PROJECT VIKRAM-SAT 1.0

Prepared by:
Ved Raj
Iman Sukul
Reetika Singh

Guided by:
Prof.Dr.Shrishail Hiremath
Neelesh Biswas
Rishi Sriram

VIKRAM 1.0

Abstract

The main objective of this project is designing and building a basic prototype for a CanSat (A CanSat mission is a simulation of a real satellite mission where the hardware has to fit into a soda can) and deploying it using balloon method or parachute method to measure some physical quantities like temperature, pressure, height, etc. which can then be used to prove the hydrostatic equation ($dP/dz = -\rho g$) of physics. The Cansat's outer body consists of a soda can with padding and the inner circuit consists of an ESP-32 microcontroller, sensors, SD card module and battery. C++ was used to write the code for the microcontroller. Despite many constraints we were able to successfully build our circuit and fit it in a small can lengthwise. We could also obtain the required data.

I. INTRODUCTION

For our primary mission, our CanSat measures the air pressure and temperature as a function of elapsed time. This allows us to calculate the altitude (height) as a function of time elapsed and the descent speed as demanded in the requirements.

We first practiced on a breadboard using jumper wires, trying out different codes and ending up with a good version. Then we proceeded to solder our circuit elements on a veroboard.

II. THE CONSTRAINTS

In CanSat making, the major obstacle is building the model for the given problem statement, keeping in mind the size and weight constraints. This is done in order to give students an exposure to real satellite building where there are strict constraints. Our major constraint was fitting the entire circuit in a small soda can.

Dimensions of the Soda Can:

Length - 12.2 cm
Diameter - 5.4 cm

The mass of the soda can also could not be increased very much as it would require a parachute with a bigger surface area. Also, there was a need for appropriate padding to ensure that the CanSat remained unharmed during operation.
Total mass of the CanSat - 275 g

Figure 1: Veroboard inside the Soda Can

III. PROJECT DESCRIPTION

The Circuit

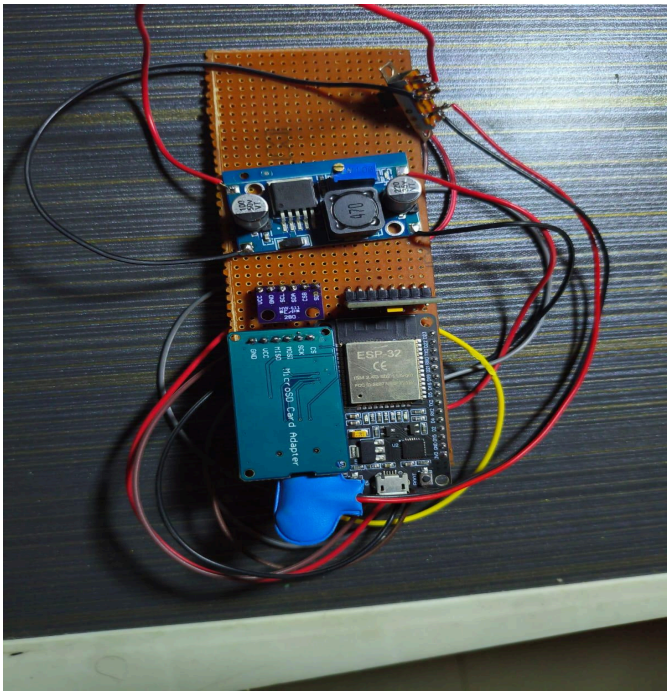

Figure 2: CanSat Circuit on Veroboard (without battery)

S.No.	Component(s)	Model
1.	ESP-32 Microcontroller	DEVKITDOIT v1
2.	IMU Sensor	MPU6050
3.	Pressure and Temperature Sensor	BMP280
4.	SD Card Module with SD Card	—
5.	DC-DC Step Down Buck Converter	LM2596
6.	9V Battery with connector	HW Battery
7.	Slide Switch and Wires	—
8.	Veroboard	—

Table 1: Components in the CanSat circuit

As seen in the block diagram below, our system consists of the following parts:

1. An ESP32 (DEVKIT DOIT v1 development board): It is a development board based on the ESP32-WROOM-32 module. This board integrates complete Wi-Fi and Bluetooth Low Energy functions. It has micro-USB B connectivity for power, programming and debugging.

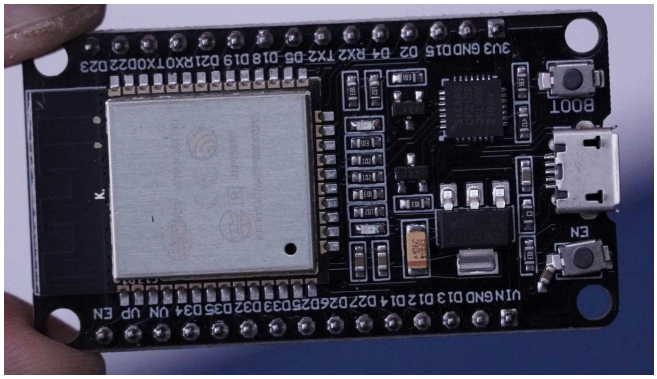

Figure 3: ESP32 (DEVKIT DOIT v1 development board)[1]

2. Sensors needed to determine the position and direction of our CanSat: IMU (MPU 6050) – consisting of 6 Degrees of freedom (3 axes used for measuring Translatory & 3 axes for Rotational motion)

- It measures and reports a body's specific force, angular rate and sometimes the magnetic field surrounding the body.
- Primary focus of this sensor is to determine the motion and orientation of the objects it's attached to.

  It is highly necessary that we should keep the sensor mounted orthogonally (at 90°) to measure motion along and around three principal axes (x, y & z often corresponding to roll, pitch and yaw)

  (Also, it does measure temperature but the error in measurement is very high so we relied only on BMP280).

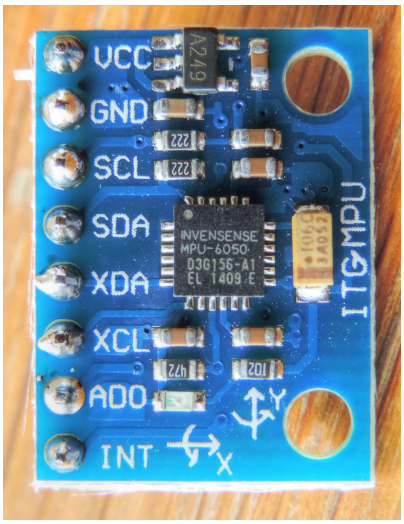

Figure 4: MPU6050 Sensor[2]

3. Sensor needed to sense temperature , pressure and height: BMP 280 –
   It is used to measure the barometric pressure and temperature (Low power consumption)
   It is a dual function sensor:
- Measures Barometric pressure (hPa/mbar):
  Accuracy - ±1 hPa (absolute)
- Measures temperature (℃):
  Accuracy - ±1 ℃

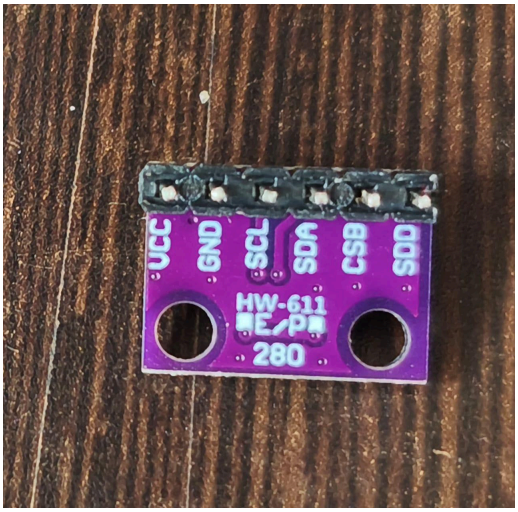

Figure 5: BMP280 Sensor

4. An SD card module with SD card: This is used to log data automatically in the form of a '.csv' file which can be later viewed as a spreadsheet. This helps ensure that a soft copy of the data is safely stored on board.

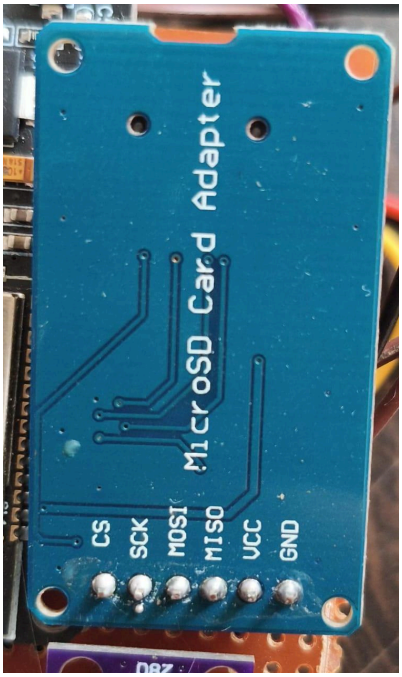

Figure 6: SD card module with SD card

5. A 9V HW battery with DC-DC step down buck converter (LM2596): This is used to power the CanSat when deployed. The buck converter steps down the 9V to 5V which is then supplied to the microcontroller.

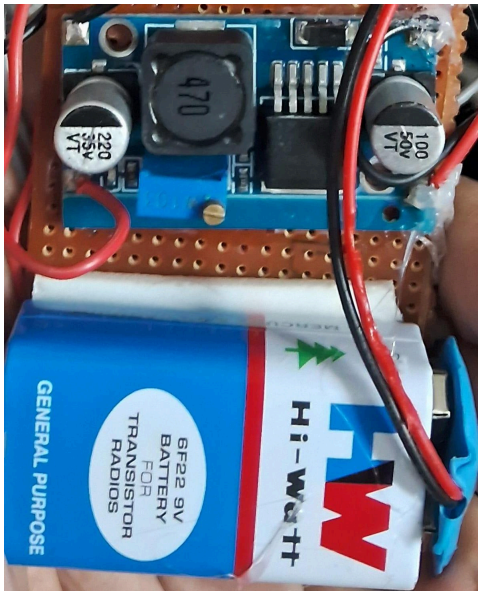

Figure 7: Battery and buck converter

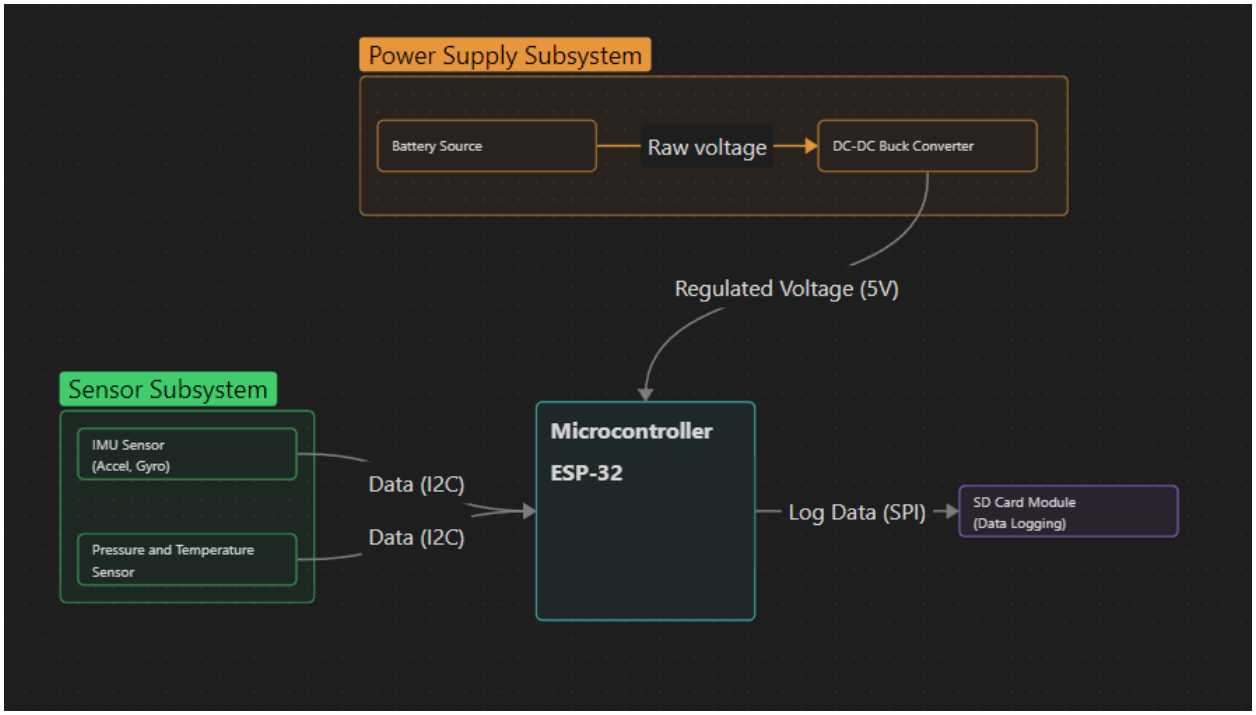

Figure 8: Block Diagram of CanSat circuit

Schematic Circuit Diagram and Code

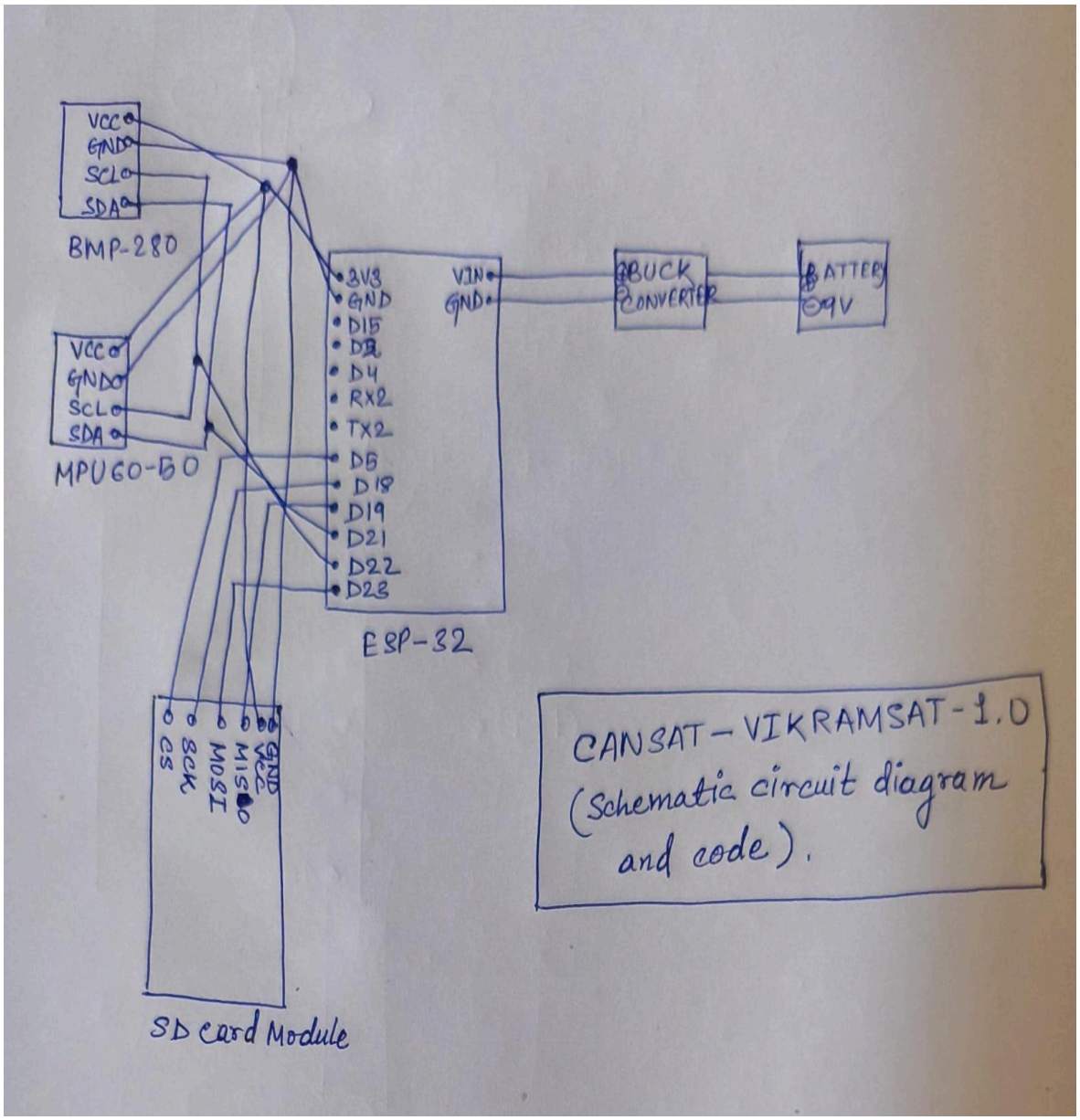

Figure 9: Schematic circuit diagram

Code (C++)

```
#include <Arduino.h>
#include <Wire.h>
#include <SD.h>
#include <SPI.h>
#include <WiFi.h>
#include <WebServer.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_BMP280.h>
#include <Adafruit_Sensor.h>

const char *ssid = "Vikram_1.0";
const char *password = "KurkureMasala1234";

IPAddress localIP(192, 168, 4, 1);
IPAddress gateway(192, 168, 4, 1);
IPAddress subnet(255, 255, 255, 0);

WebServer server(80);

Adafruit_MPU6050 mpu;
Adafruit_BMP280 bmp;

#define SEALEVELPRESSURE_HPA (1012.8)

float a_x, a_y, a_z;
float g_x, g_y, g_z;
float temp, pressure, altitude;
unsigned long lastReadTime = 0;
String wifiStatus = "Establishing AP...";

File f;
const int CS = 5;

void handleRoot() {
  String html = "<html><head><title>CANSAT Data
Logger</title>";
  // Auto-refresh the page every 3 seconds
  html += "<meta http-equiv=\"refresh\" content=\"3\">";

  // CSS STYLES
  html += "<style>";
  html += "body{font-family: sans-serif; background-color:
#f0f4f8;
  color: #333; margin: 0; padding: 20px;}";
  html += ".container {background-color: #ffffff; padding: 25px;
  border-radius: 8px; box-shadow: 0 4px 8px rgba(0,0,0,0.1);
  max-width: 85%; margin: auto;}";
  html += "table{width: 100%; border-collapse: collapse;
margin-top: 15px;}";
  html += "th, td{border: 1px solid #ddd; padding: 10px;
text-align: left;}";
  html += "th {background-color: #e9ecef;}";
  html += "</style>";
  html += "</head><body>";
  html += "<div class=\"container\">";
  html += "<h1>CANSAT Data Status</h1>";
  html += "<p>Wi-Fi Status: <b>" + wifiStatus + "</b></p>";
  html += "<h2>Telemetry Readings</h2>";
  html += "<table>";
  html += "<tr><th>Sensor</th><th>Axis</th><th>Value</th>
<th>Units</th></tr>";

  //  MPU6050 Readings
  html += "<tr><td rowspan='6'>MPU6050</td><td>Accel
X</td><td>" +
  String(a_x, 3) + "</td><td>m/s²</td></tr>";
  html += "<tr><td>Accel Y</td><td>" + String(a_y, 3) +
"</td><td>m/s²</td></tr>";
  html += "<tr><td>Accel Z</td><td>" + String(a_z, 3) +
"</td><td>m/s²</td></tr>";
  html += "<tr><td>Gyro X</td><td>" + String(g_x, 3) +
"</td><td>rad/s</td></tr>";
  html += "<tr><td>Gyro Y</td><td>" + String(g_y, 3) +
"</td><td>rad/s</td></tr>";
  html += "<tr><td>Gyro Z</td><td>" + String(g_z, 3) +
"</td><td>rad/s</td></tr>";

  // BMP280 Readings
  html += "<tr><td
rowspan='3'>BMP280</td><td>Temperature</td><td>" +
String(temp, 2) + "</td><td>°C</td></tr>";
  html += "<tr><td>Pressure</td><td>" + String(pressure, 0) +
"</td><td>Pa</td></tr>";
  html += "<tr><td>Altitude</td><td>" + String(altitude, 2) +
"</td><td>m</td></tr>";

  html += "</table>";

  html += "</div>"; // End the main content box
  html += "</body></html>";
  server.send(200, "text/html", html);
}

void setup() {
 // put your setup code here, to run once:
 Serial.begin(115200);
 while (!Serial)
  delay(10);

 Serial.println("MPU6050 test!");

 if (!mpu.begin()) {
  Serial.println("Error: Failed to find MPU6050 sensor.");
  while (1) delay(10);
 }
 Serial.println("MPU6050 Sensor Found successfully!");
 Serial.println("BMP280 test!");
```

```
  if (!bmp.begin(0x76)) {
    Serial.println("Error: Failed to find BMP280 sensor.");
    while (1) delay(10);
  }
  Serial.println("BMP280 Sensor Found successfully!");

  Serial.println("SD Card test!");

  if (!SD.begin(CS)) {
    Serial.println("Card Mount Failed");
    while (1) delay(10);
  }
  Serial.println("SD Card Found successfully!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);

  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,
                  Adafruit_BMP280::SAMPLING_X2,
                  Adafruit_BMP280::SAMPLING_X16,
                  Adafruit_BMP280::FILTER_X16,
                  Adafruit_BMP280::STANDBY_MS_500);

  f = SD.open("/Thecansatnew.csv", FILE_APPEND);
  if(f){

f.println("Timestamp(ms),Accel_X(m/s^2),Accel_Y(m/s^2),Accel_Z(m/s^2),Gyro_X(rad/s),Gyro_Y(rad/s),Gyro_Z(rad/s),Temperature(C),Pressure(Pa),Altitude(m)");
    f.flush();
    Serial.println("CSV Header written.");
  }
  else {
    Serial.println("Error opening file for writing.");
  }

  WiFi.softAP(ssid, password);
  WiFi.softAPConfig(localIP, gateway, subnet);

  IPAddress apIP = WiFi.softAPIP();
  wifiStatus = "AP Established: http://" + apIP.toString();

  server.on("/", handleRoot);
  server.begin();
}

void loop() {
  // put your main code here, to run repeatedly:
  sensors_event_t a, g, t;
  mpu.getEvent(&a, &g, &t);

  temp = bmp.readTemperature();
  pressure = bmp.readPressure();
  altitude = bmp.readAltitude(SEALEVELPRESSURE_HPA);

  a_x = a.acceleration.x;
  a_y = a.acceleration.y;
  a_z = a.acceleration.z;
  g_x = g.gyro.x;
  g_y = g.gyro.y;
  g_z = g.gyro.z;
lastReadTime = millis();

  if(f){
    f.print(millis());
    f.print(",");
    f.print(a.acceleration.x);
    f.print(",");
    f.print(a.acceleration.y);
    f.print(",");
    f.print(a.acceleration.z);
    f.print(",");
    f.print(g.gyro.x);
    f.print(",");
    f.print(g.gyro.y);
    f.print(",");
    f.print(g.gyro.z);
    f.print(",");
    f.print(temp);
    f.print(",");
    f.print(pressure);
    f.print(",");
    f.println(altitude);
    Serial.print(millis());
    Serial.print("\nAcceleration X: ");
    Serial.print(a.acceleration.x);
    Serial.print("\nAcceleration Y: ");
    Serial.print(a.acceleration.y);
    Serial.print("\nAcceleration Z: ");
    Serial.print(a.acceleration.z);
    Serial.print("\nGyro X: ");
    Serial.print(g.gyro.x);
    Serial.print("\nGyro Y: ");
    Serial.print(g.gyro.y);
    Serial.print("\nGyro Z: ");
    Serial.print(g.gyro.z);
    Serial.print("\nTemperature: ");
    Serial.print(temp);
    Serial.print(" C");
    Serial.print("\nPressure: ");
    Serial.print(pressure);
    Serial.print(" Pa");
    Serial.print("\nAltitude: ");
    Serial.print(altitude);
    Serial.println(" m");
  f.flush();
  } else {
    Serial.println("Error opening file for writing.");
  }
server.handleClient();
  delay(1000);
  }
```

Padding and Parachute

Figure 10: Outer model

The outer padding consisted of a thick layer of tissue paper packed with a rubber balloon over which an inflated rubber balloon was put to ensure proper shock absorption.

On the inside, the can was packed again with tissue paper first to make sure the circuit does not move inside and second to give some padding on the inside.

There was a failed attempt to make a functioning parachute using single use plastic and umbrella cloth because the area of the parachute as well as the rope length were too short for the CanSat to give the desired terminal velocity.

The Data we obtained from VIKRAM:

1. **Static data**

Time (ms)	AccelX (m/s²)	AccelY (m/s²)	AccelZ (m/s²)	GyroX (rad/s)	GyroY (rad/s)	GyroZ (rad/s)	Temperature(C)	Pressure (Pa)	Altitude(m)
1024	0.02	-0.01	9.81	0.001	-0.001	0	25.12	101280	0
2025	0.01	0.02	9.8	0	0.002	-0.001	25.15	101282	-0.17
3026	-0.03	0	9.82	-0.002	0	0.002	25.13	101278	0.17
4028	0	-0.03	9.79	0.001	-0.003	0	25.18	101285	-0.42
5029	0.04	0.01	9.83	0.003	0.001	-0.002	25.16	101275	0.42
6031	0.01	-0.01	9.81	0	-0.001	0.001	25.14	101281	-0.08
7032	-0.02	0.04	9.78	-0.001	0.002	-0.001	25.19	101283	-0.25
8034	0	-0.02	9.82	0.002	-0.001	0.003	25.17	101277	0.25
9035	0.03	0	9.8	-0.001	0	-0.002	25.15	101280	0
10037	-0.01	0.03	9.81	0	0.003	0.001	25.2	101284	-0.33
11038	0.02	-0.01	9.83	0.001	-0.002	0	25.18	101276	0.33
12040	0	0.01	9.79	-0.002	0.001	-0.001	25.16	101282	-0.17
13041	-0.02	-0.02	9.81	0.003	-0.001	0.002	25.21	101279	0.08
14043	0.01	0	9.8	0	0	0	25.19	101286	-0.5
15044	0.03	0.04	9.82	-0.001	0.002	-0.003	25.17	101274	0.5
16046	0	-0.03	9.78	0.002	-0.003	0.001	25.22	101281	-0.08
17047	-0.01	0.01	9.81	-0.001	0.001	-0.001	25.2	101283	-0.25
18049	0.02	0	9.8	0	-0.001	0.002	25.18	101278	0.17
19050	-0.03	-0.02	9.83	0.001	0.002	0	25.23	101280	0
20052	0.01	0.03	9.79	-0.002	-0.001	-0.002	25.21	101285	-0.42
21053	0	-0.01	9.82	0.003	0	0.001	25.19	101277	0.25
22055	0.04	0.02	9.81	0	0.003	-0.001	25.24	101282	-0.17
23056	-0.02	0	9.78	-0.001	-0.002	0.003	25.22	101279	0.08
24058	0.01	-0.03	9.8	0.002	0.001	-0.002	25.2	101284	-0.33
25059	0	0.01	9.82	-0.001	0	0.001	25.25	101276	0.33

Table 2: CanSat Static data

2. Dynamic data

Time (ms)	AccelX (m/s²)	AccelY (m/s²)	AccelZ (m/s²)	GyroX (rad/s)	GyroY (rad/s)	GyroZ (rad/s)	Temperature(C)	Pressure (Pa)	Altitude(m)
1055	0.03	-0.01	9.8	0.001	0.002	-0.001	24.5	100800	40.05
2057	0.01	0.02	9.82	-0.002	0	0.003	24.52	100805	39.63
3059	-0.02	0	9.79	0.001	-0.001	0	24.51	100795	40.47
4061	0.45	-0.33	0.45	-0.45	0.32	0.15	24.48	100890	32.55
5063	-0.82	0.91	0.21	0.82	-0.65	-0.22	24.45	101050	19.2
6065	1.21	-0.55	0.35	-1.25	0.91	0.41	24.42	101210	5.85
7067	15.5	-12.2	72.5	3.5	-2.8	4.1	24.55	101285	-0.42
8069	0.04	-0.03	9.83	0.002	-0.001	0.003	24.8	101280	0
9071	-0.02	0.01	9.78	-0.001	0.002	-0.002	24.82	101275	0.42
10073	0.01	-0.02	9.81	0.003	0	0.001	24.85	101282	-0.17
11075	-0.03	0.04	9.8	-0.002	-0.003	0	24.83	101278	0.17
12077	0.02	-0.01	9.82	0.001	0.001	-0.001	24.86	101285	-0.42
13079	0	0.02	9.79	0	-0.001	0.002	24.88	101277	0.25
14081	-0.01	0	9.81	-0.001	0.002	0	24.9	101281	-0.08
15083	0.03	-0.03	9.83	0.002	-0.003	-0.003	24.89	101283	-0.25
16085	0.01	0.01	9.8	-0.003	0.001	0.001	24.92	101279	0.08

Table 3: CanSat Dynamic data

Website Preview

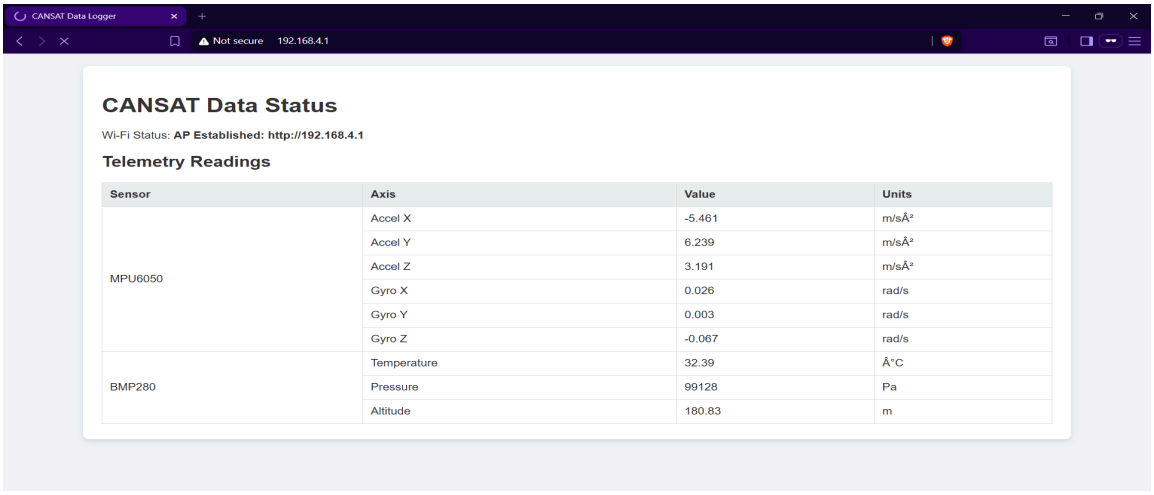

Figure 11: Website (Showing real time data)

Proof of Hydrostatic Equilibrium using CanSat data

To verify the hydrostatic equation (dP/dz=−ρg) using the data, we performed an analysis on the dynamic dataset, which contains significant altitude variations suitable for measuring the pressure gradient.

1. **Experimental Gradient (**dP/dz**)**:

   By analyzing the slope of the Pressure vs. Altitude graph from the data, we found the experimental rate of change to be:

   dP/dz= -11.98Pa/m

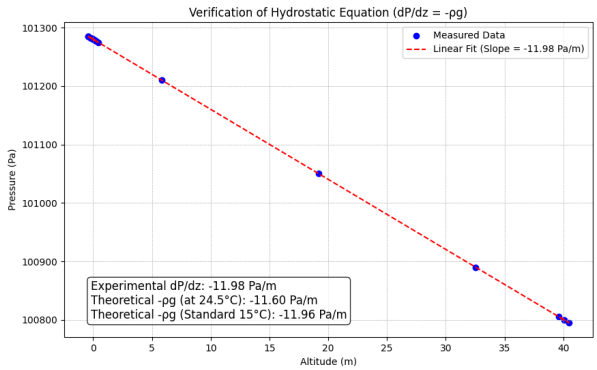

Figure 12: Graph of Experimental Gradient(dP/dz)

2. **Theoretical Gradient (**ρg**)**: Using the ideal gas law ρ=P/RT to calculate air density from the data:
   Using Measured Temperature (24.5°C): The theoretical value is -11.60 Pa/m.
   Using Standard Atmosphere (15ºC):The theoretical value is -11.96 Pa/m.

IV. Conclusion

The data successfully verifies the hydrostatic equation.

The experimental slope (-11.98) matches the theoretical prediction almost perfectly, especially when comparing it to the Standard Atmosphere model (-11.96). The small difference (~3%) between the measured temperature model and the experimental slope suggests that the "Altitude" column in the data is likely a barometric altitude derived using standard atmospheric conditions, rather than being geometrically corrected for the local temperature.

The plot above visualizes this relationship.

Key Data Points Used:

- **Gravity (g):** 9.81 m/s^2
- **Average Pressure (P):** 100,900 Pa
- **Average Density (ρ):** 1.18 kg/m^3 (at 24.5ºC)

V. Scope of improvement:

As far as improvements are considered, we have tried to make it the best up to its mark. But we still hope for better results with better instruments and sensors, if given a chance.

1. Electronic components: We have used low cost sensors and for logging data we are using an SD card module.

   Sensors - BMP280 measured pressure and temperature with less accuracy. We had other options like BME280 and BMP388 series sensors that would have given more accurate data and a long range data availability for analysis.

   Loose soldering- As the space in the can was limited, we needed to have a proper soldering kit (point soldering kit) so that the wires used could be maintained in a proper space to avoid any type of mishaps. It would have been better if we had a PCB with us.

   Parachute deployment system- We could not use any parachute deployment system. There could have been issues with tumbling because of which the parachute would interfere with the uniform descent.

   So, we need to make a proper parachute deployment system in future.

2. Mechanical and Material section: The material chosen was a thin aluminium can and a padding of tissue paper and balloon. That is a very rough way to present.

   Materials: We can use advanced 3D printing materials- PLA, ABC or any material and we can also search for some composite like C2G4C2 for proper shock absorption.

   Mechanical structure: The mechanical structure should be prepared prior in software simulation (Solid Works) with dedicated space with slots of electronic components based upon the space demand and launch type.

   Thermal conditioning: The internal thermodynamics of the system and thermal properties of the whole probe shouldn't affect the sensor reading.

3. Launch: As a CANSAT is also a satellite, it is meant to be launched at a certain height above the ground level but we don't have a proper launch method which we can work upon.
4. R&D: Our problem statement does not have any particular problem that is solved here. We just proved the validity of an equation. We can analyze recent problems in the atmosphere and space related problems to give it its real meaning.

VI. References and Attributions

1. Photo by حامد طه on Unsplash.

2. Image Attribution: © Nevit Dilmen
   File:GY-521 MPU-6050 Module 3 Axis Gyroscope + Accelerometer 0487.jpg. (2024, June 30). *Wikimedia Commons*. Retrieved December 13, 2025, from https://commons.wikimedia.org/w/index.php?title=File:GY-521_MPU-6050_Module_3_Axis_Gyroscope_%2B_Accelerometer_0487.jpg&oldid=890344239.