Week: 1

	Entry	Description / Remarks
	Discussion with supervisor about research Methodology and timeline for PSM 2	

Supervisor Signature

_____

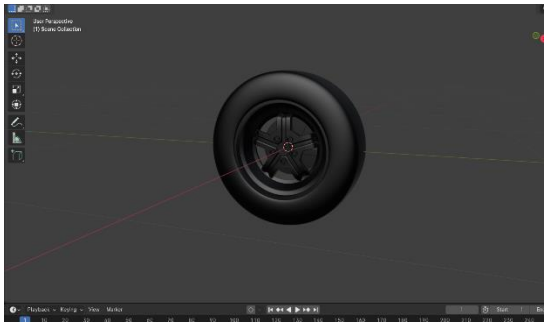
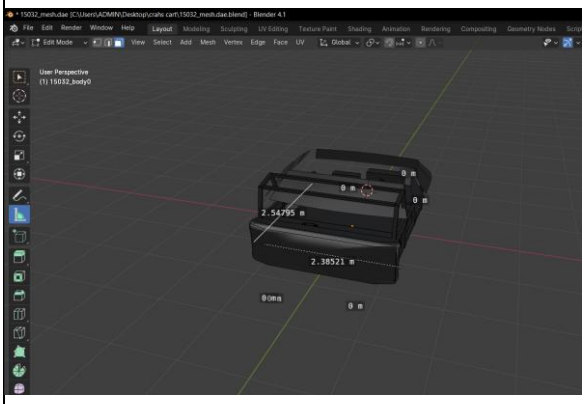
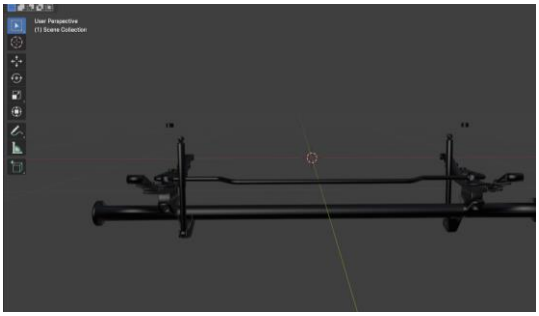
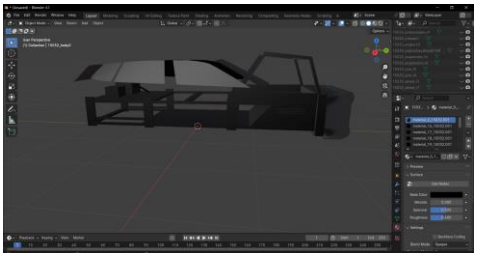
Week: 2

	Entry	Description / Remarks
	Creation of Trolley model in Blender application.	

Supervisor Signature

_____

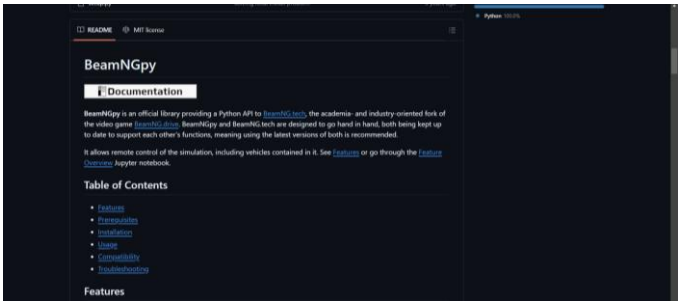
Week: 3

	Entry	Description / Remarks
	-Creation of Trolley model in blender	 Wheel  Body panels  Drive shaft  Full body structure

Supervisor Signature

_____

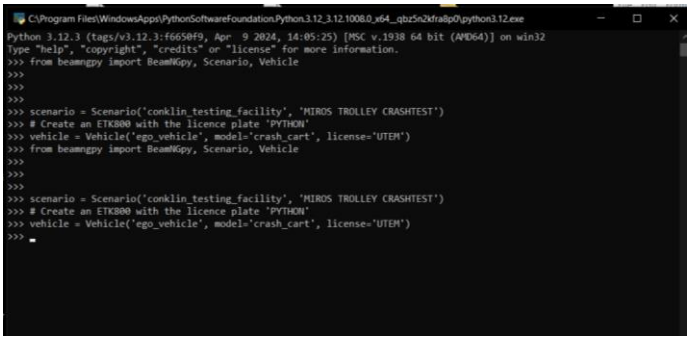
Week: 4

	Entry	Description / Remarks
	Initialization of BeamNGpy, Python interpreter and Notepad++	
	Learning of BeamNGpy codes from Github website	

Supervisor Signature

_____

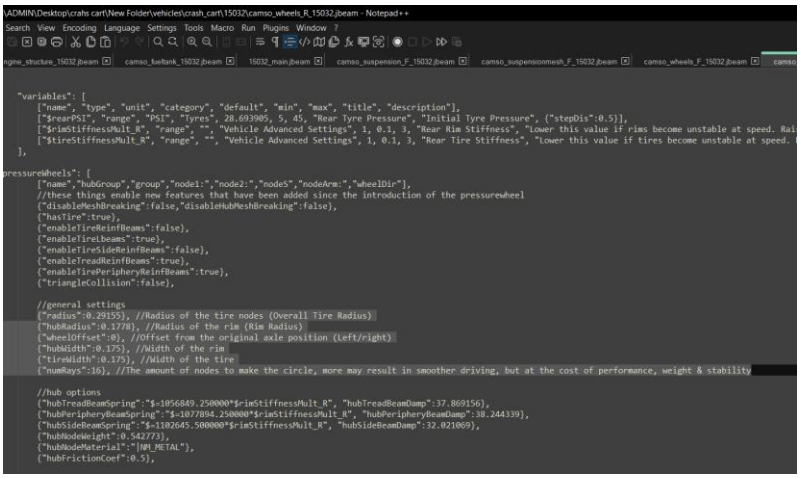
Week: 5

	Entry	Description / Remarks
	Connection of Beamng.tech application through python interpreter in localhost	 Python interpreter code to connect Beamng.Tech application through local host.

Supervisor Signature

_____

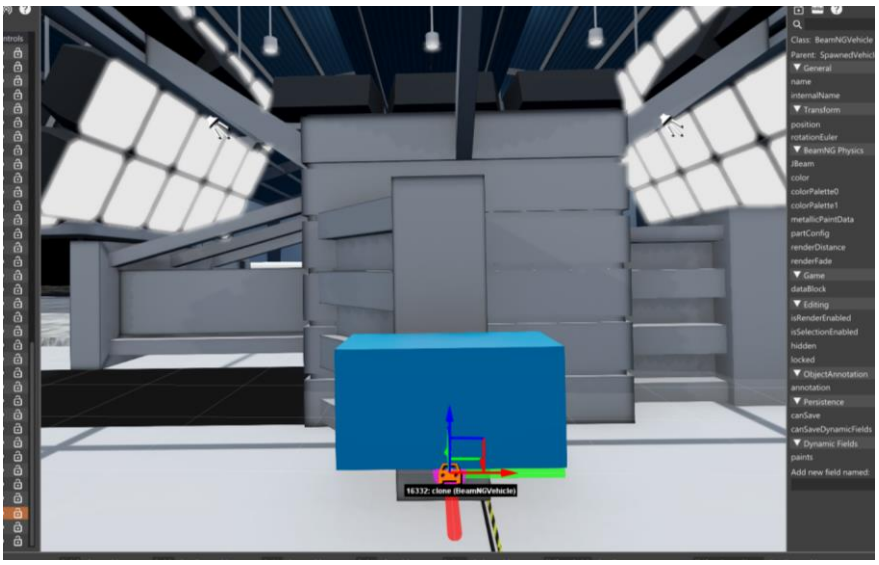
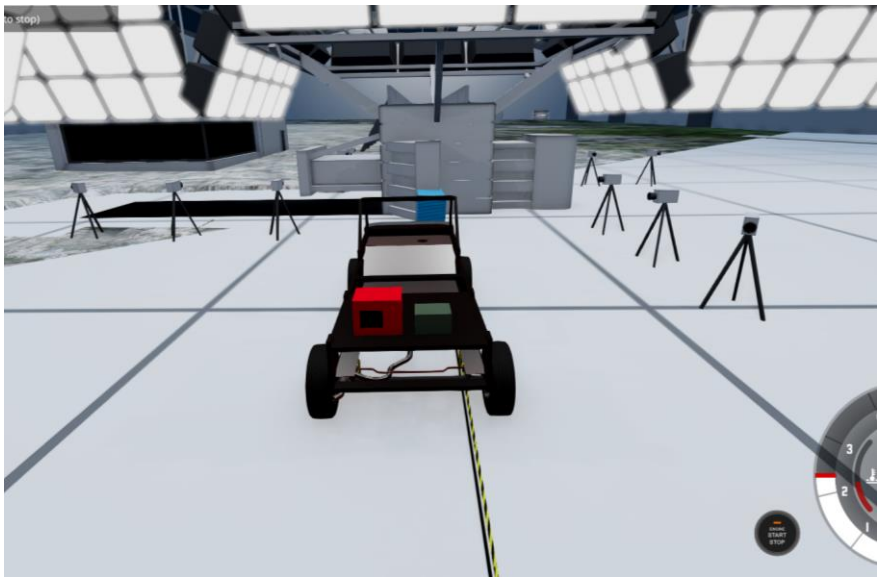
Week: 6

	Entry	Description / Remarks
	Editing of trolley model Spring and damper values using notepad++	 Values of the Spring stiffness and damping coefficient are saved in note format to edit easily. The values are counted as ‘nodes’ in Beamng.Tech software.

Supervisor Signature

_____

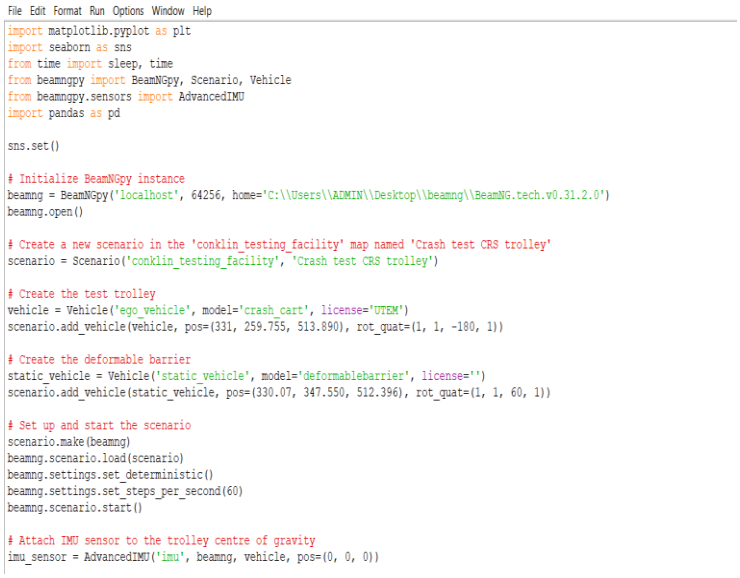
Week: 7

	Entry	Description / Remarks
	Creation of test scenario in Beamng.Tech application	 The scenario uses a map called “concklin_test_facility” built inside the software. A deformable barrier was placed in the middle of the impact point to increase the realism.  The trolley models and deformable barrier coordinates were calculated to ensure a linear movement is achieved and the deformable barrier experiences a head on collision.

Supervisor Signature

_____

Week: 8

	Entry	Description / Remarks
	Coding for scenario initialization and object placement in simulation	

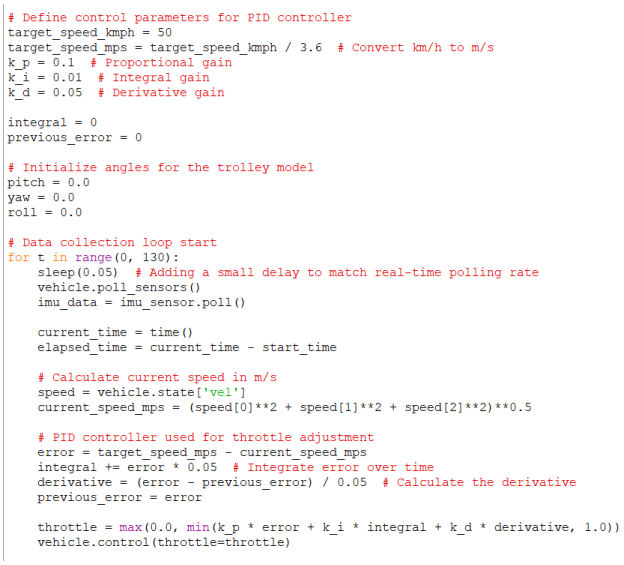
```
File  Edit  Format  Run  Options  Window  Help
import matplotlib.pyplot as plt
import seaborn as sns
from time import sleep, time
from beamngpy import BeamNGpy, Scenario, Vehicle
from beamngpy.sensors import AdvancedIMU
import pandas as pd

sns.set()

# Initialize BeamNGpy instance
beamng = BeamNGpy('localhost', 64256, home='C:\\Users\\ADMIN\\Desktop\\beamng\\BeamNG.tech.v0.31.2.0')
beamng.open()

# Create a new scenario in the 'conklin_testing_facility' map named 'Crash test CRS trolley'
scenario = Scenario('conklin_testing_facility', 'Crash test CRS trolley')

# Create the test trolley
vehicle = Vehicle('ego_vehicle', model='crash_cart', license='UTEM')
scenario.add_vehicle(vehicle, pos=(331, 259.755, 513.890), rot_quat=(1, 1, -180, 1))

# Create the deformable barrier
static_vehicle = Vehicle('static_vehicle', model='deformablebarrier', license='')
scenario.add_vehicle(static_vehicle, pos=(330.07, 347.550, 512.396), rot_quat=(1, 1, 60, 1))

# Set up and start the scenario
scenario.make(beamng)
beamng.scenario.load(scenario)
beamng.settings.set_deterministic()
beamng.settings.set_steps_per_second(60)
beamng.scenario.start()

# Attach IMU sensor to the trolley centre of gravity
imu_sensor = AdvancedIMU('imu', beamng, vehicle, pos=(0, 0, 0))
```

The above picture shows the code to initialize the scenario, create the deformable barrier and trolley model in the map with said coordinates, and start the application.

Supervisor Signature

_____

Week: 9

	Entry	Description / Remarks
	Coding speed control of trolley	

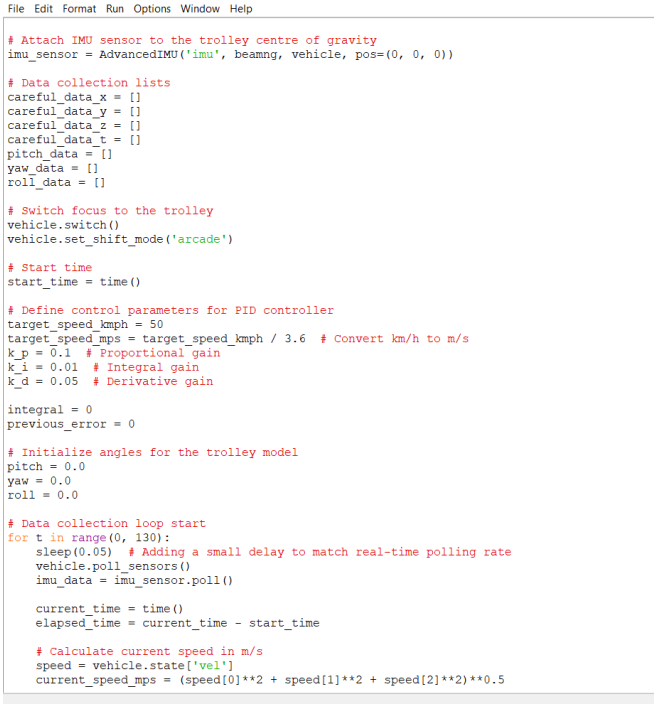
```
# Define control parameters for PID controller
target_speed_kmph = 50
target_speed_mps = target_speed_kmph / 3.6  # Convert km/h to m/s
k_p = 0.1  # Proportional gain
k_i = 0.01  # Integral gain
k_d = 0.05  # Derivative gain

integral = 0
previous_error = 0

# Initialize angles for the trolley model
pitch = 0.0
yaw = 0.0
roll = 0.0

# Data collection loop start
for t in range(0, 130):
    sleep(0.05)  # Adding a small delay to match real-time polling rate
    vehicle.poll_sensors()
    imu_data = imu_sensor.poll()

    current_time = time()
    elapsed_time = current_time - start_time

    # Calculate current speed in m/s
    speed = vehicle.state['vel']
    current_speed_mps = (speed[0]**2 + speed[1]**2 + speed[2]**2)**0.5

    # PID controller used for throttle adjustment
    error = target_speed_mps - current_speed_mps
    integral += error * 0.05  # Integrate error over time
    derivative = (error - previous_error) / 0.05  # Calculate the derivative
    previous_error = error

    throttle = max(0.0, min(k_p * error + k_i * integral + k_d * derivative, 1.0))
    vehicle.control(throttle=throttle)
```

The above picture shows the code to control the speed of the trolley model. The trolley must have a consistent speed of 50 km/h. A PID controller was used to control the speed as the normal code to control the speed caused the trolley to accelerate infinitely. The PID controller allowed the trolley to maintain a speed of 50 – 51 km/h throughout the simulation.

Supervisor Signature

_____

Week: 10

	Entry	Description / Remarks
	Coding for data collection of trolley model simulation	vehicle_logging5.py - C:\Users\ADMIN\Desktop\vehicle_logging5.py (3.12.3) File Edit Format Run Options Window Help  The above picture shows the code that initializes the data collection lists in the trolley model. An Inertial Measurement Unit (IMU) sensor was added at the centre of gravity of trolley model for the data collection. A for loop was used to create the data collection and a timeframe of 130 seconds were used with a polling rate of 0.05 seconds for each reading of the sensor.

```
# Attach IMU sensor to the trolley centre of gravity
imu_sensor = AdvancedIMU('imu', beamng, vehicle, pos=(0, 0, 0))

# Data collection lists
careful_data_x = []
careful_data_y = []
careful_data_z = []
careful_data_t = []
pitch_data = []
yaw_data = []
roll_data = []

# Switch focus to the trolley
vehicle.switch()
vehicle.set_shift_mode('arcade')

# Start time
start_time = time()

# Define control parameters for PID controller
target_speed_kmph = 50
target_speed_mps = target_speed_kmph / 3.6  # Convert km/h to m/s
k_p = 0.1  # Proportional gain
k_i = 0.01  # Integral gain
k_d = 0.05  # Derivative gain

integral = 0
previous_error = 0

# Initialize angles for the trolley model
pitch = 0.0
yaw = 0.0
roll = 0.0

# Data collection loop start
for t in range(0, 130):
    sleep(0.05)  # Adding a small delay to match real-time polling rate
    vehicle.poll_sensors()
    imu_data = imu_sensor.poll()

    current_time = time()
    elapsed_time = current_time - start_time

    # Calculate current speed in m/s
    speed = vehicle.state['vel']
    current_speed_mps = (speed[0]**2 + speed[1]**2 + speed[2]**2)**0.5
```

Supervisor Signature

_____

Week: 11

	Entry	Description / Remarks

	Coding for tabulation and graphing of collected data	vehicle_logging5.py - C:\Users\ADMIN\Desktop\vehicle_logging5.py (3.12.3)
		File Edit Format Run Options Window Help
		`vehicle.control(throttle=throttle)`
		`for i in range(0, len(imu_data)):`
		`careful_data_x.append(imu_data[i]['accSmooth'][0])  # The reading in the IMU's x-axis.`
		`careful_data_y.append(imu_data[i]['accSmooth'][1])  # The reading in the IMU's y-axis.`
		`careful_data_z.append(imu_data[i]['accSmooth'][2])  # The reading in the IMU's z-axis.`
		`careful_data_t.append(imu_data[i]['time'])  # The time stamp for the tri-axial reading.`
		`# Update angles using angular velocities reading`
		`roll += imu_data[i]['angVelSmooth'][0] * 0.05  # Roll from angular velocity around x-axis`
		`pitch += imu_data[i]['angVelSmooth'][1] * 0.05  # Pitch from angular velocity around y-axis`
		`yaw += imu_data[i]['angVelSmooth'][2] * 0.05  # Yaw from angular velocity around z-axis`
		`pitch_data.append(pitch)`
		`yaw_data.append(yaw)`
		`roll_data.append(roll)`
		`# Close the connection with the simulation`
		`beamng.close()`
		`# Create a DataFrame with the collected data`
		`data = {`
		`'Time (s)': careful_data_t,`
		`'Acceleration X (m/s^2)': careful_data_x,`
		`'Acceleration Y (m/s^2)': careful_data_y,`
		`'Acceleration Z (m/s^2)': careful_data_z,`
		`'Roll (radians)': roll_data,`
		`'Pitch (radians)': pitch_data,`
		`'Yaw (radians)': yaw_data`
		`}`
		`df = pd.DataFrame(data)`
		`# show the full DataFrame`
		`pd.set_option('display.max_rows', None)`
		`pd.set_option('display.max_columns', None)`
		`pd.set_option('display.width', None)`
		`pd.set_option('display.max_colwidth', None)`
		`# Display the DataFrame`
		`print(df)`
		The data was collected using a for loop. ‘accSmooth’ variable was used to ensure the data that is being received has unwanted noise and data removed from the sensor. The data collected is then saved in CSV format to be used in excel. The ‘print(df)’ variable then opens an excel file to log all the data collected.
		vehicle_logging5.py - C:\Users\ADMIN\Desktop\vehicle_logging5.py (3.12.3)
		File Edit Format Run Options Window Help
		`'Roll (radians)': roll_data,`
		`'Pitch (radians)': pitch_data,`
		`'Yaw (radians)': yaw_data`
		`}`
		`df = pd.DataFrame(data)`
		`# show the full DataFrame`
		`pd.set_option('display.max_rows', None)`
		`pd.set_option('display.max_columns', None)`
		`pd.set_option('display.width', None)`
		`pd.set_option('display.max_colwidth', None)`
		`# Display the DataFrame`
		`print(df)`
		`# Save the DataFrame to a CSV file`
		`df.to_csv('collected_data.csv', index=False)`
		`# Plot the data collected`
		`figure, ax = plt.subplots(2, 3, figsize=(15, 10), sharey=False)`
		`ax[0, 0].plot(df['Time (s)'], df['Acceleration X (m/s^2)'], 'b-')`
		`ax[0, 0].set_xlabel('t (s)')`
		`ax[0, 0].set_ylabel('accel (m/s^2)')`
		`ax[0, 0].set_title('Acceleration in X-axis')`
		`ax[0, 1].plot(df['Time (s)'], df['Acceleration Y (m/s^2)'], 'b-')`
		`ax[0, 1].set_xlabel('t (s)')`
		`ax[0, 1].set_ylabel('accel (m/s^2)')`
		`ax[0, 1].set_title('Acceleration in Y-axis')`
		`ax[0, 2].plot(df['Time (s)'], df['Acceleration Z (m/s^2)'], 'b-')`
		`ax[0, 2].set_xlabel('t (s)')`
		`ax[0, 2].set_ylabel('accel (m/s^2)')`
		`ax[0, 2].set_title('Acceleration in Z-axis')`
		`ax[1, 0].plot(df['Time (s)'], df['Roll (radians)'], 'g-')`
		`ax[1, 0].set_xlabel('t (s)')`
		`ax[1, 0].set_ylabel('Roll (radians)')`
		`ax[1, 0].set_title('Roll over Time')`
		`ax[1, 1].plot(df['Time (s)'], df['Pitch (radians)'], 'g-')`
		`ax[1, 1].set_xlabel('t (s)')`
		`ax[1, 1].set_ylabel('Pitch (radians)')`
		`ax[1, 1].set_title('Pitch over Time')`
		`ax[1, 2].plot(df['Time (s)'], df['Yaw (radians)'], 'g-')`
		`ax[1, 2].set_xlabel('t (s)')`
		`ax[1, 2].set_ylabel('Yaw (radians)')`
		`ax[1, 2].set_title('Yaw over Time')`
		`plt.tight_layout()`
		`plt.show()`
		The data is then used to plot 6 Figures that display the y-axis, x-axis, z-axis, pitch, yaw, and roll values of the trolley model

Week: 12

	Entry	Description / Remarks
	Results and Discussion writing	

Supervisor Signature

_____

Week: 13

	Entry	Description / Remarks
	Final report writing	

Supervisor Signature

_____

Week: 14

	Entry	Description / Remarks
	-PSM banner creation -Final presentation -Final report submission	

Supervisor Signature

_____