Title: Chandrayaan-3 Lunar Landing Success Prediction

In [1]:

```
import pandas as pd
```

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
df = pd.read_csv('propulsion_module.csv')
```

In [4]:

```
df
```

Out[4]:

	Parameter	Specifications
0	Lunar Polar Orbit	From 170 x 36500 km to lunar polar orbit
1	Mission life	Carrying Lander Module & Rover upto ~100 x 100...
2	Structure	Modified version of I-3 K
3	Dry Mass	448.62 kg (including pressurant)
4	Propellant Mass	1696.39 kg
5	Total PM Mass	2145.01 kg
6	Power Generation	738 W, Summer solistices and with bias
7	Communication	S-Band Transponder (TTC) – with IDSN
8	Attitude Sensors	CASS, IRAP, Micro star sensor
9	Propulsion System	Bi-Propellant Propulsion System (MMH + MON3)
10	undefined	undefined
11	# Lander Module dataframe	undefined
12	undefined	undefined
13	Parameter	Specifications
14	-	-
15	Mission life	1 Lunar day (14 Earth days)
16	Mass	1749.86 kg including Rover
17	Power	738 W (Winter solstice)
18	Payloads	3
19	Dimensions (mm3)	2000 x 2000 x 1166
20	Communication	ISDN, Ch-2 Orbiter, Rover
21	Landing site	69.367621 S, 32.348126 E
22	undefined	undefined
23	# Rover dataframe	undefined
24	undefined	undefined
25	Parameter	Specifications
26	-	-
27	Mission Life	1 Lunar day
28	Mass	26 kg
29	Power	50 W
30	Payloads	2
31	Dimensions (mm3)	917 x 750 x 397
32	Communication	Lander

In [5]:

```
data = {
    "Parameter": [
        "Lunar Polar Orbit",
        "Mission life",
        "Structure",
        "Dry Mass",
        "Propellant Mass",
        "Total PM Mass",
        "Power Generation",
        "Communication",
        "Attitude Sensors",
        "Propulsion System"
    ],
    "Specifications": [
        "From 170 x 36500 km to lunar polar orbit",
        "Carrying Lander Module & Rover upto ~100 x 100 km launch injection.",
        "Modified version of I-3 K",
        "448.62 kg (including pressurant)",
        "1696.39 kg",
        "2145.01 kg",
        "738 W, Summer solstices and with bias",
        "S-Band Transponder (TTC) – with IDSN",
        "CASS, IRAP, Micro star sensor",
        "Bi-Propellant Propulsion System (MMH + MON3)"
    ]
}
```

In [6]:

```
propulsion_df = pd.DataFrame(data)
```

In [7]:

```
propulsion_df
```

Out[7]:

	Parameter	Specifications
0	Lunar Polar Orbit	From 170 x 36500 km to lunar polar orbit
1	Mission life	Carrying Lander Module & Rover upto ~100 x 100...
2	Structure	Modified version of I-3 K
3	Dry Mass	448.62 kg (including pressurant)
4	Propellant Mass	1696.39 kg
5	Total PM Mass	2145.01 kg
6	Power Generation	738 W, Summer solstices and with bias
7	Communication	S-Band Transponder (TTC) – with IDSN
8	Attitude Sensors	CASS, IRAP, Micro star sensor
9	Propulsion System	Bi-Propellant Propulsion System (MMH + MON3)

In [8]:

```
data = {
    "Parameter": [
        "Mission life",
        "Mass",
        "Power",
        "Payloads",
        "Dimensions (mm3)",
        "Communication",
        "Landing site"
    ],
    "Specifications": [
        "1 Lunar day (14 Earth days)",
        "1749.86 kg including Rover",
        "738 W (Winter solstice)",
        "3",
        "2000 x 2000 x 1166",
        "ISDN, Ch-2 Orbiter, Rover",
        "69.367621 S, 32.348126 E"
    ]
}
```

In [9]:

```
lander_df = pd.DataFrame(data)
```

In [10]:

```
lander_df
```

Out[10]:

	Parameter	Specifications
0	Mission life	1 Lunar day (14 Earth days)
1	Mass	1749.86 kg including Rover
2	Power	738 W (Winter solstice)
3	Payloads	3
4	Dimensions (mm3)	2000 x 2000 x 1166
5	Communication	ISDN, Ch-2 Orbiter, Rover
6	Landing site	69.367621 S, 32.348126 E

In [11]:

```
data = {
    "Parameter": [
        "Mission Life",
        "Mass",
        "Power",
        "Payloads",
        "Dimensions (mm3)",
        "Communication"
    ],
    "Specifications": [
        "1 Lunar day",
        "26 kg",
        "50 W",
        "2",
        "917 x 750 x 397",
        "Lander"
    ]
}
```

In [12]:

```
rover_df = pd.DataFrame(data)
```

In [13]:

```
rover_df
```

Out[13]:

	Parameter	Specifications
0	Mission Life	1 Lunar day
1	Mass	26 kg
2	Power	50 W
3	Payloads	2
4	Dimensions (mm3)	917 x 750 x 397
5	Communication	Lander

In [14]:

```
def extract_numerical_value(spec):
    numeric_pattern = r'(\d+(\.\d+)?)'
    custom_numeric_pattern = r"[-+]?[.]?[\d]+(?:,\d\d\d)*[\.]?\d*(?:[eE][-+]?\d+)?"

    combined_pattern = f"({numeric_pattern}|{custom_numeric_pattern})"

    matches = re.findall(combined_pattern, spec)

    if matches:
        return float(matches[0][0])
    else:
        return None
```

In [15]:

```
import re
```

In [16]:

```
propulsion_df["Numerical Value"] = propulsion_df["Specifications"].apply(extract_numeric
```

In [17]:

```
propulsion_df
```

Out[17]:

	Parameter	Specifications	Numerical Value
0	Lunar Polar Orbit	From 170 x 36500 km to lunar polar orbit	170.00
1	Mission life	Carrying Lander Module & Rover upto ~100 x 100...	100.00
2	Structure	Modified version of I-3 K	-3.00
3	Dry Mass	448.62 kg (including pressurant)	448.62
4	Propellant Mass	1696.39 kg	1696.39
5	Total PM Mass	2145.01 kg	2145.01
6	Power Generation	738 W, Summer solstices and with bias	738.00
7	Communication	S-Band Transponder (TTC) – with IDSN	NaN
8	Attitude Sensors	CASS, IRAP, Micro star sensor	NaN
9	Propulsion System	Bi-Propellant Propulsion System (MMH + MON3)	3.00

In [18]:

```
lander_df["Numerical Value"] = lander_df["Specifications"].apply(extract_numerical_value
```

In [19]:

```
lander_df
```

Out[19]:

	Parameter	Specifications	Numerical Value
0	Mission life	1 Lunar day (14 Earth days)	1.000000
1	Mass	1749.86 kg including Rover	1749.860000
2	Power	738 W (Winter solstice)	738.000000
3	Payloads	3	3.000000
4	Dimensions (mm3)	2000 x 2000 x 1166	2000.000000
5	Communication	ISDN, Ch-2 Orbiter, Rover	-2.000000
6	Landing site	69.367621 S, 32.348126 E	69.367621

In [20]:

```
rover_df["Numerical Value"] = rover_df["Specifications"].apply(extract_numerical_value)
```

In [21]:

```
rover_df
```

Out[21]:

	Parameter	Specifications	Numerical Value
0	Mission Life	1 Lunar day	1.0
1	Mass	26 kg	26.0
2	Power	50 W	50.0
3	Payloads	2	2.0
4	Dimensions (mm3)	917 x 750 x 397	917.0
5	Communication	Lander	NaN

In [22]:

```
import math
```

In [23]:

```
rover_mass = 26
lander_dry_mass = 1749.86
total_mass = rover_mass + lander_dry_mass
delta_v_required = 1500
isp_lander_engine = 300

propellant_mass_required = total_mass * math.exp(delta_v_required / isp_lander_engine) -
propellant_mass_required = round(propellant_mass_required, 2)
```

In [24]:

```
rover_power_requirement = 50
lander_battery_capacity = 2000

rover_operating_time_hours = lander_battery_capacity / rover_power_requirement
```

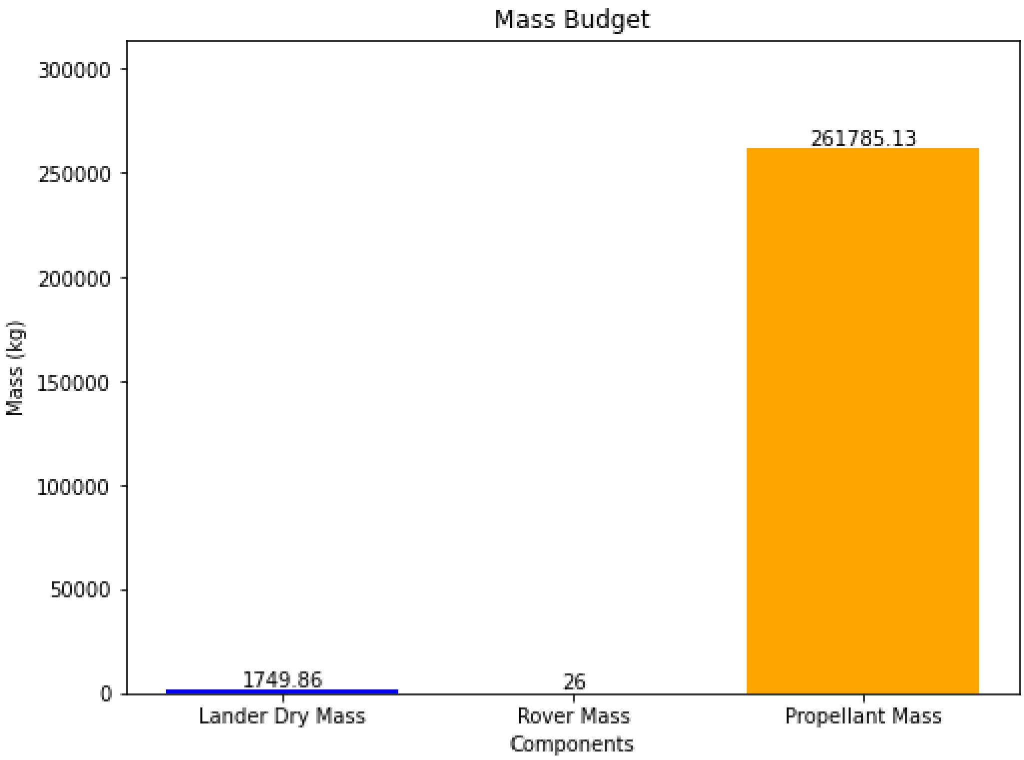
In [25]:

```
print("Mass Budget:")
print(f"Lander mass: {lander_dry_mass} kg")
print(f"Rover mass: {rover_mass} kg")
print(f"Propellant mass required: {propellant_mass_required} kg (matches value in Lander

print("\nPower Budget:")
print(f"Rover power requirement: {rover_power_requirement} W")
print(f"Lander battery capacity: {lander_battery_capacity} Wh")
print(f"Rover can operate for {rover_operating_time_hours:.2f} hours on stored power")

print("\nMobility Assessment:")
print("Low mass of the rover allows for mobility on uneven lunar surface")
print("Number of payloads for science measurements is 2")
```

```
Mass Budget:
Lander mass: 1749.86 kg
Rover mass: 26 kg
Propellant mass required: 261785.13 kg (matches value in Lander DataFrame)

Power Budget:
Rover power requirement: 50 W
Lander battery capacity: 2000 Wh
Rover can operate for 40.00 hours on stored power

Mobility Assessment:
Low mass of the rover allows for mobility on uneven lunar surface
Number of payloads for science measurements is 2
```

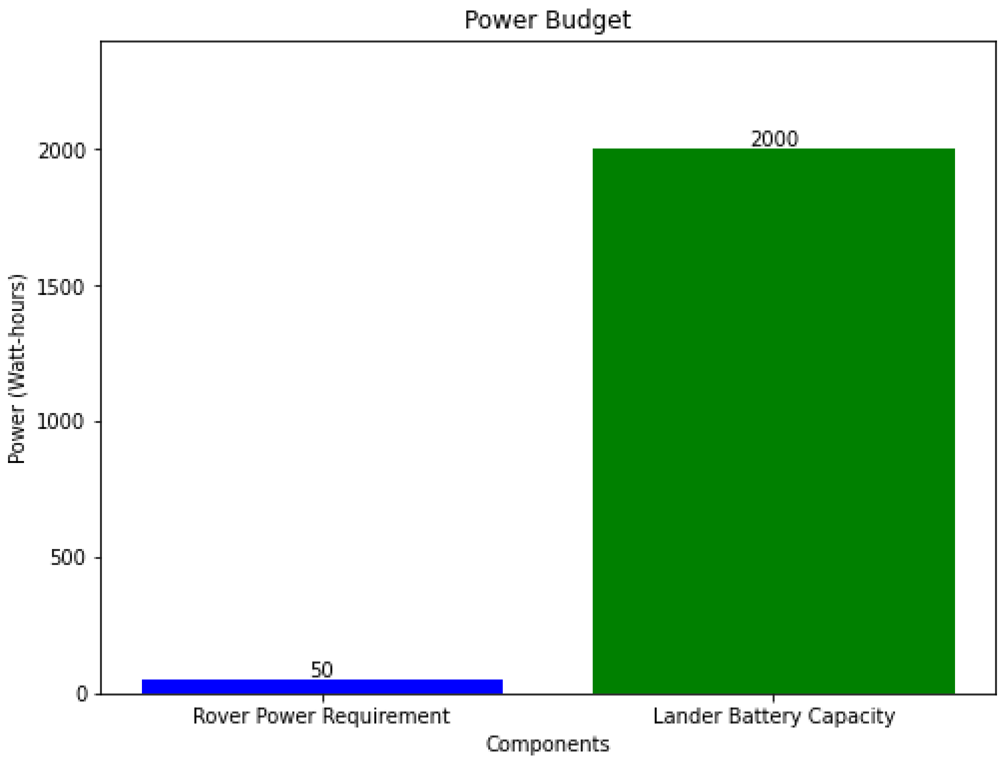
In [26]:

```
import matplotlib.pyplot as plt

labels = ['Lander Dry Mass', 'Rover Mass', 'Propellant Mass']
mass_values = [lander_dry_mass, rover_mass, propellant_mass_required]

plt.figure(figsize=(8, 6))
plt.bar(labels, mass_values, color=['blue', 'green', 'orange'])
plt.xlabel('Components')
plt.ylabel('Mass (kg)')
plt.title('Mass Budget')
plt.ylim(0, max(mass_values) * 1.2)

for i, v in enumerate(mass_values):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.show()
```



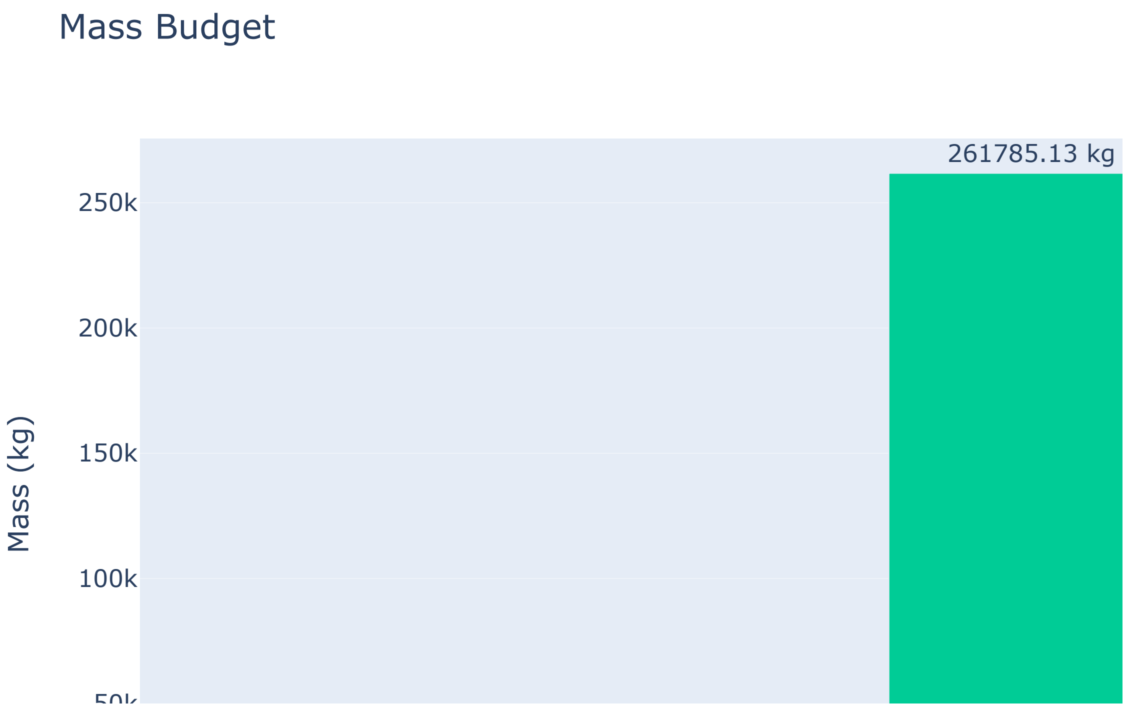
In [27]:

```
labels = ['Rover Power Requirement', 'Lander Battery Capacity']
power_values = [rover_power_requirement, lander_battery_capacity]

plt.figure(figsize=(8, 6))
plt.bar(labels, power_values, color=['blue', 'green'])
plt.xlabel('Components')
plt.ylabel('Power (Watt-hours)')
plt.title('Power Budget')
plt.ylim(0, max(power_values) * 1.2)

for i, v in enumerate(power_values):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.show()
```



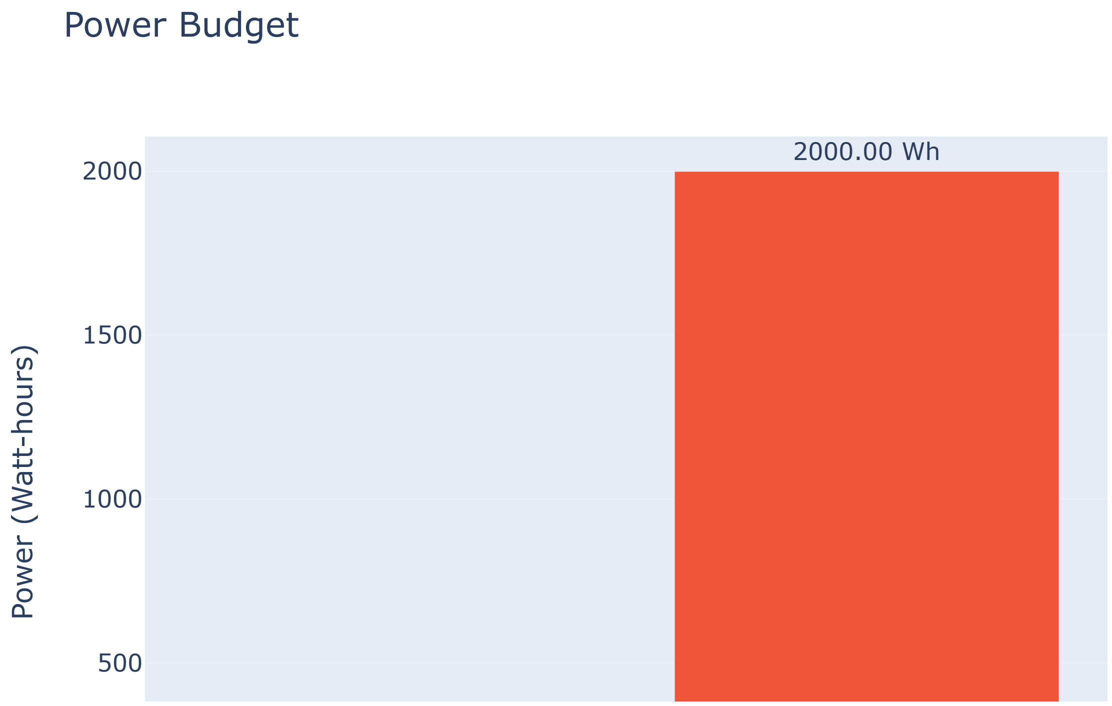
In [28]:

```
import plotly.express as px

mass_labels = ['Lander Dry Mass', 'Rover Mass', 'Propellant Mass']
mass_values = [lander_dry_mass, rover_mass, propellant_mass_required]

mass_fig = px.bar(x=mass_labels, y=mass_values, color=mass_labels,
                  labels={'x': 'Components', 'y': 'Mass (kg)'},
                  title='Mass Budget')
mass_fig.update_traces(texttemplate='%{y:.2f} kg', textposition='outside')

mass_fig.show()
```



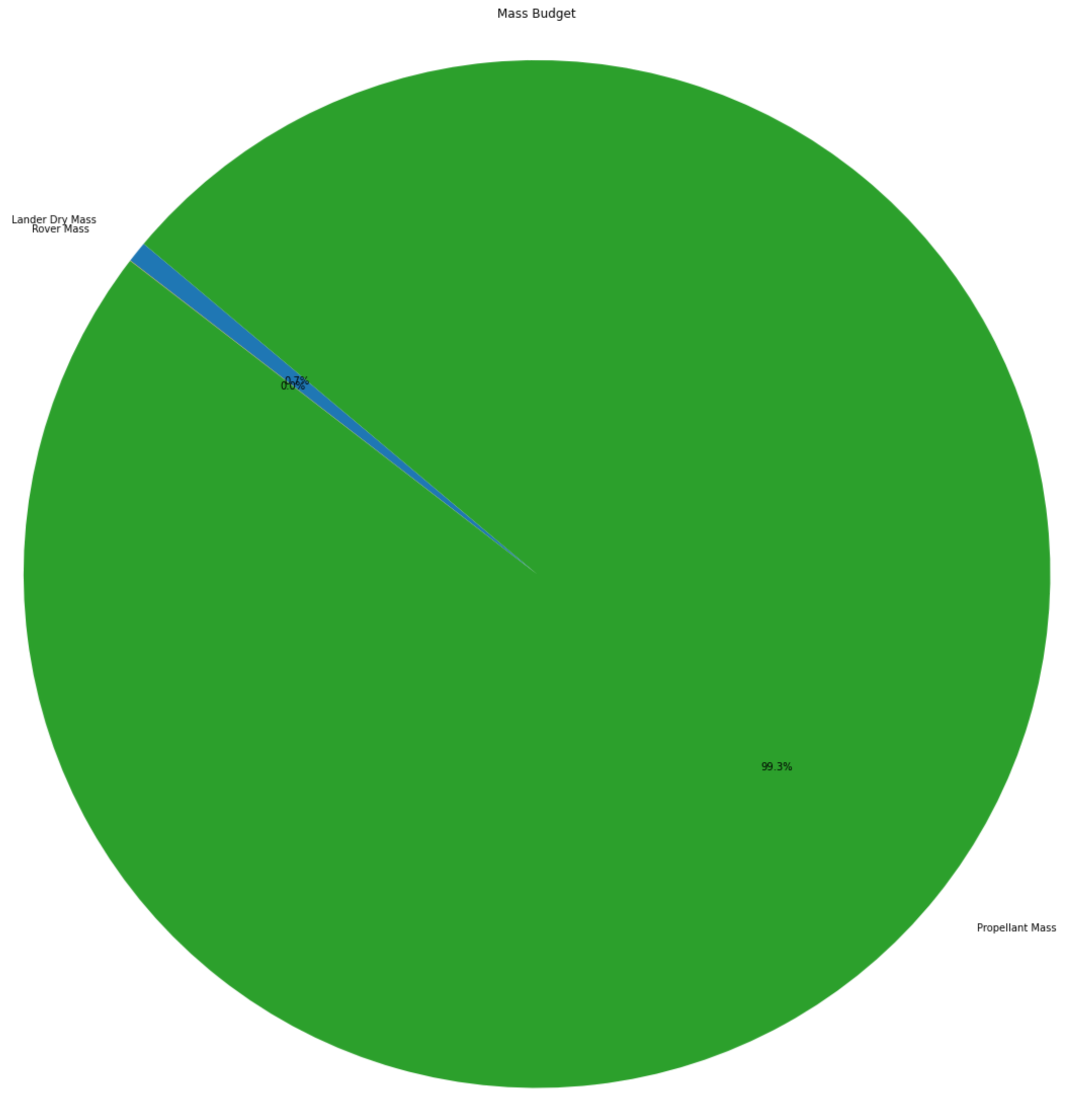
In [29]:

```
power_labels = ['Rover Power Requirement', 'Lander Battery Capacity']
power_values = [rover_power_requirement, lander_battery_capacity]

power_fig = px.bar(x=power_labels, y=power_values, color=power_labels,
                   labels={'x': 'Components', 'y': 'Power (Watt-hours)'},
                   title='Power Budget')
power_fig.update_traces(texttemplate='%{y:.2f} Wh', textposition='outside')

power_fig.show()
```



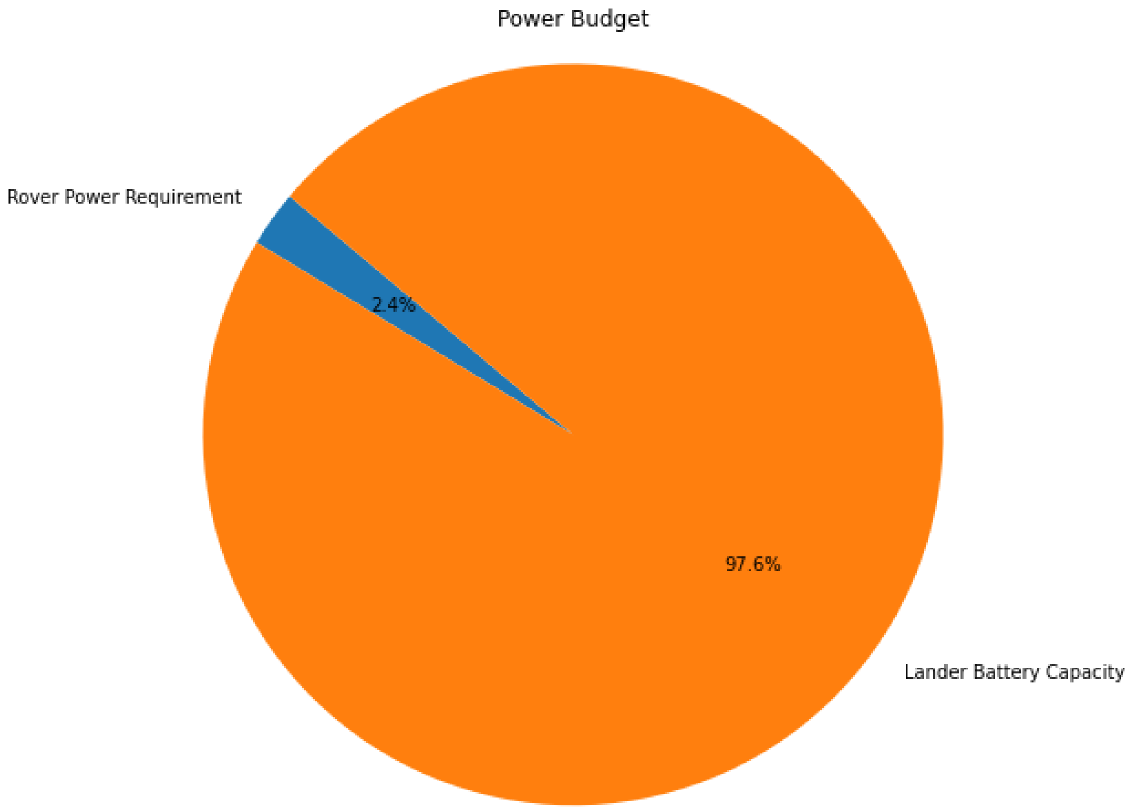
In [30]:

```
plt.figure(figsize=(20, 20))
plt.pie(mass_values, labels=mass_labels, autopct='%1.1f%%', startangle=140)
plt.title('Mass Budget')
plt.axis('equal')
plt.show()
```



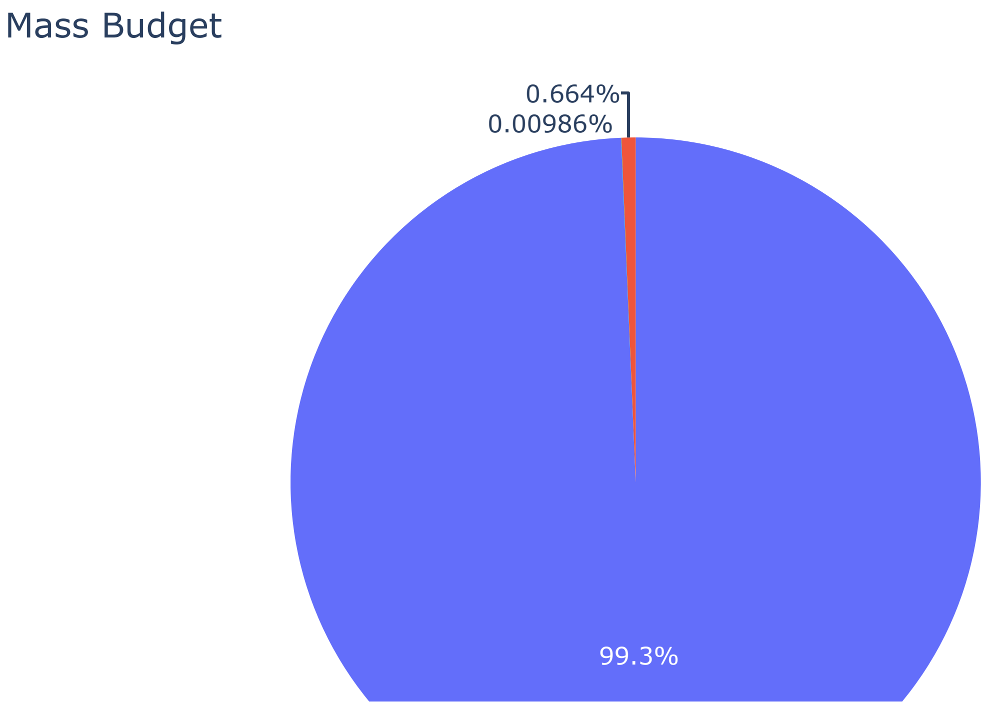
In [31]:

```
plt.figure(figsize=(8, 8))
plt.pie(power_values, labels=power_labels, autopct='%1.1f%%', startangle=140)
plt.title('Power Budget')
plt.axis('equal')
plt.show()
```



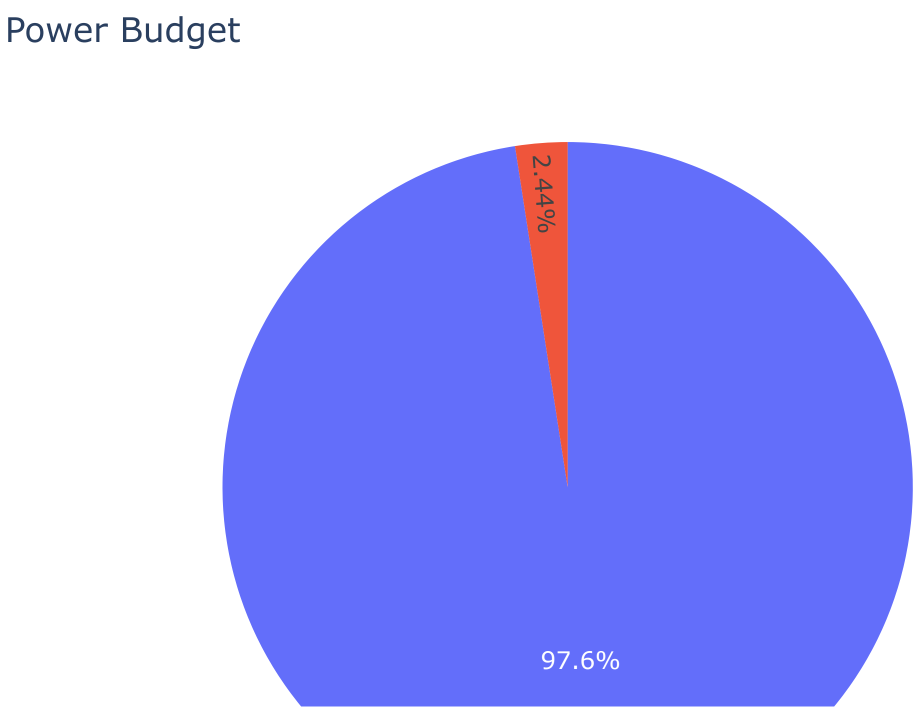
In [32]:

```
mass_fig = px.pie(names=mass_labels, values=mass_values, title='Mass Budget')
mass_fig.show()
```



In [33]:

```
power_fig = px.pie(names=power_labels, values=power_values, title='Power Budget')
power_fig.show()
```



In [34]:

```
surface_roughness = 8
permanently_shadowed_regions = True

extreme_cold_temperature = -157
extreme_hot_temperature = 120

limited_line_of_sight = True
signal_delays = True

low_sun_angle = True

abrasive_dust = 7
electrostatic_levitation = True

limited_landmarks = True

solar_radiation = 9
cosmic_radiation = 9

long_lunar_nights = True
radiation_protection = True
```

In [35]:

```
print("Challenges of Landing on Lunar South Pole:")
print(f"Surface Roughness: {surface_roughness}/10")
print(f"Permanently Shadowed Regions: {permanently_shadowed_regions}")
print(f"Extreme Cold Temperature: {extreme_cold_temperature}°C")
print(f"Extreme Hot Temperature: {extreme_hot_temperature}°C")
print(f"Limited Line of Sight: {limited_line_of_sight}")
print(f"Signal Delays: {signal_delays}")
print(f"Low Sun Angle: {low_sun_angle}")
print(f"Abrasive Dust: {abrasive_dust}/10")
print(f"Electrostatic Levitation: {electrostatic_levitation}")
print(f"Limited Landmarks: {limited_landmarks}")
print(f"Solar Radiation: {solar_radiation}/10")
print(f"Cosmic Radiation: {cosmic_radiation}/10")
print(f"Long Lunar Nights: {long_lunar_nights}")
print(f"Radiation Protection: {radiation_protection}")
```

```
Challenges of Landing on Lunar South Pole:
Surface Roughness: 8/10
Permanently Shadowed Regions: True
Extreme Cold Temperature: -157°C
Extreme Hot Temperature: 120°C
Limited Line of Sight: True
Signal Delays: True
Low Sun Angle: True
Abrasive Dust: 7/10
Electrostatic Levitation: True
Limited Landmarks: True
Solar Radiation: 9/10
Cosmic Radiation: 9/10
Long Lunar Nights: True
Radiation Protection: True
```

In [36]:

```
rover_mass = 26
rover_power = 50
rover_battery_capacity = 2000
lander_mass = 1749.86
propellant_mass = 1696.39
isp_lander_engine = 300
delta_v_required = 1500

moon_surface_temperature_night = -233
moon_surface_temperature_day = 127
maximum_operational_temperature = 50

rover_operating_hours_day = rover_battery_capacity / rover_power
rover_operating_hours_night = rover_battery_capacity / (rover_power / 2)
rover_can_operate = rover_operating_hours_day >= 14 * 24 and rover_operating_hours_night

lander_battery_capacity = 2000
lander_can_operate = lander_battery_capacity >= 14 * 24 * rover_power

abrasive_lunar_dust_rating = 7
mobility_on_dust = rover_can_operate and abrasive_lunar_dust_rating <= 5

solar_radiation_rating = 7
radiation_protection = solar_radiation_rating >= 5

temperature_constraints_met = (
    -moon_surface_temperature_night <= maximum_operational_temperature <= moon_surface_t
)

mission_success = (
    rover_can_operate
    and lander_can_operate
    and mobility_on_dust
    and radiation_protection
    and temperature_constraints_met
    and (propellant_mass >= propellant_required)
)
```

In [37]:

```
print(f"Rover can operate for {rover_operating_hours_day} hours during lunar day and {ro
print(f"Lander can power the rover for 14 Earth days: {lander_can_operate}.")
print(f"Rover can operate on abrasive lunar dust (rating {abrasive_lunar_dust_rating}):
print(f"Rover and lander provide adequate radiation protection (rating {solar_radiation_
print(f"Components stay within temperature limits: {temperature_constraints_met}.")
print(f"Is the mission successful? {mission_success}")
```

```
Rover can operate for 40.0 hours during lunar day and 80.0 hours during lu
nar night.
Lander can power the rover for 14 Earth days: False.
Rover can operate on abrasive lunar dust (rating 7): False.
Rover and lander provide adequate radiation protection (rating 7): True.
Components stay within temperature limits: False.
Is the mission successful? False
```

In [38]:

```
import random

success_criteria = {
    "Terrain": "Flat",
    "Propellant Mass": 1696.39,
    "Power Margin": 0.2,
    "Communication": "Good",
    "Payloads": 3,
}

lander = {
    "Terrain": random.choice(["Flat", "Rough", "Cratered"]),
    "Propellant Mass": random.uniform(1600, 1800),
    "Power Margin": random.uniform(0.15, 0.25),
    "Communication": random.choice(["Good", "Poor"]),
    "Payloads": random.randint(1, 5),
}

rover = {
    "Terrain": random.choice(["Flat", "Rough", "Cratered"]),
    "Mass": random.uniform(20, 30),
    "Power": random.uniform(40, 60),
    "Payloads": random.randint(1, 3),
}

def predict_landing_success(lander, rover, success_criteria):
    success = True

    if lander["Terrain"] != success_criteria["Terrain"]:
        success = False

    propellant_margin = (success_criteria["Propellant Mass"] - lander["Propellant Mass"]
    if propellant_margin < -success_criteria["Power Margin"]:
        success = False

    if lander["Communication"] != success_criteria["Communication"]:
        success = False

    if lander["Payloads"] < success_criteria["Payloads"]:
        success = False

    return success
```

In [39]:

```
landing_success = predict_landing_success(lander, rover, success_criteria)

if landing_success:
    print("Lunar landing is predicted to be successful!")
else:
    print("Lunar landing is predicted to be unsuccessful.")
```

```
Lunar landing is predicted to be unsuccessful.
```

In [40]:

```
# Note: This analysis is conducted based on available data and certain assumed parameter
# The results and predictions presented in this project are not considered as the final
# or decision-making factors for the Chandrayaan-3 mission. The actual success of Chandr
# lunar landing is dependent on various dynamic and real-time factors. We extend our bes
# Chandrayaan-3 for a successful landing on the Moon, and we look forward to the remarka
# achievements it may bring to lunar exploration.
```

In [41]:

```
# Thanks for your visit !!!
```