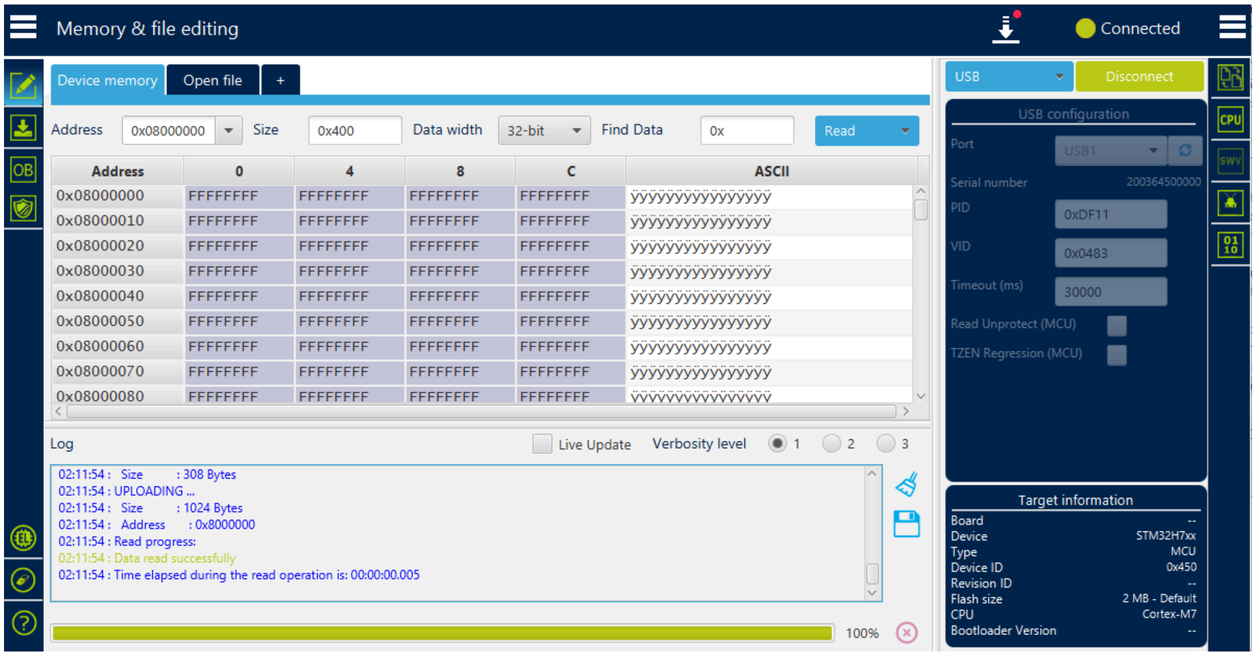

KEYSIGHT 34410A 6½ Digit Multimeter
01.7995 VDC
KEYSIGHT 34410A 6½ Digit Multimeter
03.3261 VDC
KEYSIGHT 34410A 6½ Digit Multimeter
05.0255 VDC

Figure 17: Successful outputs of the three regulator test points.

8.2 MCU Connection

For initial MCU programming, STM32CubeProgrammer was used to flash the compiled .elf file onto the device. Upon connecting the board via USB, the application successfully detected the MCU and completed the programming process without errors. Confirmation of a successful upload is shown in Figure 18.



Address 0x08000000 Size 0x400 Data width 32-bit Find Data 0x

Address	0	4	8	C	ASCII
0x08000000	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x08000010	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x08000020	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x08000030	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x08000040	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x08000050	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x08000060	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x08000070	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x08000080	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ

Log — Verbosity level [✓ 1] 2 3

```
02:11:54 :  Size        : 308 Bytes
02:11:54 : UPLOADING ...
02:11:54 :  Size        : 1024 Bytes
02:11:54 :  Address      : 0x8000000
02:11:54 : Read progress:
02:11:54 : Data read successfully
02:11:54 : Time elapsed during the read operation is: 00:00:00.005
```

100%

USB

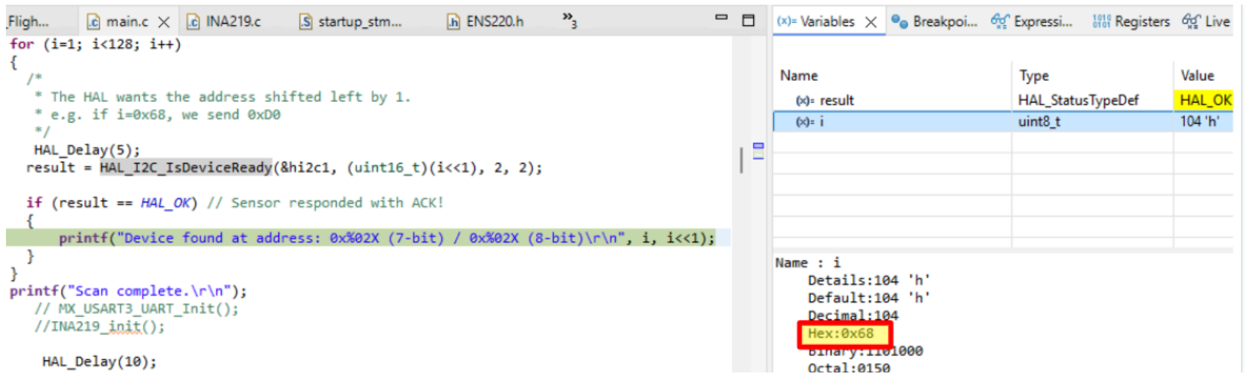
Port	USB1
Serial number	200364500000
PID	0xDF11
VID	0x0483
Timeout (ms)	30000
Read Unprotect (MCU)	☐
TZEN Regression (MCU)	☐

Board	--
Device	STM32H7xx
Type	MCU
Device ID	0x450
Revision ID	--
Flash size	2 MB - Default
CPU	Cortex-M7
Bootloader Version	--

Figure 18: Successful .elf file loaded onto the board.

8.3 BMI323 Connection

The IMU was a little trickier to set up because it required writing a peripheral file to interact and send data between the BMI and the MCU. The STMCubeIDE set up the I2C protocol through a UI, which made that part easy. The user manual stated that the device's I2C address was hex 0x68. To verify this, a loop was created in the code to send a signal and look for an I2C acknowledgement bit from multiple addresses. An acknowledgement bit was found at address 0x68, as shown in Figure 19 below.



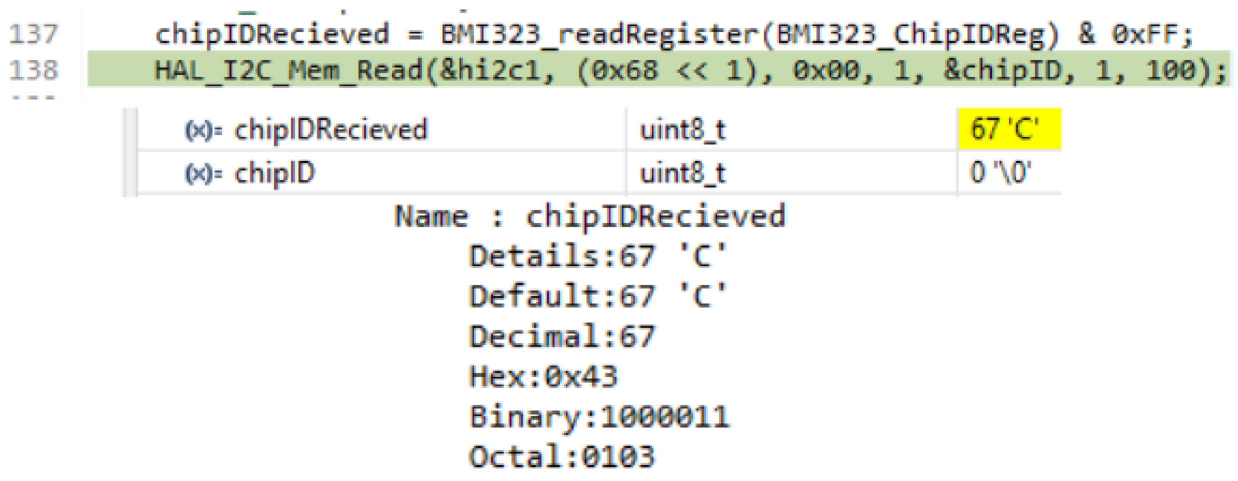
```
for (i=1; i<128; i++)
{
  /*
   * The HAL wants the address shifted left by 1.
   * e.g. if i=0x68, we send 0xD0
   */
   HAL_Delay(5);
  result = HAL_I2C_IsDeviceReady(&hi2c1, (uint16_t)(i<<1), 2, 2);

  if (result == HAL_OK) // Sensor responded with ACK!
  {
      printf("Device found at address: 0x%02X (7-bit) / 0x%02X (8-bit)\r\n", i, i<<1);
  }
}
printf("Scan complete.\r\n");
   // MX_USART3_UART_Init();
   //INA219_init();

    HAL_Delay(10);
```

Name	Type	Value
(x)= result	HAL_StatusTypeDef	HAL_OK
(x)= i	uint8_t	104 'h'

```
Name : i
    Details:104 'h'
    Default:104 'h'
    Decimal:104
    Hex:0x68
    Binary:1101000
    Octal:0150
```

Figure 19: Successful I2C Connection to BMI323's address.

Once the device was verified to be present, another check was conducted to verify its validity. According to the BMI323 user manual, the device's unique chip ID is located at address 0x0 and has a value of 0x43. A line of code was written to check that, and the successful Chip ID reading is shown below in Figure 20.



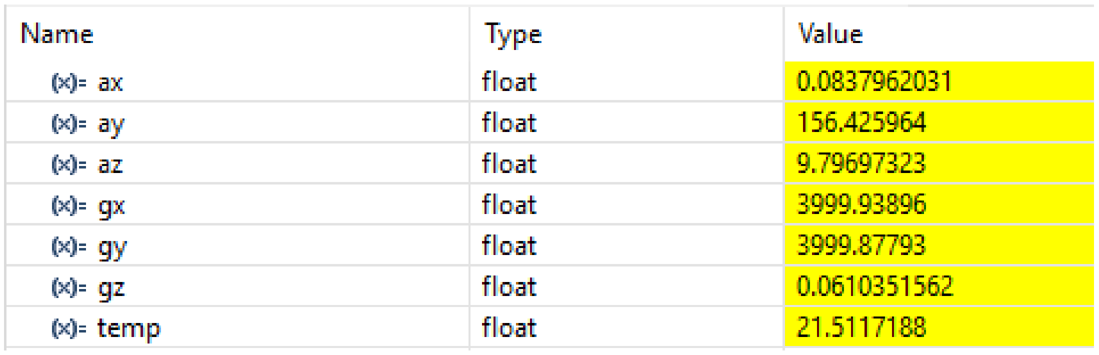
```
    chipIDRecieved = BMI323_readRegister(BMI323_ChipIDReg) & 0xFF;
    HAL_I2C_Mem_Read(&hi2c1, (0x68 << 1), 0x00, 1, &chipID, 1, 100);
```

(x)= chipIDRecieved	uint8_t	67 'C'
(x)= chipID	uint8_t	0 '\0'

```
Name : chipIDRecieved
    Details:67 'C'
    Default:67 'C'
    Decimal:67
    Hex:0x43
    Binary:1000011
    Octal:0103
```

Figure 20: Successful BMI323 Chip ID Reading.

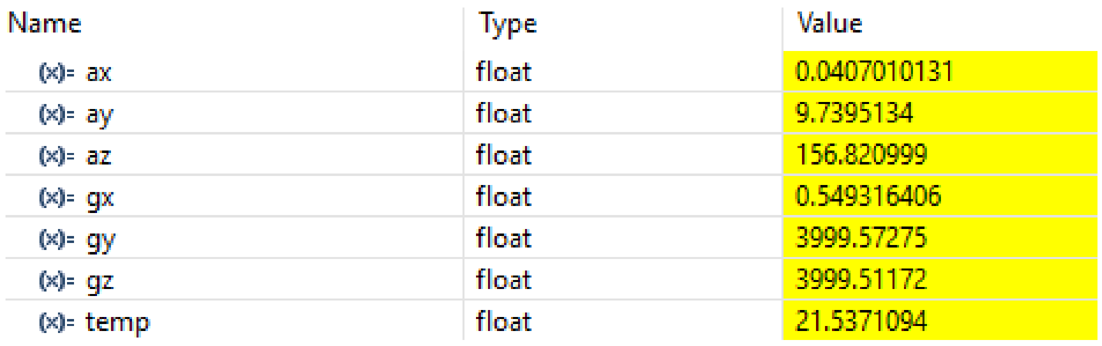
The accelerometer and gyroscope were both configured to run at a 400kHz sampling rate, and an averaging of 8 samples. Once they were configured, data values were able to be read from each. The first time data was collected, the board was sitting on a flat surface, which resulted in the expected acceleration of 9.81 in the z-direction, as observed by the team in Figure 21 below.



Name	Type	Value
(x)= ax	float	0.0837962031
(x)= ay	float	156.425964
(x)= az	float	9.79697323
(x)= gx	float	3999.93896
(x)= gy	float	3999.87793
(x)= gz	float	0.0610351562
(x)= temp	float	21.5117188

Figure 21: BMI323 data from the board on a flat surface.

The board was then turned on its side to see if the values shifted. As shown below in Figure 22, the values did precisely that. The 9.81 moved to the y direction. Also, in both Figures, the temperature is shown to be 21.5 °C, which is approximately 70°F. That is quite accurate because the thermostat was also set to 68°F.

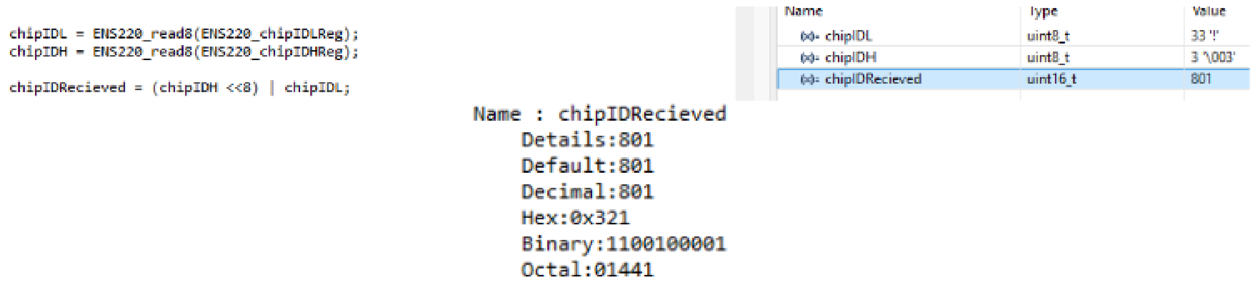


Name	Type	Value
(x)= ax	float	0.0407010131
(x)= ay	float	9.7395134
(x)= az	float	156.820999
(x)= gx	float	0.549316406
(x)= gy	float	3999.57275
(x)= gz	float	3999.51172
(x)= temp	float	21.5371094

Figure 22: BMI323 data from the board turned on its side.

8.4 ENS220 Connection

The ENS220 differs from the BMI in that it has 8-bit registers, whereas the BMI has 16-bit registers. Also, the ENS220 does not have any dummy bytes, whereas the BMI has 2. This required the team to create a new library in order to configure and communicate seamlessly with the ENS220. The exact process was used to establish connectivity between the ENS220 and the MCU. The I2C address was found to be 0x20, which matches the information stated in the manual. The manual also noted that the chip ID would be 0x321, which was read from the chip ID register, as shown in Figure 23.



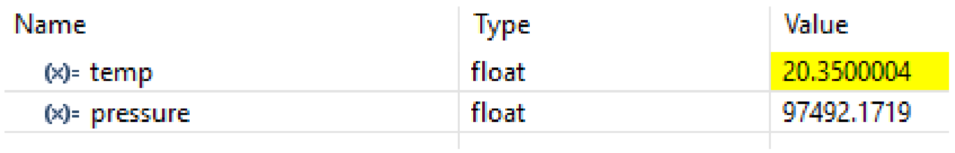
```
chipIDL = ENS220_read8(ENS220_chipIDLReg);
chipIDH = ENS220_read8(ENS220_chipIDHReg);

chipIDRecieved = (chipIDH <<8) | chipIDL;
```

Name	Type	Value
chipIDL	uint8_t	33 '!'
chipIDH	uint8_t	3 '\003'
chipIDRecieved	uint16_t	801

```
Name : chipIDRecieved
    Details:801
    Default:801
    Decimal:801
    Hex:0x321
    Binary:1100100001
    Octal:01441
```

Figure 23: ENS220 Chip ID Reading.

The data was read and translated from the sensor, and the values are shown below in Figure 24. Again, the temperature is in Celsius and equates to 68 degrees Fahrenheit, which is highly accurate. The pressure is also within a valid range, with an expected pressure of 97,000 pascals.



Name	Type	Value
temp	float	20.3500004
pressure	float	97492.1719

Figure 24: ENS220 Collected Values.

9.0 Preventing Issues in Future Builds

The major takeaway for the team is to read the data sheets multiple times. The team selected devices with robust datasheets to avoid the headache of confusing jargon. However, they neglected to pay close attention to those datasheets, which caused significant problems with the 3.3V regulator on the revision one board and the crystal oscillator.

10.0 Conclusion

In conclusion, this project presents the implementation of a flight controller based on the STM32H7 microcontroller. While the STM32H7 offered more processing power than what was necessary for this application, utilizing it provided more exposure to the design flows required in modern embedded systems design used in industry, and additionally allowed us to know the fundamentals of hardware design for Embedded system applications, through the process of power regulation, MCU supporting networks, and sensor implementation.

This project enabled us to apply theoretical topics covered in the classroom to our design. This included designing a Switching and Linear voltage regulation for our system, as well as addressing MCU requirements such as decoupling networks, NRST, Boot Switches, and Analog voltage filtering. Then, understand the ppm requirements for a given oscillator used for our clock. Finally, learning about designing RF Layout, including calculating our critical length for our GPS Module, and Impedance matching both RF lines.