Session 2 Chamodi : Gait Analysis

Time : 2025-05-31 21:44:45.469000 to 2025-05-31 21:45:25.529000

Sensor ID : 601

Steps : 30

Fetching Raw Data from DynamoDB

In [40]:
```
import sys
import os
from datetime import datetime

# Append app to sys.path to access modules like app.utils.dynamo
sys.path.append(os.path.abspath("../app"))
```

In [41]:
```
from utils.dynamo import fetch_session_data
```

In [42]:
```
sensor_id = 601
start_time = datetime.fromisoformat("2025-05-31 23:09:16.722000")
end_time = datetime.fromisoformat("2025-05-31 23:09:40.685000")

data = fetch_session_data(sensor_id, start_time, end_time)
print(f"📦 Retrieved {len(data)} records")
```

```
🔍 [DEBUG] Fetching data for Sensor ID: 601
🕒 [DEBUG] Start Time: 2025-05-31 23:09:16.722000 -> 1748713156722
🕒 [DEBUG] End Time:   2025-05-31 23:09:40.685000 -> 1748713180685
📦 [DEBUG] Retrieved 238 items from DynamoDB.

📦 Retrieved 238 records
```

In [ ]:
```
data
```

Step 1 : Preprocess and Sort

Sort By Timestamp

In [ ]:
```
import pandas as pd
from decimal import Decimal

# Convert to DataFrame
df = pd.json_normalize(data)
```

```
# Convert all Decimal values to float (optional but useful)
df = df.map(lambda x: float(x) if isinstance(x, Decimal) else x)

# Sort by timestamp
df = df.sort_values(by="timestamp").reset_index(drop=True)

# Show first few rows
df.head()
```

Check for Missing FSR Values

In [45]:
```
required_fsrs = [f"FSR_{i}" for i in range(1, 17)]
missing = [col for col in required_fsrs if col not in df.columns]
print("Missing FSR columns:", missing)
```

```
Missing FSR columns: []
```

Clipping FSR values (0 to 4095)

In [46]:
```
for col in required_fsrs:
    df[col] = df[col].clip(0, 4095)
```

Normalizing Timestamps

In [47]:
```
df["time_sec"] = df["timestamp"] - df["timestamp"].min()
```

Replacing FSR_14 values with FSR_11

FSR_14 continously High since it is connected to GPIO 0

In [52]:
```
df["FSR_14"] = df["FSR_11"]
```

Foot Region to Sensor Mapping

Region Name	Sensors Included	Function in Gait Cycle
Forefoot	FSR_5, FSR_6, FSR_7, FSR_8, FSR_9, FSR_12, FSR_15, FSR_16	Toe-off, Push-off
Midfoot	FSR_1, FSR_2, FSR_10, FSR_11, FSR_13, FSR_14	Load-bearing, Mid-stance
Rearfoot	FSR_3, FSR_4	Heel strike, Initial contact

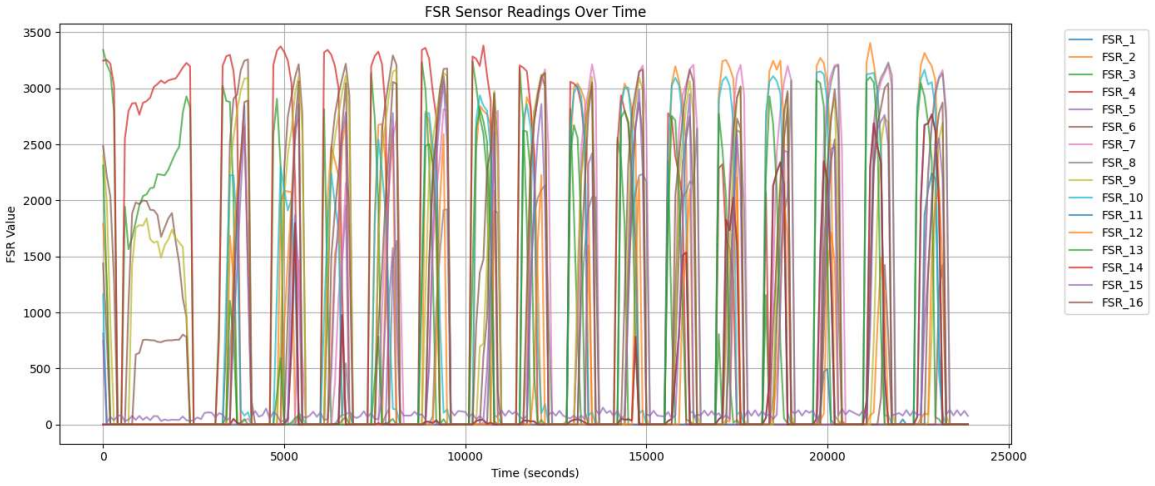
In [53]:
```
REGION_MAP = {
    "forefoot": [5, 6, 7, 8, 9, 12, 15, 16],
```

```
    "midfoot": [1, 2, 10, 11, 13,14],
    "rearfoot": [3, 4]
}
```

Step 2: Plotting Raw Sensor Readings

FSR Readings

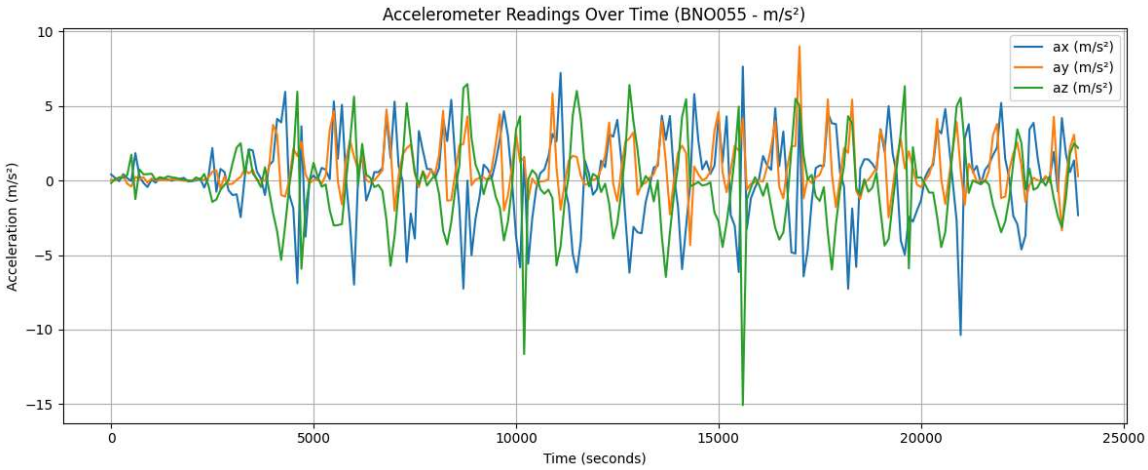
In [54]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(14, 6))

for i in range(1, 17):
    plt.plot(df["time_sec"], df[f"FSR_{i}"], label=f"FSR_{i}", alpha=0.

plt.xlabel("Time (seconds)")
plt.ylabel("FSR Value")
plt.title("FSR Sensor Readings Over Time")
plt.legend(bbox_to_anchor=(1.05, 1), loc="upper left", ncol=1)
plt.tight_layout()
plt.grid(True)
plt.show()
```



Accelrometer Readings

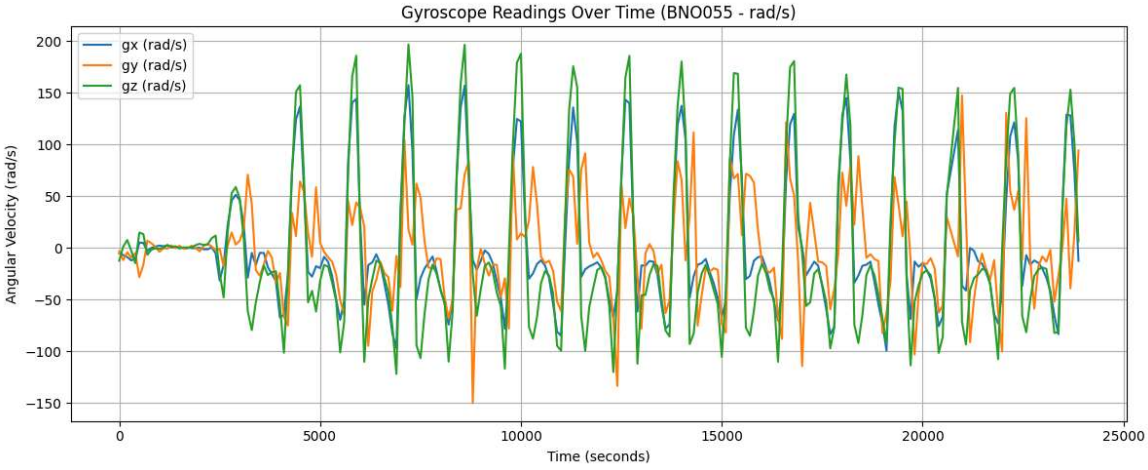
In [58]:
```
plt.figure(figsize=(12, 5))

plt.plot(df["time_sec"], df["ax"], label="ax (m/s²)")
plt.plot(df["time_sec"], df["ay"], label="ay (m/s²)")
plt.plot(df["time_sec"], df["az"], label="az (m/s²)")

plt.title("Accelerometer Readings Over Time (BNO055 - m/s²)")
plt.xlabel("Time (seconds)")
plt.ylabel("Acceleration (m/s²)")
plt.legend()
plt.grid(True)
```

```
plt.tight_layout()
plt.show()
```



Gyroscope Readings

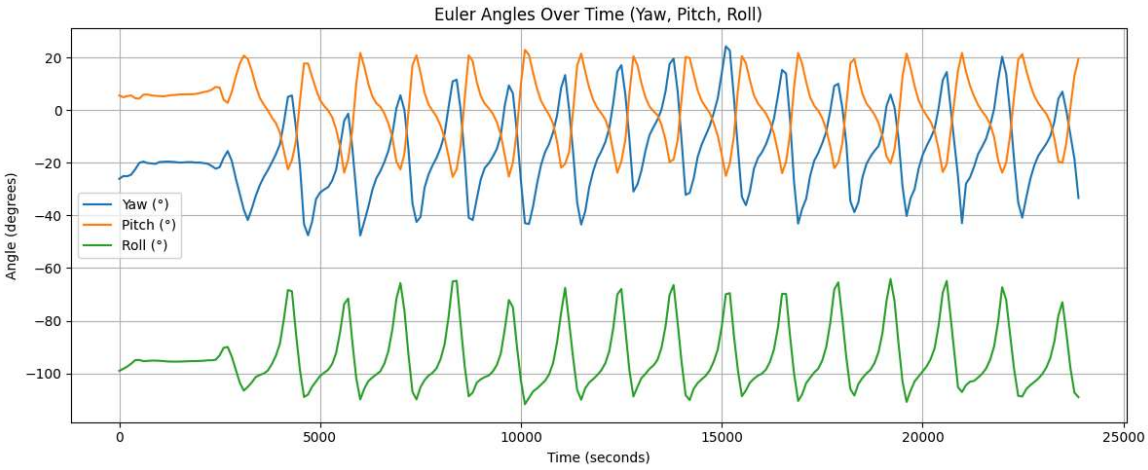
In [57]:
```
plt.figure(figsize=(12, 5))

plt.plot(df["time_sec"], df["gx"], label="gx (rad/s)")
plt.plot(df["time_sec"], df["gy"], label="gy (rad/s)")
plt.plot(df["time_sec"], df["gz"], label="gz (rad/s)")

plt.title("Gyroscope Readings Over Time (BNO055 - rad/s)")
plt.xlabel("Time (seconds)")
plt.ylabel("Angular Velocity (rad/s)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



Yaw Pitch Role Raw Readings

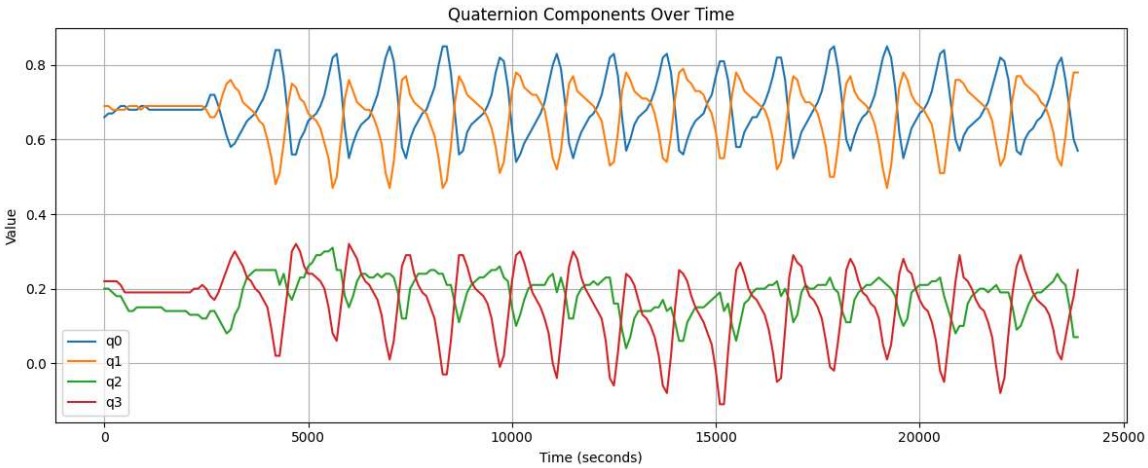
In [59]:
```
plt.figure(figsize=(12, 5))

plt.plot(df["time_sec"], df["yaw"], label="Yaw (°)")
plt.plot(df["time_sec"], df["pitch"], label="Pitch (°)")
```

```
plt.plot(df["time_sec"], df["roll"], label="Roll (°)")

plt.title("Euler Angles Over Time (Yaw, Pitch, Roll)")
plt.xlabel("Time (seconds)")
plt.ylabel("Angle (degrees)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



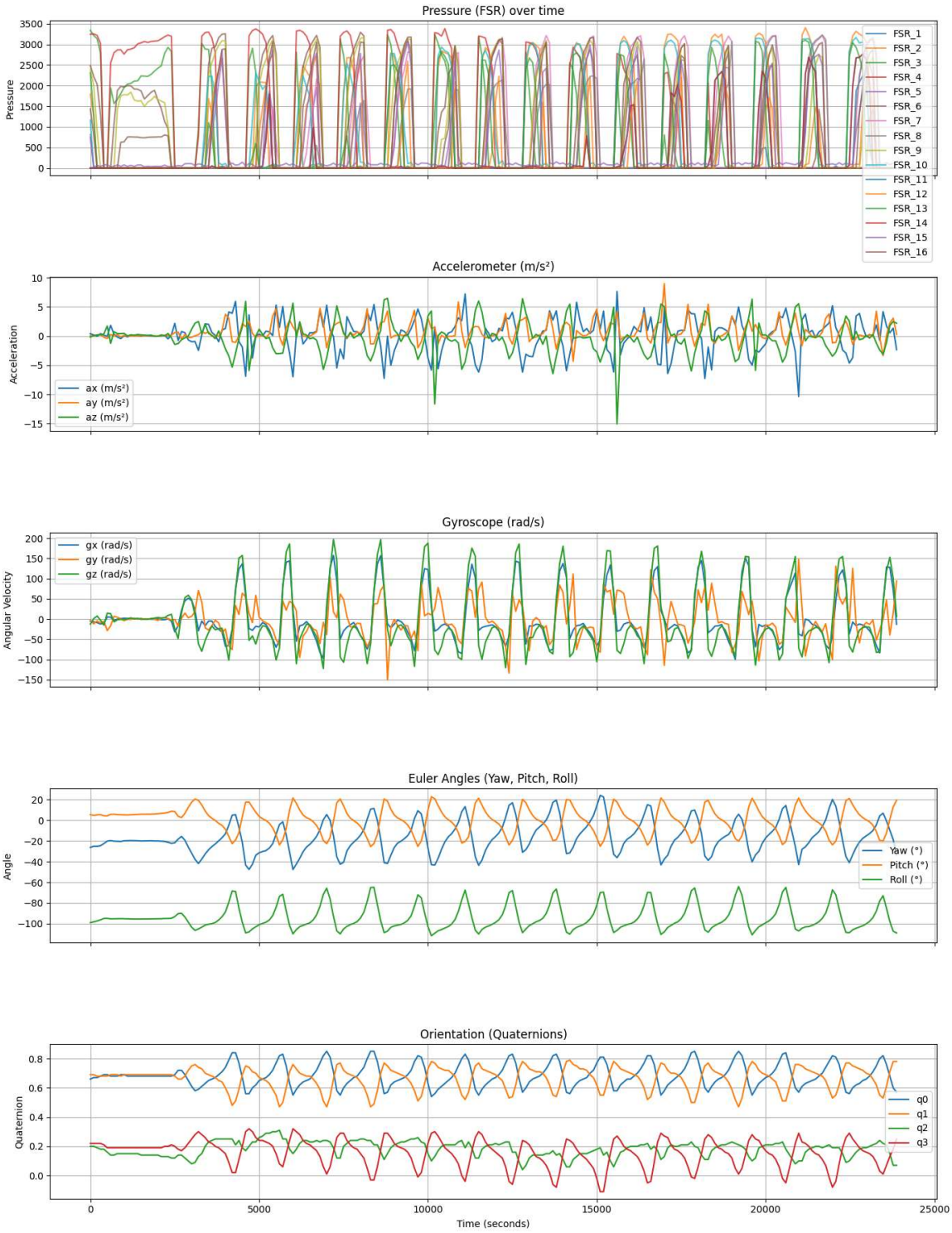
Raw Quternions Readings

In [60]:
```
plt.figure(figsize=(12, 5))

plt.plot(df["time_sec"], df["q0"], label="q0")
plt.plot(df["time_sec"], df["q1"], label="q1")
plt.plot(df["time_sec"], df["q2"], label="q2")
plt.plot(df["time_sec"], df["q3"], label="q3")

plt.title("Quaternion Components Over Time")
plt.xlabel("Time (seconds)")
plt.ylabel("Value")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [63]:

```
fig, axs = plt.subplots(5, 1, figsize=(14, 18), sharex=True)

# 1. Region Pressure
for i in range(1, 17):
    axs[0].plot(df["time_sec"], df[f"FSR_{i}"], label=f"FSR_{i}", alpha
axs[0].set_ylabel("Pressure")
axs[0].set_title("Pressure (FSR) over time")
axs[0].legend()
axs[0].grid(True)

# 2. Accelerometer (m/s²)
axs[1].plot(df["time_sec"], df["ax"], label="ax (m/s²)")
axs[1].plot(df["time_sec"], df["ay"], label="ay (m/s²)")
axs[1].plot(df["time_sec"], df["az"], label="az (m/s²)")
axs[1].set_ylabel("Acceleration")
axs[1].set_title("Accelerometer (m/s²)")
axs[1].legend()
axs[1].grid(True)

# 3. Gyroscope (rad/s)
axs[2].plot(df["time_sec"], df["gx"], label="gx (rad/s)")
axs[2].plot(df["time_sec"], df["gy"], label="gy (rad/s)")
axs[2].plot(df["time_sec"], df["gz"], label="gz (rad/s)")
axs[2].set_ylabel("Angular Velocity")
axs[2].set_title("Gyroscope (rad/s)")
axs[2].legend()
axs[2].grid(True)

# 4. Euler Angles (degrees)
axs[3].plot(df["time_sec"], df["yaw"], label="Yaw (°)")
axs[3].plot(df["time_sec"], df["pitch"], label="Pitch (°)")
axs[3].plot(df["time_sec"], df["roll"], label="Roll (°)")
axs[3].set_ylabel("Angle")
axs[3].set_title("Euler Angles (Yaw, Pitch, Roll)")
axs[3].legend()
axs[3].grid(True)

# 5. Quaternions
axs[4].plot(df["time_sec"], df["q0"], label="q0")
axs[4].plot(df["time_sec"], df["q1"], label="q1")
axs[4].plot(df["time_sec"], df["q2"], label="q2")
axs[4].plot(df["time_sec"], df["q3"], label="q3")
axs[4].set_xlabel("Time (seconds)")
axs[4].set_ylabel("Quaternion")
axs[4].set_title("Orientation (Quaternions)")
axs[4].legend()
axs[4].grid(True)

plt.tight_layout()
plt.show()
```


Pressure (FSR) over time
Pressure
FSR_1
FSR_2
FSR_3
FSR_4
FSR_5
FSR_6
FSR_7
FSR_8
FSR_9
FSR_10
FSR_11
FSR_12
FSR_13
FSR_14
FSR_15
FSR_16
Accelerometer (m/s²)
Acceleration
ax (m/s²)
ay (m/s²)
az (m/s²)
Gyroscope (rad/s)
Angular Velocity
gx (rad/s)
gy (rad/s)
gz (rad/s)
Euler Angles (Yaw, Pitch, Roll)
Angle
Yaw (°)
Pitch (°)
Roll (°)
Orientation (Quaternions)
Quaternion
q0
q1
q2
q3
Time (seconds)