Preparation of Project Technical Reports for the LASC

Team 15 (SIGHT CubeSat) Project Technical Report for the 2023 LASC

Han-Kun Chen[1] , Yin-Rui Su, Ching-Jui Kuan
Department of Space Science & Engineering, National Central University
No. 300, Zhongda Rd., Zhongli District, Taoyuan City 320317, Taiwan (R.O.C.)

Abstract

The SIGHT cube satellite is an amateur cube satellite made by the students of the Department of Space Science and Engineering of Central University. It can detect a variety of tropospheric scientific data. This article aims to introduce the introduction of the various systems of the satellite, including structure, Power, scientific payload, and GPS recovery subsystems. In addition, there is also a functional test plan and functional test results of the subsystems.

I. Introduction

A. Structure Subsystem (STR)
The structure subsystem is used to support the overall structure. It is made of Al 6061.

B. Electrical Power Subsystem (EPS)
The Electrical Power Subsystem power the CubeSat. EPS consists of three lithium batteries, a power distribution board, and switches.

C. Payload

1) MPU6050

Measuring three-axis acceleration and angular velocity data.

2) BMP280

Measuring barometric pressure and temperature data, and calculates relative altitude changes based on pressure variations.

3) DHT22

Measuring temperature and humidity data.

4) NEO-6m GPS Tracker (GPS1)

Receiving the longitude and latitude data of the CubeSat.

5) Pi Camera

Recording videos during CubeSat's flight.

D. GL300 Real Time GPS Tracker (GPS2)
To enable real-time transmission of the satellite's current position, allowing us to track and recover the satellite after landing.

E. On Board Data Handling System
Our On-board computer consists of two parts, Raspberry Pi 3 Model A+ and Raspberry Pi Zero 2W. Raspberry Pi 3 Model A+ is responsible for capturing images using the Raspberry Pi Camera, while Raspberry Pi Zero 2W handles other sensors, including NEO-6M-0-001, MPU6050, BMP280, and DHT22.

Upon booting, the system will automatically execute the program, initiating data measurements from all sensors and video recording from the Pi Camera. The files will be stored on the SD card until the CubeSat loses power.

[1]Kevin91chen@gmail.com Han-Kun Chen

F. Data processing
After the CubeSat lands, we read the data stored in the SD card, and use MATLAB to consolidate the data and save it into a new txt file.

For the Inertial data, it needs to be calculated using trigonometric functions, utilizing the original three-axis acceleration and angular velocity data to obtain the CubeSat's three-axis attitude angles over time.

II. System Architecture Review

A. Structure Subsystem (STR)
The entire structural sub-system is made of Al6061, including four side panels of about 200mm*100mm*5mm and four partitions of 90mm*90mm*5mm to support and separate each subsystem inside.

The production of the structural subsystem is first modeled by Fusion 360, and then converted into a tool path and then input to Datron Neo CNC for processing.

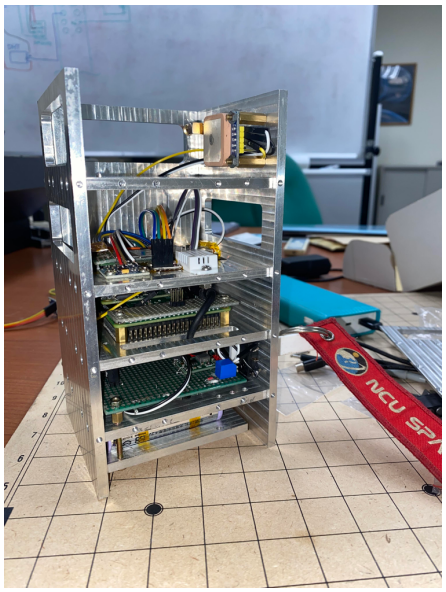


Image.1 SIGHT CubeSat

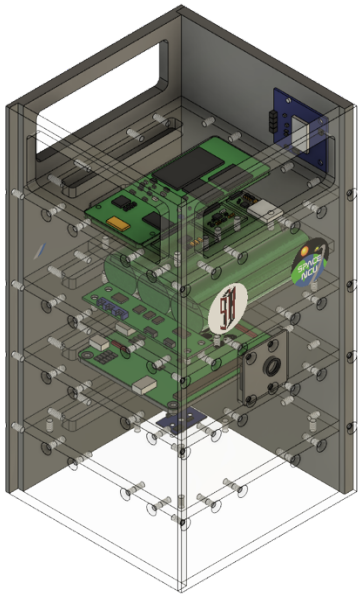


Image.2 3D model

B. Electrical Power Subsystem (EPS)
The Electrical Power System of SIGHT CubeSat uses 18650 Li-ion batteries produced by Kinyo company, with a specification of DC 3.7V 2350mAh and a weight of 33g. It is expected to be powered by four batteries connected in series.

For battery charging and voltage regulation, a TP4056 is planned to be connected in the circuit. The batteries will be charged before launch and during the functional testing phase. TP4056 can be charged through a Type C connector, with a charging current up to 1A and input voltage of 5A. Its size is 0.5*2.6*1.7CM, and the Type C USB socket and the + - solder pad next to it are the power input terminals, connected to 5V voltage. B+ is connected to the positive pole of the lithium battery, and B- is connected to the negative pole of the lithium battery. OUT+ and OUT- are connected to the load. The red light indicates that it is charging, and the blue light indicates that it is fully charged.

For power output, a self-designed PCB will be used, which contains 2 sets of small transformers that can provide 3.3V, 5V, power supply.

In the work of voltage transformation and voltage stabilization, we use MP1584, which is a small voltage transformation module that can receive an input voltage of 0.7V~28V

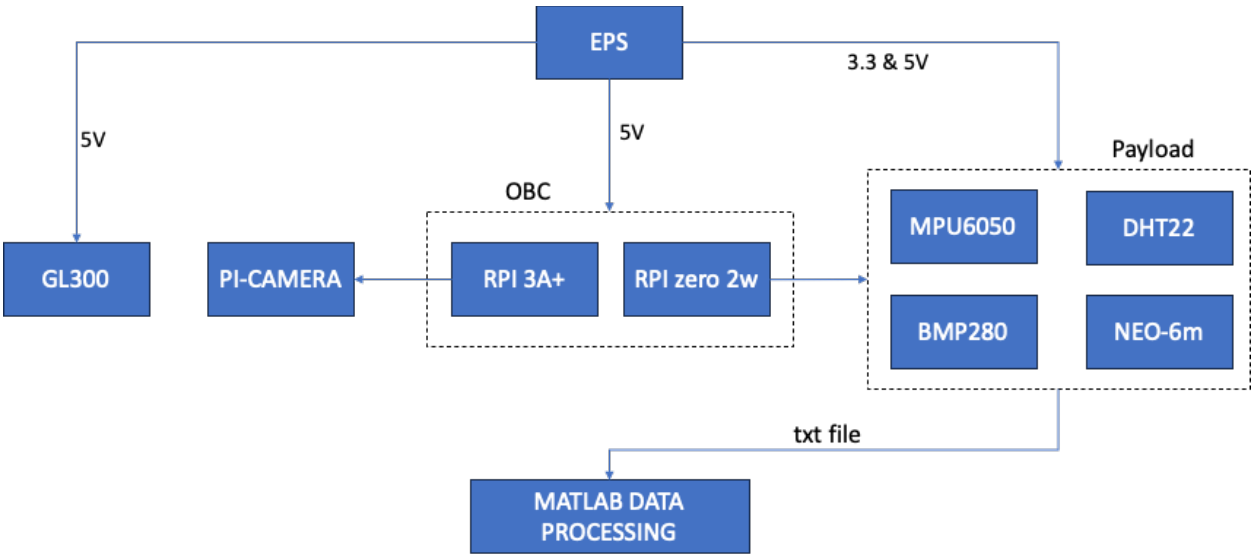


Figure1.Block diagram

C. Payload

1) MPU6050

Inertial measurement unit, provide 3-Axis Accelerometer & Gyro data

- Digital interface: I2C communication protocol (SCL, SDA)
- Gyroscope range: ±250 / 500 / 1000 / 2000°/s
- Accelerometer range: ±2 / ±4 / ±8 / ±16g
- Dimensions : 21.2 x 16.4 x 3.3mm
- Supply voltage : 3.3V (2.375 V ~ 3.46 V)
- Track record Cycle: 0.2 sec

Accelerometer measures the inertial force, assuming CubeSat is only influenced by gravity, it will measure gravity as inertial force pointing to the ground. The total acceleration vector always points to the opposite direction of the ground, then the attitude angles of the CubeSat relative to Earth can be obtained.
Gyroscope measures angular velocity, obtaining the variation of the attitude angles over time.

2) BMP280

provided temperature and barometric pressure data

- Digital interface : I2C communication protocol (SCL,SDA)
- Temperature range : -40 to 85(±0.5)°C
- Pressure range : 300 – 1100(±1.0) hPa
- Dimensions: 11.5 x 15mm
- Supply Voltage: 3.3V or 5V
- Track record Cycle: 1 sec

Altitude calculation method: using changes in air pressure to calculate altitude
Model used in the troposphere:

$$P = P_b \left[\frac{T_b+(h-h_b)L_b}{T_b}\right]^{\frac{-g_0 M}{R*L_b}}$$ ----(formula#1)

It requires measuring the reference sea level pressure to obtain changes in air pressure and then calculate the altitude.However, since we cannot access real-time sea-level pressure data, we will use the ground as a reference point and measure the relative altitude change.

- Altitude accuracy: ±1 meter

3) DHT22

provided Humidity and temperature(data verified with BMP280)

- Digital Interface: GPIO
- Relative Humidity Range: 0 to 100% (±2.0% accuracy)
- Temperature Range: -40 to 80°C (±0.5 accuracy)
- Track record Cycle: 1 sec

4) NEO-6M GPS Tracker (GPS1)

It is responsible for receiving the longitude and latitude of the satellite, and after receiving them, the longitude and latitude will be recorded to the Raspberry Pi's SD card at a frequency of one record per second. The longitude and latitude will be stored separately as lat.txt and lng.txt, respectively.

- Power Supply Voltage (type): VCC = 5.0 V
- Antenna Gain: 50 dB
- Receiver Chain Noise Figure: 50 dB
- Operating Temperature: -40 to 85 °C
- Horizontal Position Accuracy: 2.5 m
- Sensitivity:
  I. Tracking & Navigation: -161 dBm
  II. Reacquisition: -160 dBm
  III. Cold Start (without aiding): -147 dBm
  IV. Hot Start: -156 dBm
- Track record Cycle: 1 sec

5) Pi Camera

The Pi Camera will record video data during the flight and store it in .h264 file format. The storage interval will be every one minute, meaning that a new video file will be created and saved every minute.

- Resolution: 1920*1080
- Frame rate: 30

D. GL300 Real Time GPS Tracker (GPS2)

This module allows us to view the current position of the satellite using a mobile phone. Finally, after the satellite lands, we can retrieve the SD card from the Raspberry Pi by using the GPS location sent back to the mobile phone.

- Power supply Voltage: VCC = 3.7 V
- Operating Temperature : -20°C ~ 55°C
- RF connector: MMCX for external GPS antenna

E. On Board Data Handling System

1) On board computer:

Raspberry Pi 3 Model A+ for Raspberry Pi Camera
Raspberry Pi zero 2w for other sensor

2) Specification

- CPU of Raspberry Pi 3 Model A+ : Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- CPU of Raspberry Pi zero 2w : Broadcom BCM2710A1(RP3A0), Arm Cortex-A53 @ 1GHz+
- GPIO : HAT-compatible 40 pin I/O header footprint
- camera port for connecting a Raspberry Pi Camera
- data storing : Micro SD (512MB LPDDR2)
- 5V/2.5A DC power input

F. Satellite specification sheet

Component name	Size	Current	Voltage	Number of pins	Special requirements
Raspberry Pi Camera	3.69 × 2.81 mm	0.2A	5V	15	CSI-2 serial data output.
EPS board	35mm*10mm			None	
NEO-6M-0-001	38.9*25.2*10mm	50mA	5V	1	Tx

MPU6050(IMU)	21.2*16.4*3.3mm	3.8mA	3.3V	2	sda,scl
BMP280	11.5*15mm	1mA	3.3V	2	sda,scl
DHT22	40*23mm	1.5mA	3.3V	1	Backup senser

Table1. Satellite specification sheet

G. Pinout diagram

I/O：DHT22→ Physical Pin7

UART(Rx)：NEO-6M-0-001→ Physical Pin10

I2C：MPU6050、BMP280→Physical Pin3,5(2 slave connect on I2C)

camera：Raspberry Pi Camera→15-Pin Connector

H. Program

To enable automatic execution of programs during startup, we edit the "autostart" file on the Raspberry Pi.
On the Raspberry Pi Zero 2W, three programs will be automatically executed:

- gpspress_print.py: This program reads GPS longitude, latitude, and altitude, and plots the three-dimensional motion path along with barometric, temperature and humidity data.
- gpsprint.py: This program only reads GPS longitude and latitude. In case altitude data is missing, it provides two-dimensional coordinate information.
- mpu6050_raw.py: This program reads the three-axis acceleration and angular velocity data measured by the MPU6050.

On the Raspberry Pi 3 Model A+, the program "camera.py" will be automatically executed.
It will continuously record videos after startup, until the satellite loses power, and it will save a file every minute.

1) gpspress_print.py

```
import serial
import pynmea2
import os
import time
from bmp280 import BMP280
from smbus import SMBus
from datetime import datetime
# clear previous data in data,txt
if os.path.exists("latdata.txt"):
   os.remove("latdata.txt")
if os.path.exists("lngdata.txt"):
   os.remove("lngdata.txt")
if os.path.exists("g_timedata.txt"):
   os.remove("g_timedata.txt")
if os.path.exists("temp.txt"):
   os.remove("temp.txt")
if os.path.exists("press.txt"):
   os.remove("press.txt")
if os.path.exists("alt.txt"):
   os.remove("alt.txt")
file1 = open('latdata.txt','w')
file2 = open('lngdata.txt','w')
file3 = open('g_timedata.txt','w')
file4 = open('temp.txt','w')
file5 = open('press.txt','w')
file6 = open('alt.txt','w')
bus = SMBus(1)
bmp280 = BMP280(i2c_dev=bus)
data_count = 0
while True:
```

```
    port="/dev/ttyAMA0"
    ser=serial.Serial(port, baudrate=9600, timeout=0.5)
    dataout = pynmea2.NMEAStreamReader()
    newdata=ser.readline()
    n_data=newdata.decode("latin-1")
    # get current time
    current_time = datetime.now()
    # turn current_time into string
    milliseconds = current_time.microsecond // 1000
    formatted_time = current_time.strftime("%Y.%m.%d_%H:%M:%S") + ".{:03d}".format(milliseconds)
    temperature = bmp280.get_temperature()
    pressure = bmp280.get_pressure()
    altitude = bmp280.get_altitude()
    format_temp = "{:05.2f}".format(temperature)
    format_pressure = "{:05.2f}".format(pressure)
    format_altitude = "{:05.2f}".format(altitude)
    print(format_temp + "  " + format_pressure + "  " + format_altitude)
    if n_data[0:6] == "$GPRMC":
        newmsg=pynmea2.parse(n_data)
        lat=newmsg.latitude
        lng=newmsg.longitude
        lat1=round(lat,4)
        lng1=round(lng,4)
        gps = "Latitude="+str(lat1)+", Longitude="+str(lng1)+", Time="+formatted_time
        print(gps)
        file1.write(str(lat1))
        file1.write("\n") #change row
        file2.write(str(lng1))
        file2.write('\n')
        file3.write(str(formatted_time))
        file3.write("\n")
        file4.write(str(format_temp))
        file4.write("\n")
        file5.write(str(format_pressure))
        file5.write("\n")
        file6.write(str(format_altitude))
        file6.write("\n")
    data_count +=1
    if data_count ==3:
        file1.flush()
        file2.flush()
        file3.flush()
        file4.flush()
        file5.flush()
        file6.flush()
        os.fsync(file1.fileno())
        os.fsync(file2.fileno())
        os.fsync(file3.fileno())
        os.fsync(file4.fileno())
        os.fsync(file5.fileno())
        os.fsync(file6.fileno())
        data_count = 0
    time.sleep(1.5)
file1.close()
file2.close()
file3.close()
```

```
file4.close()
file5.close()
file6.close()
```

Description:
By reading the $GPRMC data returned by NEO-6m to the Raspberry Pi, we can obtain the longitude and latitude information. And read data from BMP280 and DHT22 to the Raspberry Pi, providing barometric pressure, temperature, and humidity data. Finally, it saves the longitude, latitude, altitude, barometric pressure, temperature, humidity, and corresponding timestamp data into seven separate TEXT files. To prevent data loss in case of power turn off after the CubeSat lands, the code saves data after every three readings, ensuring data integrity during power turn off.

2) gpspress_print.py

```
import serial
import pynmea2
import os
import time
from datetime import datetime
# clear previous data in data,txt
if os.path.exists("latdata.txt"):
    os.remove("latdata.txt")
if os.path.exists("lngdata.txt"):
    os.remove("lngdata.txt")
if os.path.exists("timedata.txt"):
    os.remove("timedata.txt")
file1 = open('latdata.txt','w')
file2 = open('lngdata.txt','w')
file3 = open('timedata.txt','w')
while True:
    port="/dev/ttyAMA0"
    ser=serial.Serial(port, baudrate=9600, timeout=0.5)
    dataout = pynmea2.NMEAStreamReader()
    newdata=ser.readline()
    n_data=newdata.decode("latin-1")
    # get current time
    current_time = datetime.now()
    # turn current_time into string
    formatted_time = current_time.strftime("%Y.%m.%d_%H:%M:%S")
    if n_data[0:6] == "$GPRMC":
        newmsg=pynmea2.parse(n_data)
        lat=newmsg.latitude
        lng=newmsg.longitude
        lat1=round(lat,4)
        lng1=round(lng,4)
        gps = "Latitude="+str(lat1)+", Longitude="+str(lng1)+", Time="+formatted_time
        print(gps)
        file1.write(str(lat1))
        file1.write("\n") #change row
        file2.write(str(lng1))
        file2.write('\n')
        file3.write(str(formatted_time))
        file3.write("\n")
file1.close()
file2.close()
file3.close()
```

Description:

Read data from NEO-6m, and save longitude, latitude, and corresponding timestamp data into three separate TEXT files. As before, save the data after every three readings.

3) mpu6050_raw.py

```
import mpu6050 import mpu6050
import os
import time
from datetime import datetime
sensor = mpu6050 (0x68)
if os.path.exists("accx.txt"):
    os.remove("accx.txt")
if os.path.exists("accy.txt"):
    os.remove("accy.txt")
if os.path.exists("accz.txt"):
    os.remove("accz.txt")
if os.path.exists("gyrox.txt"):
    os.remove("gyrox.txt")
if os.path.exists("gyroy.txt"):
    os.remove("gyroy.txt")
if os.path.exists("gyroz.txt"):
    os.remove("gyroz.txt")
if os.path.exists("m_timedata.txt"):
    os.remove("m_timedata.txt")
file1 = open(accx.txt','w')
file2 = open(accy.txt','w')
file3 = open(accz.txt','w')
file4 = open(gyrox.txt','w')
file5 = open(gyroy.txt','w')
file6 = open(gyroz.txt','w')
file7 = open('m_timedata.txt','w')
data_count = 0
while True:
    # get current time
    current_time = datetime.now()
    # turn current_time into string
    milliseconds = current_time.microsecond // 1000
    formatted_time = current_time.strftime("%Y.%m.%d_%H:%M:%S")
    file7.write(str(formatted_time))
    file7.write('\n')
    accel_data = sensor.get_accel_data()
    file1.write(str(accel_data['x']))
    file1.write("\n")
    file2.write(str(accel_data['y']))
    file2.write('\n')
    file3.write(str(accel_data['z']))
    file3.write("\n")
    gyro_data = sensor.get_gyro_data()
    file4.write(str(gyro_data['x']))
    file4.write("\n")
    file5.write(str(gyro_data['y']))
    file5.write('\n')
    file6.write(str(gyro_data['z']))
    file6.write("\n")
    data_count += 1
    if data_count == 3:
        file1.flush()
```

```
        file2.flush()
        file3.flush()
        file4.flush()
        file5.flush()
        file6.flush()
        file7.flush()
        os.fsync(file1.fileno())
        os.fsync(file2.fileno())
        os.fsync(file3.fileno())
        os.fsync(file4.fileno())
        os.fsync(file5.fileno())
        os.fsync(file6.fileno())
        os.fsync(file7.fileno())
        data_count = 0
    time.sleep(0.2)
file1.close()
file2.close()
file3.close()
file4.close()
file5.close()
file6.close()
file7.close()
```

Description:
Read data from MPU6050, and save three-axis acceleration, three-axis angular velocity, and corresponding timestamp data into seven separate TEXT files. Save the data after every three readings.

4) camera.py

```
import time
import picamera
import os
from gpiozero import Button
time.sleep(610)
for i in range(1,20):
    camera = picamera.PiCamera()
# image resolution
    camera.resolution = (1920,1080)
#camera.rotation = 180
# delete exist video
    video_file = 'video'+str(i)+'.h264'
    if  os.path.exists(video_file):
        renamed_file = 'video'+str(i)+'.old'
        os.remove(video_file, renamed_file)
# save to file(video)
    camera.start_recording('/home/user/video'+str(i)+'.h264')
    time.sleep(60)
    camera.stop_recording()
    camera.close()
```

Description:
The Pi Camera will record the satellite's flight video in 1920*1080 resolution and 30 frames per second. To prevent data loss after the satellite lands and power is disconnected, Pi Camera saves the video every one minute of recording, and saves them in .h264 format.

I. Data processing

1) Get CubeSat's attitude vector

Our final goal is to obtain the correct attitude vector R of the CubeSat over time.
To achieve this, we start with the acceleration vector A as the initial value for the correct attitude vector R:

Rx(1) = Ax(1)
Ry(1) = Ay(1)
Rz(1) = Az(1)
Calculate the angles Axy, Axz, and Ayz between the correct attitude vector and the coordinate axes using the tangent function:
Ayz(i) = arctan(Ry(i),Rz(i))
Axz(i) = arctan(Rx(i),Rz(i))
Axy(i) = arctan(Rx(i),Ry(i))
Use the angular velocity data RateAyz, RateAxz, and RateAxy multiplied by the time interval to update the angles at the next time step:
Ayz(i+1) = Ayz(i) + RateAyz * delt(i)
Axz(i+1) = Axz(i) + RateAxz * delt(i)
Axy(i+1) = Axy(i) + RateAxy * delt(i)
Convert the angles Axy, Axz, and Ayz into gyroscope vectors G using trigonometric functions:
Gx(i+1) = sin(Axz(i+1)) / sqrt(1+cos(Axz(i+1))^2 * tan(Ayz(i+1))^2)
Gy(i+1) = sin(Ayz(i+1)) / sqrt(1+cos(Ayz(i+1))^2 * tan(Axz(i+1))^2)
Gz(i+1) = cos(Ayz(i+1)) / sqrt(1+sin(Ayz(i+1))^2 * tan(Axy(i+1))^2)
Estimate the correct attitude vector R using a weighted average of the acceleration vector A and the gyroscope vector G. Let the trust weight for the gyroscope be wGyro (set as 5 in this case):
Rx(i+1) = (Ax(i+1) + Gx(i+1) * wGyro) / (1 + wGyro)
Ry(i+1) = (Ay(i+1) + Gy(i+1) * wGyro) / (1 + wGyro)
Rz(i+1) = (Az(i+1) + Gz(i+1) * wGyro) / (1 + wGyro)
By repeating this loop, you can successfully calculate the attitude vector R for the next time step and continue the process to obtain the attitude variation over time.

2) Data Integration and Graph Plotting

We use MATLAB for data organization and GPS map plotting. The processed data will be saved in three new txt files:
m_finaldata.txt for the CubeSat's three-axis attitude data,
g_finaldata.txt for the individual GPS data with timestamps, and **b_finaldata.txt** for temperature, humidity, pressure, and altitude data. Finally, we will plot the GPS locations on the map, create three-dimensional visualizations for GPS and altitude data, and then display the attitude variation graphs.

III. Mission Concept of Operations Overview

A. Mission objectives

The mission objectives of this CubeSat including

1) To monitor temperature, humidity, GPS data, and atmospheric pressure at various altitudes during descent.
2) To capture the CubeSat's flight path using an onboard Pi Camera.
3) To test the effectiveness of a CubeSat for descent from a height of up to 3 kilometers.
4) To verify the practicality and performance of a CubeSat designed and built by students.
5) To cultivate students' ability to construct CubeSats as a hobby.

B. ConOps

The ConOps for this CubeSat mission can be divided into three main parts:

1) CubeSat: The CubeSat is responsible for collecting the necessary data and storing it on an SD card or transmitting it to the ground station via the Communication Subsystem.
2) Rocket: The rocket is responsible for launching the CubeSat to the designated altitude, with two primary requirements: it must allow the CubeSat to be ejected in the direction of the designated coordinate axis, and it must be able to carry a 2U-sized structure for the CubeSat.
3) Ground Station: The Ground Station is responsible to processing CubeSat data and GPS signals, collecting data, and processing and analyzing the CubeSat's location and data.

IV. Conclusion and Lessons Learned

A. Structure Subsystem (STR)

1) CNC thread cutter breaks off

We encountered two tool breakages during machining, one of which was due to the high feed rate of the machining, and the tool was four years old, which we attributed to normal wear and tear.

2) Broken manual screwdriver

Another tool breakage occurred when processing the thread. At the beginning, it was easy to tap too deep at one time. Later, the solution was to strictly follow the steps during processing, and the subsequent thread drilling did not have any problems.

B. Electrical Power Subsystem (EPS)

1) Short circuit

There were several short circuits in the power subsystem in the early stage of production. The reason was that the contacts and the wires were prone to contact short circuits when the wires were squeezed. The solution to this part was to increase the length of the studs supporting the circuit board to avoid extrusion. The situation arises.

2) Payload damage due to wrong wiring

In the previous test, due to unfamiliarity with the circuit, there was a situation where Vcc was connected to SDA and the payload was damaged. This part of the solution is to strictly standardize the colors of the wires, black for grounding, white for power supply, orange, blue and green for the signal line.

C. Payload

1) MPU6050

Measuring acceleration and angular velocity in practice is not difficult, but the process of combining these measurements to calculate attitude angles requires a thorough understanding of the underlying mathematical model to develop the transformation program. We dedicated a considerable amount of time to accomplish this.

During the actual measurements, we also observed that the sensors' minor biases would gradually accumulate, leading to increasing errors in the calculated attitude angles. To obtain precise data, we undertook calculations to completely eliminate these small biases.

2) BMP280

This sensor uses I2C communication, and we often encountered difficulties in reading data without being able to identify the root cause. Eventually, we discovered that the issue was due to a defect in the sensor itself. As a result, we learned the importance of adjusting the I2C settings in the Raspberry Pi and conducting debugging steps, such as checking the I2C address in the terminal.

3) NEO-6m GPS Tracker (GPS1)

When configuring GPS-related components in a program, it is important to set up the relevant settings first. For example, when using a Raspberry Pi, it is necessary to configure the port properly to ensure smooth reception and avoid reading failures later on.

```
sudo systemctl stop serial-getty@ttyAMA0.service
sudo systemctl disable serial-getty@ttyAMA0.service
```

Additionally, when working on indoor projects, it is more appropriate to use an external antenna to ensure proper reception and avoid issues where data is received but cannot be processed correctly or displayed.

4) Pi Camera

During the fabrication process, it is essential to ensure that the Pi Camera interface is properly enabled, and the ribbon cable is correctly connected. After connecting the camera, you can use the following command to check if the Raspberry Pi detects the Pi Camera. If the camera is detected, it will return "supported=1 detected=1," indicating that you can proceed with writing the program.

```
vcgencmd get_camera
```

D. GL300 Real Time GPS Tracker (GPS2)

When using the module, it should be outdoors as its signal reception capability is stronger. However, even in outdoor settings, it is important to first ensure successful reception and repositioning of the GPS signal before proceeding with further usage. This approach will enhance the continuous reception of GPS signals and improve overall performance.

E. On Board Data Handling System

This is our first time using Raspberry Pi, and during the coding process, we gradually became familiar with various commands and settings in the Linux system through the terminal. One time, we accidentally modified a permission file, which resulted in the inability to boot the Raspberry Pi. Fortunately, we managed to salvage the situation by using a Windows computer to modify the files on the SD card. This experience taught us about the interaction between different operating systems and how to identify errors and troubleshoot effectively.

F. Data processing

The reason for choosing MATLAB as the post-processing program is mainly due to its superior capabilities in information processing, which aligns well with our primary focus on data manipulation.
During the consolidation of data from TXT files, both GPS and MPU6050 signals require high precision decimal values to be fully represented. Initially, using "vertcat" and "horzcat" functions to concatenate the data into matrices resulted in numbers displayed in the format of 7.2E*-5 or rounded to four decimal places, which did not meet our requirements. To address this, we need to convert the numerical values to strings before merging them to ensure the data is presented in the desired format.

```
data17_str = num2str(data17,"%.7f");
data18_str = num2str(data18,"%.7f");
total_data4 = horzcat(data17_str, data18_str,data19_sort);
gpstitle = ["Latitude","Longitude","Time"];
data_with_title4 = vertcat(gpstitle,total_data4);
```

When drawing maps, it is essential to pay attention to the completeness and selection of the shapefile. For larger scales, county or city boundary shapefiles can be used, while for smaller scales, district-level shapefiles may be more suitable.

Additionally, when plotting points on the map, it is also crucial to consider potential conflicts with the Mapping Toolbox. To plot multiple points within a matrix, the "plotm" function can be used, while for individual points, the "scatterm" function is appropriate.

```
scatterm(lat_sao_paulo, lon_sao_paulo, 'r', 'filled');
textm(lat_sao_paulo, lon_sao_paulo, 'São Paulo', 'Color', 'black', 'FontSize', 10, 'FontWeight',
'bold','HorizontalAlignment', 'right');
plotm(non_zero_lat_data(:,1), non_zero_lng_data(:,1), 'r');
```

Through these challenges, we have not only gained a deeper understanding of the Raspberry Pi and Linux systems but also developed problem-solving skills that are crucial for our project's success. The journey of learning and overcoming obstacles has been rewarding, and it has allowed us to become more proficient in utilizing Raspberry Pi for our CubeSat project.

V. System Weights, Measures and Performance Data

A. Structure Subsystem (STR)

Term	Specification
Size	200*100*100mm
Weights	1602.6g

Table2. System Weights, Measures

B. Payload

1) MPU6050

- Accelerometer error is between ±1.5 m/s²
- Gyroscope error is between ±15 degree/s

2) BMP280

- temperature error is between ±0.5°C in singly test
- temperature error is between ±1.5°C on the cubesat
- Temperature data delayed by approximately 1 minute.
- pressure error is between ±0.2 hPa after corrected the data
- Relative Altitude error is between ±1.0 meter

3) DHT22

- Humidity error is between ±15% in singly test
- Humidity data delayed by approximately 30 sec.

4) NEO-6M GPS Tracker (GPS1)

- Horizontal Position Accuracy: 3.2572727m>2.5m
- Track record Cycle: 1 sec
- When the satellite is upside-down for more than 1 to 2 seconds, it may cause the GPS to lose its signal. It may result in the GPS outputting default values for Latitude and Longitude, typically set to 0.0.
- GPS can be received at higher speeds.

4) Pi Camera

- Successful focusing and capturing are achievable at long distances.
- Successful focusing and capturing are achievable at short distances.

C. GL300 Real Time GPS Tracker (GPS2)

- Can accurately correspond to the location displayed on Google Maps when stationary.
- Can accurately locate and draw the path during large-scale movements.
- Can accurately locate and draw the path even during high-speed movements.

VI. Project Test Reports

A. Payload

1) Temperature / Humidity

a. Verify the temperature/humidity sensor singly with standard thermometer and hygrometer, and test whether the error is within the manufacturer's specified error range.

Testing method:
Measure temperature and humidity at different locations while simultaneously recording data from a standard thermometer and hygrometer.

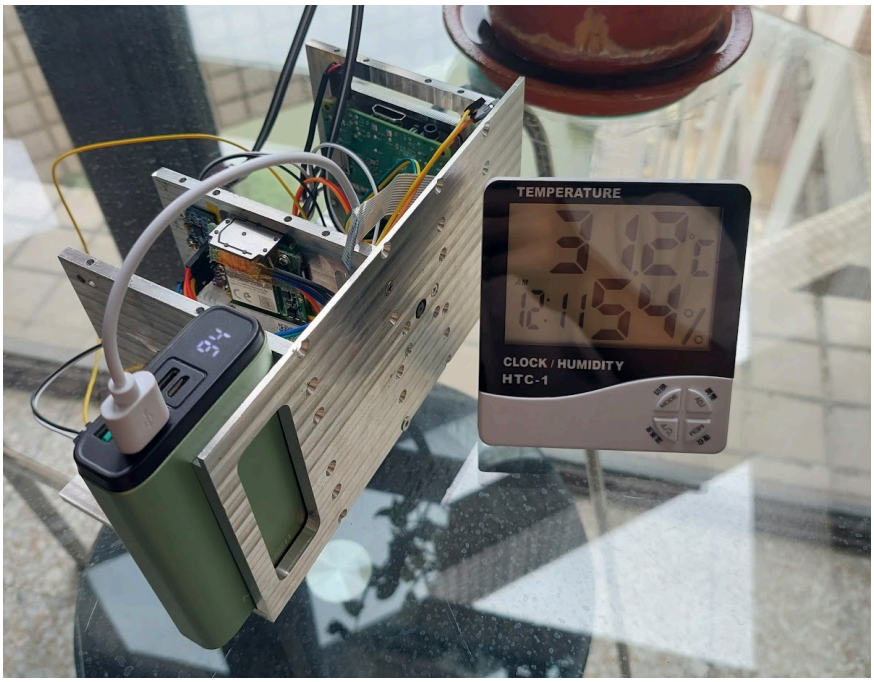


Image3. Temperature and Humidity Test.

Temperture (°C)		
BMP280 sensor	standard thermometer	error
26.73	27.2	-0.47
28.18	27.0	0.18
28.33	27.0	0.33
28.87	29.1	-0.23
29.23	29.2	0.03
29.3	29.3	0
29.11	29.3	-0.19
29.26	29.4	-0.14
29.61	29.5	0.11
29.76	29.6	0.16
29.88	29.7	0.18
30.18	29.8	0.38
29.96	29.7	0.26

29.88	29.6	0.28

Table3. Temperature and thermometer data comparison

error is between -0.47°C to 0.38°C , which is within the manufacturer's specified range, and the mean absolute error is 0.21 °C.

Humidity (%)		
DHT22 sensor	standard hygrometer	error
63.0	48.0	15
68.0	52.0	16
67.5	53.0	14.5
67.3	53.0	14.3
66.6	53.0	13.6
66.4	54.0	12.4
66.7	54.0	12.7

Table 4. Humidity and hygrometer data comparison

error is between 12% to 16% , which exceeds the manufacturer's specified range, and the mean absolute error is 15%.

b. Assemble the sensors onto the cubesat and connect to the power supply and on-board computer, measure the effect of the heat emitted by the cubesat on the temperature/humidity sensor, and adjust the final values based on this data.

Temperature(°C)		
BMP280 sensor	standard thermometer	error
27.57	26.2	1.37
28.25	27.1	1.15
28.58	27.0	1.58
28.87	28.1	0.77
28.67	28.2	0.47
28.95	28.3	0.65
28.34	28.4	-0.06
29.1	28.5	0.6

29.68	28.6	1.08
29.82	28.7	1.12
28.9	28.8	0.1
30.11	29.0	1.11
30.24	29.1	1.14

Table 5. Temperature and thermometer data comparison on cubesat

error is between -0.06°C to 1.58°C , which is within the manufacturer's specified range, and the mean absolute error is 1.02 °C.

2) Barometer

a. Verify the measured pressure with a standard barometer and test whether the error is within the manufacturer's specified range.

Testing method:
Start measuring the atmospheric pressure from the 9th floor and walk down the stairs to the 1st floor. Record data from the standard barometer at each floor, and then take the elevator back to the 9th floor.

Image 4. Atmospheric Pressure Test.

	Pressure (hPa)		
Floor	BMP280 sensor	standard barometer	error
9F	979.14	980.6	-1.46

9F	979.05	980.5	-1.45
8F	979.365	980.9	-1.53
7F	979.78	981.0	-1.22
6F	980.167	981.5	-1.33
5F	980.541	981.9	-1.35
4F	980.945	982.4	-1.45
3F	981.353	982.7	-1.34
2F	981.823	983.2	-1.37
1F	982.212	983.6	-1.38
9F	978.716	980.3	-1.584

Table 6. Pressure and barometer data comparison

error is between -1.584 hPa to -1.22 hPa, and the mean absolute error is 1.41 hPa.
we can see these errors are not random, thus they're actually bias, we add the corrected value 1.41 on the data, error will between -0.174 hPa to 0.19 hPa, which is within the manufacturer's specified range.

b. Calculate the altitude using (formula#1) and compare it with the standard altimeter.

Testing method.
Start measuring the Relative Altitude change from the 9th floor as a reference point, then take the elevator down to the 1st floor. Record the data from the standard altimeter using video recording.

Image 5. Altitude Test.

Relative Altitude (meter)		
BMP280 sensor	standard altimeter	error
0	0	0
-0.01	-1	0.98
-0.17	-1	0.83
-0.68	-1	0.31
-1.62	-2	0.37
-2.68	-3	0.31
-3.89	-3	-0.89
-5.19	-5	-0.19
-6.60	-6	-0.6
-8.13	-8	-0.13
-9.54	-9	-0.54
-10.83	-11	0.17
-12.23	-12	-0.23
-13.66	-13	-0.65
-15.11	-15	-0.11
-16.63	-16	-0.63
-18.06	-18	-0.06
-19.47	-19	-0.47
-21.74	-21	-0.74
-23.17	-23	-0.17
-24.55	-24	-0.55
-26.09	-26	-0.09
-27.35	-27	-0.35
-28.28	-28	-0.28
-28.82	-28	-0.82

Table 7. Pressure and barometer data comparison

error between -0.89m to 0.98m, which is within the manufacturer's specified range, and the mean absolute error is 0.4188 meter.

3) Accelerometer & Gyroscope

a. Place the IMU separately on three axes on a platform and measure the effect of gravity on the accelerometer.

Acceleration(m/s²)			
	Rx	Ry	Rz
+X axis is facing upwards	10.2711	0.0047	-0.5243
+Y axis is facing upwards	0.1915	9.7156	0.0957
+Z axis is facing upwards	0.0359	0.0407	10.5296

Table 8. three axis Accelerometer data comparison

Result : When the +Y axis is facing upwards, the accelerometer on the Y-axis receives a reading of 1g due to the effect of gravity, while the readings on the X and Z axes are 0g.

When the +X and +Z axis is facing upwards, the accelerometer on the Y-axis receives a reading of more than 1g, so we adjust sensor parameters to make its reading equal to 1g.

Data of the X-axis should be multiplied by 9.8/10.3=0.9514

Data of the Z-axis should be multiplied by 9.8/10.5=0.9333

b. Place the IMU in any orientation and measure the readings on the X, Y, and Z axes (Rx, Ry, Rz), which should satisfy the equation：

SQRT(Rx^2 + Ry^2 + Rz^2)=1g.

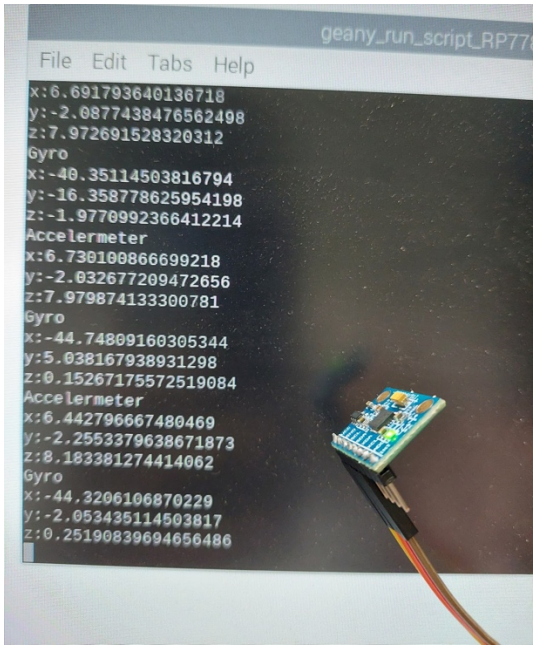


Image 6. Accelerometer test

Rx	Ry	Rz	Sqrt(Rx^2 + Ry^2 + Rz^2)
6.691	-2.087	7.972	10.615
6.442	-2.255	8.183	10.655
0.433	-0.014	10.603	10.611

Table 9. Accelerometer data verification

Its reading 10.615 exceeds 9.8 m/s, if we adjust sensor parameters as shown in the table to make its reading approach to 1g.

Rx	Ry	Rz	Sqrt(Rx^2 + Ry^2 + Rz^2)
6.366	-2.087	7.375	9.963
6.129	-2.255	7.637	10.048
0.411	-0.014	9.896	9.904

Table 10. Accelerometer data verification after adjustment

c. Fix MPU6050 on a platform and use a motor to provide acceleration, compare with standard accelerometer and measure the acceleration error to see if it falls within the manufacturer's error range.

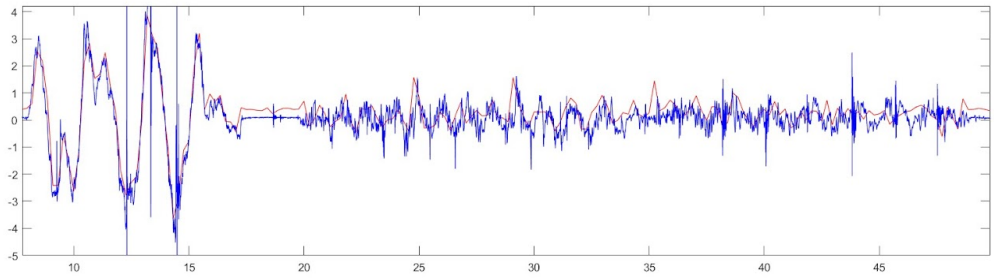


Figure 1. X axis accelerometer comparison (blue line for standard accelerometer, red line for MPU6050 accelerometer)

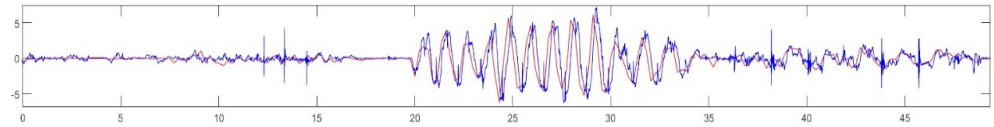


Figure 2. Y axis accelerometer comparison (blue line for standard accelerometer, red line for MPU6050 accelerometer)

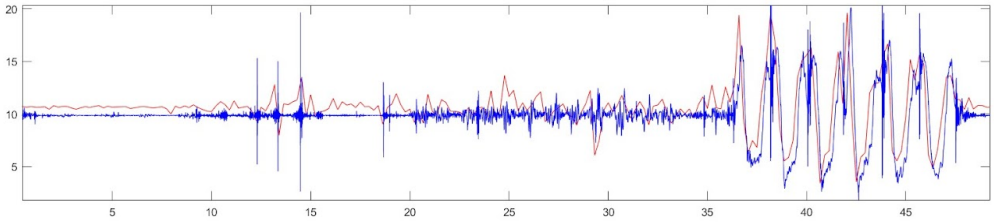


Figure 3. Z axis accelerometer comparison (blue line for standard accelerometer, red line for MPU6050 accelerometer)

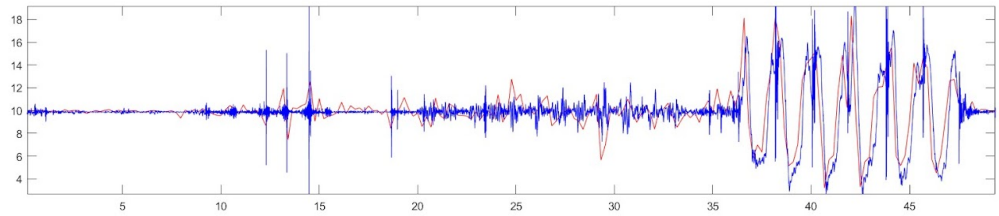


figure 4. Z axis accelerometer comparison after adjustment (blue line for standard accelerometer, red line for MPU6050 accelerometer)

Result：The X, Y, Z axis acceleration reading remains the same as standard accelerometer, The data has an error of less than 1.5 m/s², which is within the manufacturer's error range.

d. Fix the IMU in a cubesat structure and place it on a platform rotating at a velocity separately on yaw, roll and pitch vector. Measure the gyroscopic output and determine if the angular velocity error is within the manufacturer's error range.

i. yaw: The output of the Z-axis of the gyroscope changes, while the other two axes remain unchanged.
ii. roll: The output of the Y-axis of the gyroscope changes, while the other two axes remain unchanged.
iii. pitch: The output of the X-axis of the gyroscope changes, while the other two axes remain unchanged.

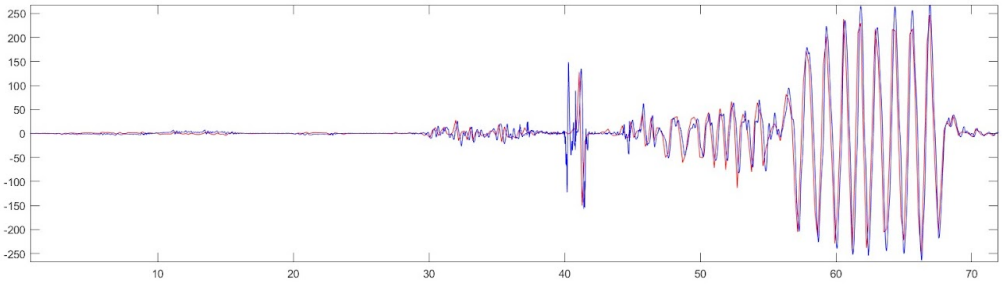


Figure 5. X axis gyroscope comparison (blue line for standard Gyroscope, red line for MPU6050 Gyroscope)

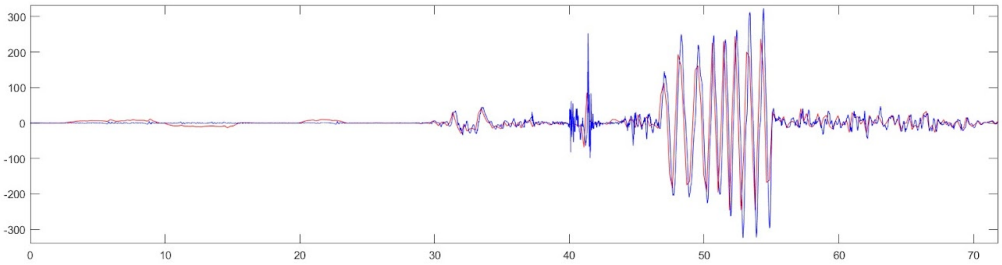


Figure 6. Y axis gyroscope comparison (blue line for standard Gyroscope, red line for MPU6050 Gyroscope)

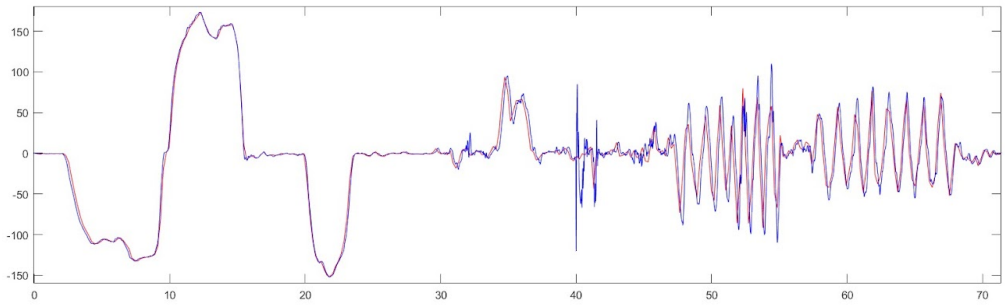


Figure 7. Z axis gyroscope comparison (blue line for standard Gyroscope, red line for MPU6050 Gyroscope)

Results: The X, Y, Z axis angular velocity reading remains the same as standard gyroscope, These data have an error of less than 15 degree/s, which is within the manufacturer's error range.

e. Attitude Stability Test

Place the CubeSat in a stationary position and measure the recorded acceleration and angular velocity data. After performing data processing, calculate the three components of the attitude vector R: Rx, Ry, and Rz.

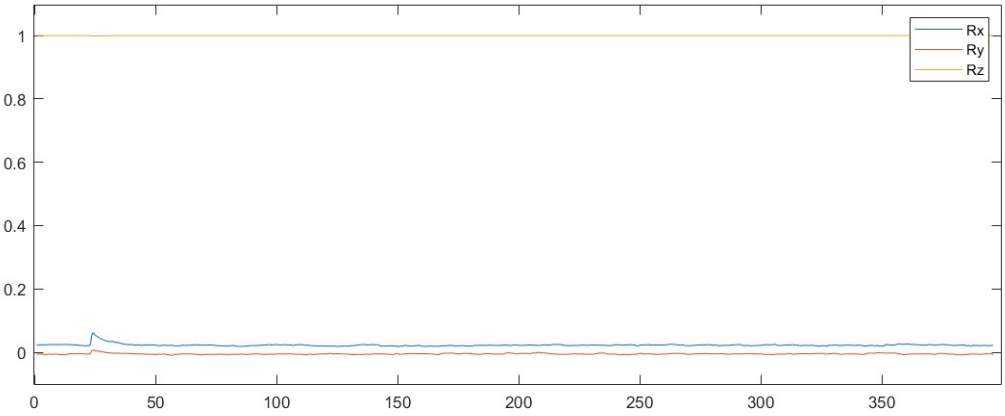


Figure 8. Attitude vector R over time

It can be observed that the attitude remains stable throughout the entire process, without accumulating errors over time. Even with slight disturbances earlier, the attitude returns to its original values.

4) Position Accuracy Test:

We conducted a test using Google Maps as the reference for location. We randomly sampled 22 points on the roads within the Chung Yuan Christian University campus. At each sample point, we paused for a few seconds and recorded the latitude and longitude values from Google Maps. This allowed us to obtain the GPS coordinates from the NEO-6M module and the corresponding latitude and longitude information from Google Maps at the same time. By comparing the two sets of data, we were able to determine the margin of error.

Using MATLAB, we converted the difference in latitude and longitude values into the distance between two points, which represents the margin of error.

```
% change lat & lng into rad
standard_latitude_rad = deg2rad(standard_latitude);
standard_longitude_rad = deg2rad(standard_longitude);
measured_latitude_rad = deg2rad(closest_coordinates(1));
measured_longitude_rad = deg2rad(closest_coordinates(2));
% define radius of earth(m)
earth_radius_meters = 6371000;
% use Haversine to calculate distance of two position
delta_latitude = measured_latitude_rad - standard_latitude_rad;
delta_longitude = measured_longitude_rad - standard_longitude_rad;
a = sin(delta_latitude/2)^2 + cos(standard_latitude_rad) *
cos(measured_latitude_rad) * sin(delta_longitude/2)^2;
c = 2 * atan2(sqrt(a), sqrt(1-a));
distance_meters = earth_radius_meters * c;
```

Google Map		NEO-6m			
Latitude	Longitude	Latitude	Longitude	Time	Change to meter(m)
24.9698006	121.1919746	24.9697992	121.1919737	23:57	0.19
24.969799	121.1924681	24.9697882	121.1924635	23:58	1.29
24.9698114	121.1936386	24.9698208	121.1935537	00:00	8.63
24.9697498	121.1943225	24.9698002	121.1943425	00:02	5.95
24.9697735	121.1950098	24.9697752	121.1950092	00:02	0.2
24.9693914	121.1955013	24.9693988	121.1954778	00:03	2.51
24.9688459	121.1954722	24.9688315	121.1953938	00:04	8.06
24.9675897	121.1954661	24.967604	121.1954838	00:05	2.39
24.9669447	121.1954782	24.9669257	121.195472	00:07	2.21
24.9667836	121.1948823	24.966769	121.1948888	00:10	1.75
24.9668	121.1946293	24.966774	121.1946647	00:11	4.59
24.9668049	121.1939011	24.9667918	121.1938878	00:12	1.97
24.9668067	121.1934471	24.9668045	121.1934495	00:13	0.34
24.9668091	121.1929982	24.9668037	121.1930338	00:14	3.64
24.9667897	121.1924832	24.9667922	121.1924902	00:15	0.75
24.9674088	121.1909577	24.9673945	121.1909545	00:16	1.62
24.9674075	121.1899012	24.9674003	121.1898993	00:18	0.82
24.9692282	121.1882567	24.9692258	121.1882553	00:20	0.3
24.9696717	121.188875	24.969723	121.1888495	00:21	6.26
24.9698385	121.1906033	24.9697357	121.1906642	00:22	12.98
24.9698261	121.1911116	24.9698123	121.1911008	00:23	1.88
24.9698197	121.1914827	24.9697937	121.1914663	00:24	3.33

Table 10. Contrast of Google Map and NEO-6m

Finally, we obtained an average error of 3.2572727.

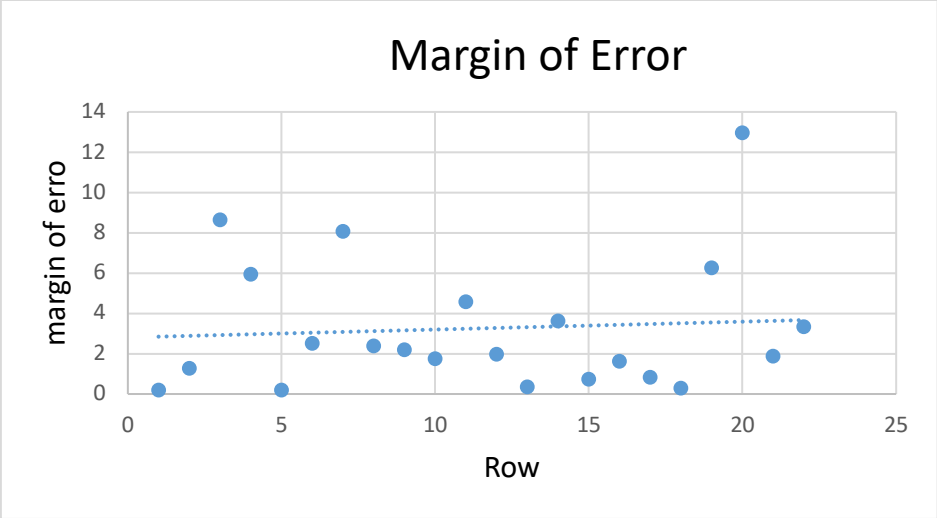


Figure 9. Margin of Error

Analysis of the Error:

On the testing day, there was a typhoon, and the large number of test locations may have contributed to the NEO-6M GPS module's accuracy being greater than 2 meters. Particularly significant errors may have occurred when passing under trees or near large buildings during the testing process. These environmental factors could have interfered with the GPS signals, leading to larger inaccuracies in certain measurements.

5) Rotation Test:

Due to the potential interference caused by satellite rotation affecting GPS signal reception, we are conducting this test. As the NEO-6m is located in the first layer of the satellite, which has an open design, we are inverting the satellite to assess the GPS reception when the satellite's main body shields the NEO-6m module from above.

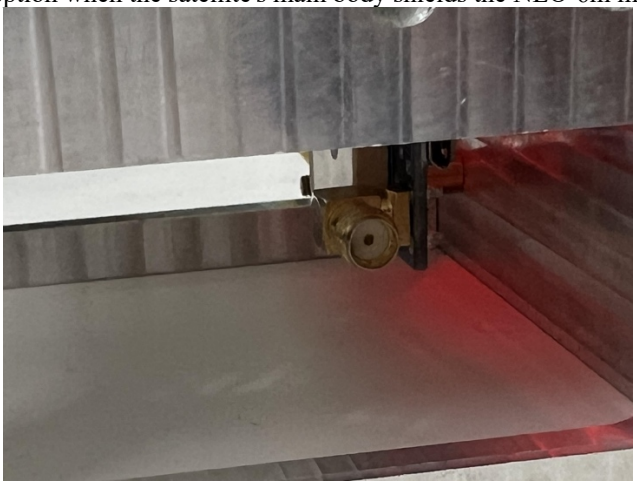


Image 7. Upside down GPS module

After conducting the test, it was observed that when the satellite is inverted, GPS signal loss may occur if the inversion lasts for more than 1 to 2 seconds. This situation can lead to the GPS outputting Latitude=Longitude=0.0. However, during flight, the satellite continues to rotate, and the probability of this condition persisting for more than 1 second is low.

6) Rotation Test:

Placing the satellite on a vehicle, we will move it on a highway at speeds ranging from 100 to 150 km/h and measure its reception performance. Finally, we will use MATLAB to plot the GPS data on a map.

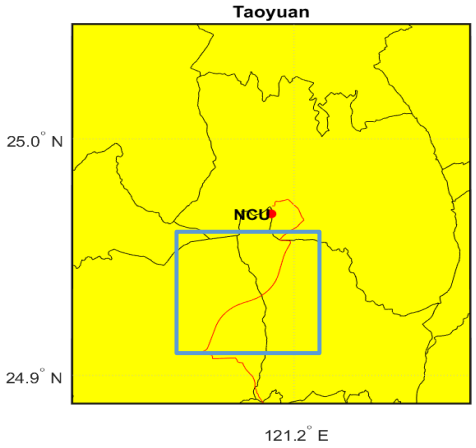


Figure 10. Matlab Map

The blue curve inside the blue box represents the smooth curve of the highway within the county. It can be observed that even at speeds ranging from 100 to 150 km/h, the GPS remains effectively received and functional.

7) Pi Camera Test

Perform two recordings at the same resolution (1920*1080) to test the ability to focus at close and distant distances.

Image 8. Short distance photo

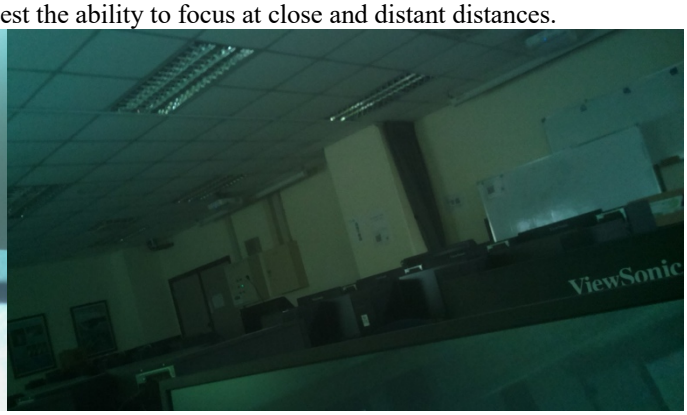


Image 9. Long distance photo

B.GL300 Real GPS Tracker (GPS2)

1) Static measurement

Using the high-speed highway and the NEO-6m GPS module simultaneously for measurement, we will observe whether their paths and point signal locations match.

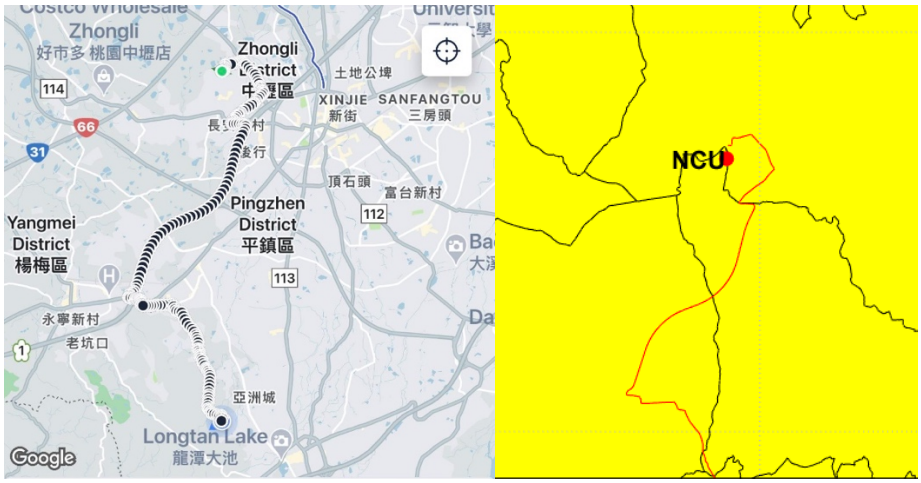


Image 10. Comparison of Google Maps and Matlab Maps

On the left side, the moving signal returned by GL300 indicates that the path matches the one shown on the right side by the NEO-6m GPS module.

VII. Hazard Analysis

subsystem	hazard descriptions	severity	Probability	Recommended control
EPS	Circuit short-circuit causes fire.	IV	Remote	Check the circuit, install fuses.
	Circuit short-circuit damages the CPU.	II	Occasional	Check the circuit, install fuses.
	Excessive current damages the components.	I	Occasional	Check the circuit, install fuses.
	Lithium battery explosion.	V	Remote	Set the thermal switch
OBC	CPU malfunction overheating.	I	Occasional	Check the code and pray.
	SD card fractured.	I	Probable	Optimize component configuration and screw tightening.
Payload	Sensor short-circuit causes fire.	IV	Improbable	Check the circuit.
Structure	hitting a person during descent	V	Improbable	Install descent subsystem.
	damaging objects (property) during descent.	III	Remote	

Table 11. Hazard analysis table

VIII. Risk Assessment

A. Structure Subsystem (STR)
STR-1 Structure disintegrates in mid-air
STR-2 Structure disintegrates after landing

STR-3 Structure cannot be successfully disengaged from the vehicle

B. Electrical Power Subsystem (EPS)

EPS-1 Abnormal power supply of battery
EPS-2 wire break away
EPS-3 After the falls to the ground, the battery is disconnected from the GPS and cannot start the GPS
EPS-4 battery overheats and catches fire
EPS-5 circuit short

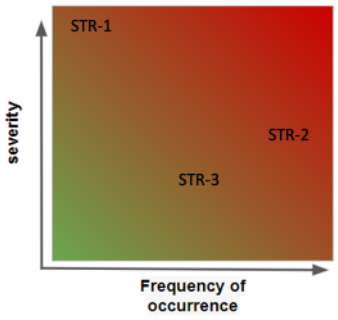


Figure 11.STR Risk Assessment.

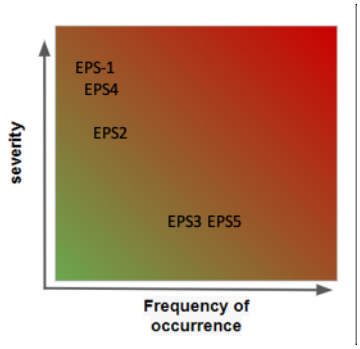


Figure 12. EPS Risk Assessment.

C. Payload

1) MPU6050

MPU-1 sensor malfunction.
MPU-2 I2C connecting failure.
MPU-3 Excessive vibration leading to significant data inaccuracies.
MPU-4 Severe vibration causing disconnection in the circuit.
MPU-5 Weather conditions causing the sensor exposure to water, rendering it inoperable.

2) BMP280

BMP-1 sensor malfunction.
BMP-2 I2C connecting failure.
BMP-3 Severe vibration causing disconnection of the circuit.
BMP-4 Weather conditions causing the sensor exposure to water, resulting in decreased accuracy.

3) DHT22

DHT-1 sensor malfunction.
DHT-2 Severe vibration causing disconnection of the circuit.
DHT-3 Weather conditions causing the sensor exposure to water, resulting in decreased accuracy.

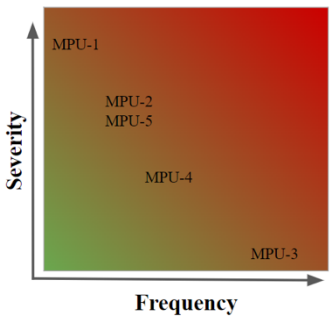


Figure 13. MPU Risk Assessment.

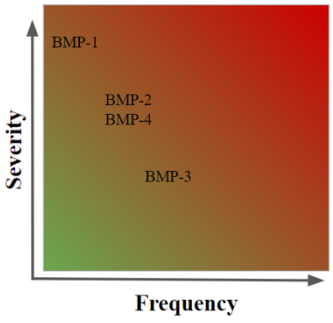


Figure 14. BMP Risk Assessment.

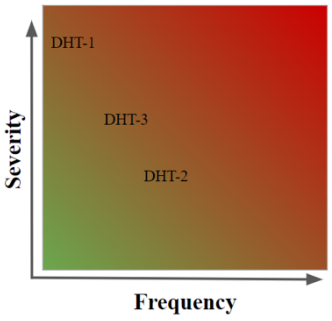


Figure 15. DHT Risk Assessment.

4) NEO-6m(GPS1):

GPS1-1 Software Bug.

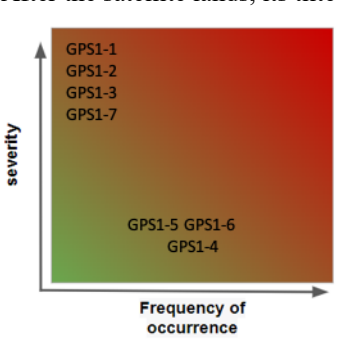
GPS1-2 Payload Failure.
GPS1-3 Antenna Failure
GPS1-4 Weather conditions cause GPS signal degradation.
GPS1-5 Weather conditions cause GPS reception failure.
GPS1-6 Satellite orientation causes temporary GPS reception loss.
GPS1-7 Satellite orientation causes complete GPS signal loss.

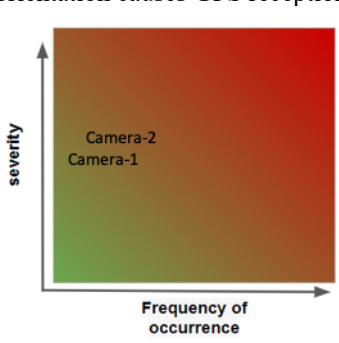
5) Pi Camera

Camera-1 Vibrations cause damage to camera lenses.
Camera-2 Vibrations cause damage to the interface.

D. GL300(GPS2)

GPS2-1 Module Failure
GPS2-2 Weather conditions cause decreased GPS signal strength and accuracy.
GPS2-3 Weather conditions cause GPS reception and transmission failure.
GPS2-4 After the satellite lands, its tilted orientation causes GPS reception and transmission failure.





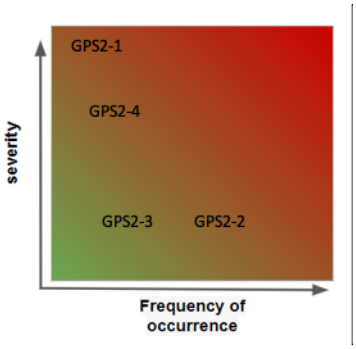
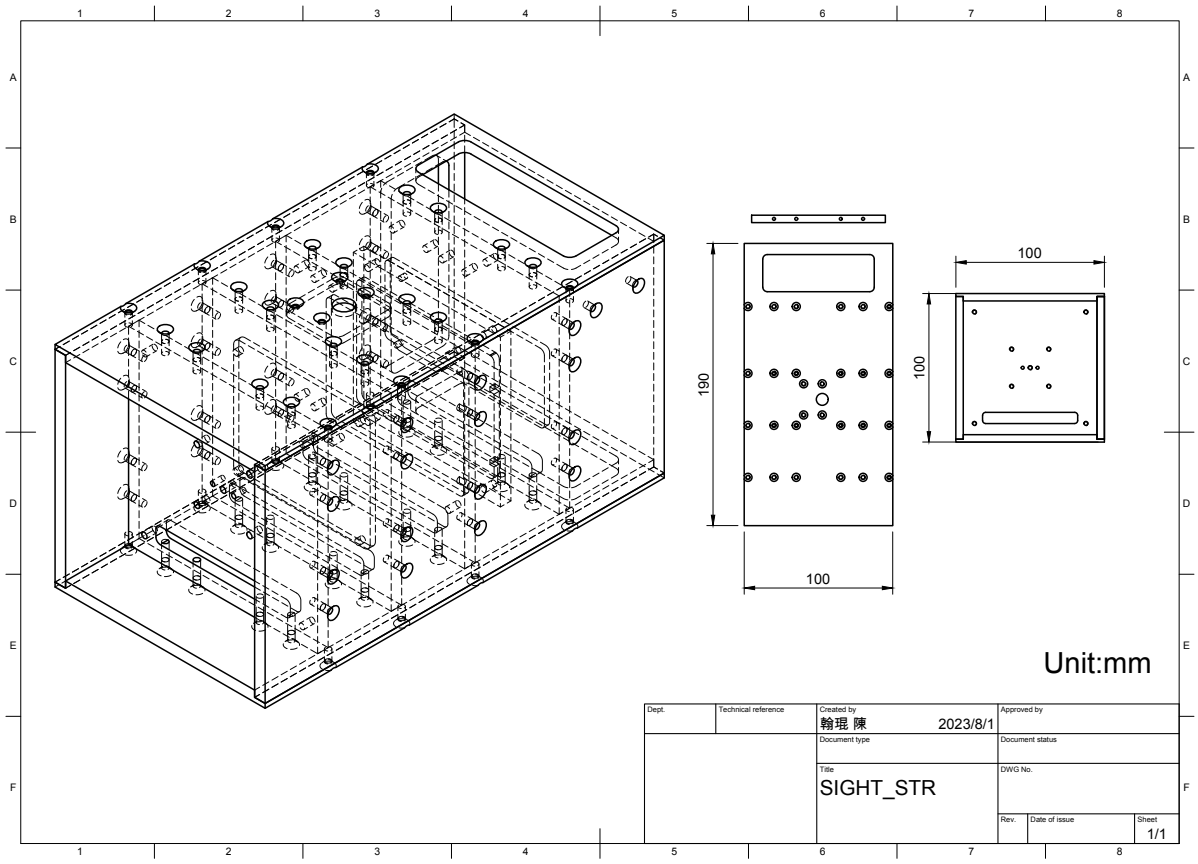


Figure 13. GPS1 Risk Assessment. Figure 14. Camera Risk Assessment. Figure 15. GPS2 Risk Assessment.

IX. Assembly, Preflight, Launch, and Recovery Checklists

X. Engineering Drawings



XI. List of References

4

A. Images, Figures, and Tables

All artwork, captions, figures, graphs, and tables will be reproduced exactly as submitted. Be sure to position any figures, tables, graphs, or pictures as you want them printed.

Do not insert your tables and figures in text boxes. Figures should have no background, borders, or outlines. In the electronic template, use the “Figure” style from the pull-down formatting menu to type caption text. You may also insert the caption by going to the References menu and choosing Insert Caption. Make sure the label is “Fig.,” and type your caption text in

the box provided. Captions are bold with a single tab (no hyphen or other character) between the figure number and figure description.

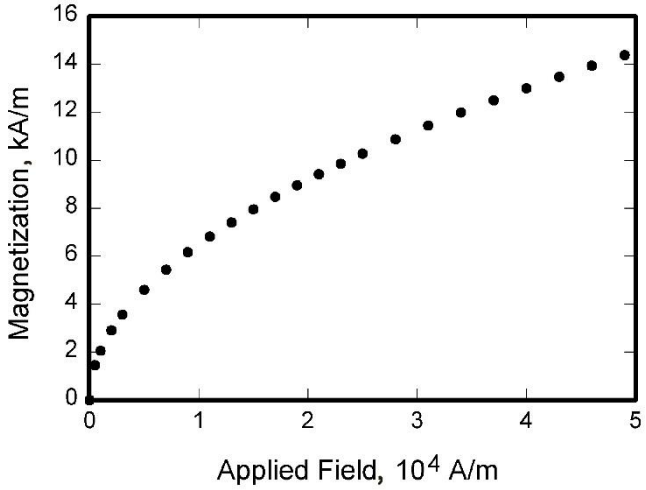


Fig. 1 Magnetization as a function of applied fields.

Place figure captions below all figures; place table titles above the tables. If your figure has multiple parts, include the labels "a)," "b)," etc. below and to the left of each part, above the figure caption. Please verify that the figures and tables you mention in the text actually exist. *Please do not include captions as part of the figures, and do not put captions in separate text boxes linked to the figures*. When citing a figure in the text, use the abbreviation "Fig." except at the beginning of a sentence. Do not abbreviate "Table." Number each different type of illustration (i.e., figures, tables, images) sequentially with relation to other illustrations of the same type.

Figure axis labels are often a source of confusion. Use words rather than symbols. As in the example to the right, write the quantity "Magnetization" rather than just "M." Do not enclose units in parenthesis, but rather separate them from the preceding text by commas. Do not label axes only with units. As in Fig. 1, for example, write "Magnetization, kA/m" not just "kA/m." Do not label axes with a ratio of quantities and units. For example, write "Temperature, K," not "Temperature/K."

Multipliers can be especially confusing. Write "Magnetization, kA/m" or "Magnetization, 10^3 A/m." Do not write "Magnetization (A/m) x 1000" because the reader would not then know whether the top axis label in Fig. 1 meant 16000 A/m or 0.016 A/m. Figure labels must be legible, and all text within figures should be uniform in style and size, no smaller than 8-point type.

B. Equations, Numbers, Symbols, and Abbreviations

Equations are centered and numbered consecutively, with equation numbers in parentheses flush right, as in Eq. (1). Insert a blank line above and below the equation. First use the equation editor to create the equation. If you are using Microsoft Word, use either the Microsoft Equation Editor or the MathType add-on (http://www.mathtype.com) for equations in your paper, use the function (Insert>Object>Create New>Microsoft Equation *or* MathType Equation) to insert it into the document. Please note that "Float over text" should *not* be selected. To insert the equation into the document:

1) Select the "Equation" style from the pull-down formatting menu and hit "tab" once.
2) Insert the equation, hit "tab" again,
3) Enter the equation number in parentheses.

A sample equation is included here, formatted using the preceding instructions. To make your equation more compact, you can use the solidus (/), the exp function, or appropriate exponents. Use parentheses to avoid ambiguities in denominators.

$$\int_0^{r_2} F(r,\varphi)\, dr\, d\varphi = [\sigma r_2 / (2\mu_0)] \cdot \int_0^{\infty} \exp(-\lambda\,|z_j - z_i\,|)\, \lambda^{-1}\, J_1(\lambda r_2)\, J_0(\lambda r_i)\, d\lambda \tag{1}$$

Be sure that the symbols in your equation are defined before the equation appears, or immediately following. Italicize symbols (*T* might refer to temperature, but T is the unit tesla). Refer to "Eq. (1)," not "(1)" or "equation (1)" except at the beginning of a sentence: "Equation (1) is…" Equations can be labeled other than "Eq." should they represent inequalities, matrices, or boundary conditions. If what is represented is really more than one equation, the abbreviation "Eqs." can be used.

Define abbreviations and acronyms the first time they are used in the text, even after they have already been defined in the abstract. Very common abbreviations such as AIAA, SI, ac, and dc do not have to be defined. Abbreviations that incorporate periods should not have spaces: write "P.R.," not "P. R." Delete periods between initials if the abbreviation has three or more initials; e.g., U.N. but ESA. Do not use abbreviations in the title unless they are unavoidable (for instance, "AIAA" in the title of this article).

C. General Grammar and Preferred Usage

Use only one space after periods or colons. Hyphenate complex modifiers: "zero-field-cooled magnetization." Avoid dangling participles, such as, "Using Eq. (1), the potential was calculated." [It is not clear who or what used Eq. (1).] Write instead "The potential was calculated using Eq. (1)," or "Using Eq. (1), we calculated the potential."

Use a zero before decimal points: "0.25," not ".25." Use "cm^2," not "cc." Indicate sample dimensions as "0.1 cm x 0.2 cm," not "0.1 x 0.2 cm^2." The preferred abbreviation for "seconds" is "s," not "sec." Do not mix complete spellings and abbreviations of units: use "Wb/m^2" or "webers per square meter," not "webers/m^2." When expressing a range of values, write "7 to 9" or "7-9," not "7~9."

A parenthetical statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within parenthesis.) In American English, periods and commas are placed within quotation marks, like "this period." Other punctuation is "outside"! Avoid contractions; for example, write "do not" instead of "don't." The serial comma is preferred: "A, B, and C" instead of "A, B and C."

If you wish, you may write in the first person singular or plural and use the active voice ("I observed that…" or "We observed that…" instead of "It was observed that…"). Remember to check spelling. If your native language is not English, please ask a native English-speaking colleague to proofread your paper.

The word "data" is plural, not singular (i.e., "data are," not "data is"). The subscript for the permeability of vacuum μ_0 is zero, not a lowercase letter "o." The term for residual magnetization is "remanence"; the adjective is "remanent"; do not write "remnance" or "remnant." The word "micrometer" is preferred over "micron" when spelling out this unit of

measure. A graph within a graph is an "inset," not an "insert." The word "alternatively" is preferred to the word "alternately" (unless you really mean something that alternates). Use the word "whereas" instead of "while" (unless you are referring to simultaneous events). Do not use the word "essentially" to mean "approximately" or "effectively." Do not use the word "issue" as a euphemism for "problem." When compositions are not specified, separate chemical symbols by en-dashes; for example, "NiMn" indicates the intermetallic compound $Ni_{0.5}Mn_{0.5}$ whereas "Ni–Mn" indicates an alloy of some composition Ni_xMn_{1-x}.

Be aware of the different meanings of the homophones "affect" (usually a verb) and "effect" (usually a noun), "complement" and "compliment," "discreet" and "discrete," "principal" (e.g., "principal investigator") and "principle" (e.g., "principle of measurement"). Do not confuse "imply" and "infer."

Prefixes such as "non," "sub," "micro," "multi," and ""ultra" are not independent words; they should be joined to the words they modify, usually without a hyphen. There is no period after the "et" in the abbreviation "et al." The abbreviation "i.e.," means "that is," and the abbreviation "e.g.," means "for example" (these abbreviations are not italicized).

VII. Conclusion

A conclusion section is required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions. *Note that the conclusion section is the last section of the paper that should be numbered. The appendix (if present), acknowledgment, and references should be listed without numbers.*

Appendix

Required appendices should appear before the acknowledgments.

Acknowledgments

An Acknowledgments section, if used, **immediately precedes** the References. Sponsorship information and funding data are included here. The preferred spelling of the word "acknowledgment" in American English is without the "e" after the "g." Avoid expressions such as "One of us (S.B.A.) would like to thank…" Instead, write "F. A. Author thanks…"

References

The following pages are intended to provide examples of the different reference types. All references should be in 9-point font, with the first line flush left and reference numbers inserted in brackets. You are not required to indicate the type of reference; different types are shown here for illustrative purposes only. The DOI (digital object identifier) should be incorporated in every reference for which it is available (see Ref. 1 sample); for more information on DOIs, visit www.doi.org or www.crossref.org.

Periodicals

[1] Vatistas, G. H., Lin, S., and Kwok, C. K., "Reverse Flow Radius in Vortex Chambers," *AIAA Journal*, Vol. 24, No. 11, 1986, pp. 1872, 1873.
doi: 10.2514/3.13046

[2] Alyanak, E. J., and Pendleton, E., "Aeroelastic Tailoring and Active Aeroelastic Wing Impact on a Lambda Wing Configuration," Journal of Aircraft, published online 10 Nov. 2016.
doi: 10.2514/1.C033040

[3] Dornheim, M. A., "Planetary Flight Surge Faces Budget Realities," *Aviation Week and Space Technology*, Vol. 145, No. 24, 9 Dec. 1996, pp. 44–46.

[4] Terster, W., "NASA Considers Switch to Delta 2," *Space News*, Vol. 8, No. 2, 13–19 Jan. 1997, pp. 1, 18.

All of the preceding information is required. The journal issue number ("No. 11" in Ref. 1) is preferred, but the month (Nov.) can be substituted if the issue number is not available. Use the complete date for daily and weekly publications. Transactions follow the same style as other journals.

Books

[5] Peyret, R., and Taylor, T. D., *Computational Methods in Fluid Flow*, 2nd ed., Springer-Verlag, New York, 1983, Chaps. 7, 14.

[6] Oates, G. C. (ed.), *Aerothermodynamics of Gas Turbine and Rocket Propulsion*, AIAA Education Series, AIAA, New York, 1984, pp. 19, 136.

[7] Volpe, R., "Techniques for Collision Prevention, Impact Stability, and Force Control by Space Manipulators," *Teleoperation and Robotics in Space*, edited by S. B. Skaar and C. F. Ruoff, Progress in Astronautics and Aeronautics, AIAA, Washington, DC, 1994, pp. 175–212.

Publisher, place, and date of publication are required for all books. No state or country is required for major cities: New York, London, Moscow, etc. A differentiation must always be made between Cambridge, MA, and Cambridge, England, UK. Note that series titles are in Roman type.

Proceedings

[8] Thompson, C. M., "Spacecraft Thermal Control, Design, and Operation," *AIAA Guidance, Navigation, and Control Conference*, CP849, Vol. 1, AIAA, Washington, DC, 1989, pp. 103–115

[9] Chi, Y. (ed.), *Fluid Mechanics Proceedings*, NASA SP-255, 1993.

[10] Morris, J. D., "Convective Heat Transfer in Radially Rotating Ducts," *Proceedings of the Annual Heat Transfer Conference*, edited by B. Corbell, Vol. 1, Inst. of Mechanical Engineering, New York, 1992, pp. 227–234.

Reports, Theses, and Individual Papers

[11] Chapman, G. T., and Tobak, M., "Nonlinear Problems in Flight Dynamics," NASA TM-85940, 1984.

[12] Brandis, A. M., Johnston, C. O., and Cruden, B. A., "Nonequilibrium Radiation for Earth Entry," AIAA Paper 2016-3690, June 2016.

[13] Steger, J. L., Jr., Nietubicz, C. J., and Heavey, J. E., "A General Curvilinear Grid Generation Program for Projectile Configurations," U.S. Army Ballistic Research Lab., Rept. ARBRL-MR03142, Aberdeen Proving Ground, MD, Oct. 1981.

[14] Tseng, K., "Nonlinear Green's Function Method for Transonic Potential Flow," Ph.D. Dissertation, Aeronautics and Astronautics Dept., Boston Univ., Cambridge, MA, 1983.

Government agency reports do not require locations. For reports such as NASA TM-85940, neither insert nor delete dashes; leave them as provided. Place of publication *should* be given, although it is not mandatory, for military and company reports. Always include a city and state for universities. Papers need only the name of the sponsor; neither the sponsor's location nor the conference name and location is required. *Do not confuse proceedings references with conference papers*.

Electronic Publications

Regularly issued electronic journals and other publications are permitted as references. Include the DOI if provided; otherwise provide the full URL. Archived data sets also may be referenced as long as the material is openly accessible, and the repository is committed to archiving the data indefinitely. References to electronic data available only from personal websites or commercial, academic, or government ones where there is no commitment to archiving the data are not permitted in the reference list.

[15] Atkins, C. P., and Scantelbury, J. D., "The Activity Coefficient of Sodium Chloride in a Simulated Pore Solution Environment," *Journal of Corrosion Science and Engineering* [online journal], Vol. 1, No. 1, Paper 2, URL: http://www.cp/umist.ac.uk/JCSE/vol1/vol1.html [retrieved 13 April 1998].

[16] Vickers, A., "10-110 mm/hr Hypodermic Gravity Design A," *Rainfall Simulation Database* [online database], URL: http://www.geog.le.ac.uk/bgrg/lab.htm [retrieved 15 March 2006].

Break website addresses after punctuation, and do not hyphenate at line breaks.

Computer Software

[17] TAPP, Thermochemical and Physical Properties, Software Package, Ver. 1.0, E. S. Microware, Hamilton, OH, 1992.

Include a version number and the company name and location of software packages.

Patents

Patents appear infrequently. Be sure to include the patent number and date.

[18] Scherrer, R., Overholster, D., and Watson, K., Lockheed Corp., Burbank, CA, U.S. Patent Application for a "Vehicle," Docket No. P-01-1532, filed 11 Feb. 1979.

Private Communications and Websites

References to private communications and personal website addresses are not permitted. They may, however, be incorporated into the main text of a manuscript or may appear in footnotes.

Unpublished Papers and Books

Unpublished works can be used as references as long as they are being considered for publication or can be located by the reader (such as papers that are part of an archival collection). If a journal paper or a book is being considered for publication, choose the format that reflects the status of the work (depending upon whether it has been accepted for publication):

[19] Doe, J., "Title of Paper," *Name of Journal* (to be published).

[20] Doe, J., "Title of Chapter," *Name of Book*, edited by…, Publisher's name and location (to be published).

[21] Doe, J., "Title of Work," Name of Archive, Univ. (or organization), City, State, Year (unpublished).

Unpublished works in an archive *must* include the name of the archive and the name and location of the university or other organization where the archive is held. Also include any cataloging information that may be provided.