ELEC330 Assignment 1

Butterfly Robot Design

By: Alvaro Mesa Giner (201656665), Gideon Tladi (201426759), Junyang Xiao (201769708), Yanzhang Wang (201608925), Yifan Wang (201677153), Zijin Wu (201769693)

Department of Electrical Engineering and Electronics

20 October 2024

Abstract

This paper abridges the chief aspects of the design of a butterfly robot named Nectar Wings. Firstly, this paper introduces the key components of a simple robotic system, followed by the specification requirements for this project. Next, taking typical butterfly locomotion into account, the robot design and purpose are described. Finally, an overall analysis is given, highlighting future improvements and research developments that could be made based on this project idea. This project serves as both an educational tool and a prototype design for an aerial robot.

Declaration

We confirm that we have read and understood the University's definitions of plagiarism and collusion from the Code of Practice on Assessment. We confirm that we have neither committed plagiarism in the completion of this work nor have we colluded with any other party in the preparation and production of this work. The work presented here is our own and in our own words except where we have clearly indicated and acknowledged that we have quoted or used figures from published or unpublished sources (including the web). We understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment (Appendix L).

Contents

List of Figures

1 Introduction

1.1 Background Information

Robots are electronic, mobile machines that can execute tasks autonomously, semi-autonomously, or when manually operated by humans. All robots need a body or frame, actuators, sensors, a power supply, and a control system [1] to be classified as such. Furthermore, safety features must be implemented in the robot design to ensure the robot will not cause harm to living beings or the environment in which it will operate. Additionally, some robots have the ability to communicate with other devices and operate unaided.

Biomimetic robots are a special type of robots designed to imitate real-life biological beings like humans or animals. In the design of biomimetic robots, the behaviour and movement structures often draw inspiration from biological systems found in nature. Research in this field merges biology, mechanical engineering, and materials science with the goal of observing and studying the motion of organisms to develop more efficient ways to tackle environmental challenges [2]. For instance, in 1903, the Wright brothers invented the Wright Flyer based on the aerodynamic principles of bird flight. Today, the applications of biomimetic robots span various fields, such as medical surgery robots that mimic human arms and hands for precise operations. As advancements in materials science and sensor technology continue, more agile and lightweight biomimetic robots have emerged, allowing for rapid development in this innovative field.

This project focuses on designing a biomimetic robot inspired by the butterfly. Based on the flight dynamics of butterflies, the robot mimics the wing flapping mechanisms that generate thrust and lift through the clap-and-fling motion [3]. This enables the robot to move with agility in confined spaces. Such efficient, small-scale motion behaviours make it particularly well-suited for applications in environmental monitoring and data collection.

1.2 Objectives

The overall aim is to develop an autonomous, biomimetic butterfly robot capable of performing take-off, flight, and landing using a single sensor (a camera) for navigation and environmental sensing. The robot will be designed and simulated in Gazebo, with hardware assembly and real-world testing following successful virtual simulations, ensuring a smooth transition from simulation to physical implementation within about 24 weeks.

The SMART project objectives are:

Specific:

- Develop a biomimetic butterfly robot capable of autonomously performing take-off, flight, and landing. The robot will be equipped with only one sensor (either a camera or LIDAR) for navigation and environmental sensing.

Measurable:

- The robot should be able to take off, fly for 10 seconds, and land autonomously.
- The accuracy of performing these actions should be above 90%, ensuring no manual intervention is required.

Achievable:

- The project team will first carry out detailed simulations using Gazebo, testing the flight dynamics, take-off, and landing capabilities of the robot in a virtual environment. After successful simulations, hardware assembly will follow, and real-world testing will be conducted to ensure the simulation results can be replicated in physical conditions. Available resources and tools will support the transition from simulation to hardware implementation.

Relevant:

- This project aligns with the current trends in autonomous robotics, particularly in the context of environmental monitoring and research in natural habitats. Using LIDAR or a camera for data collection and navigation addresses key challenges in environmental conservation and robotic applications.

Time-bound:

- During the first semester (12 weeks), complete the initial design and simulation.
- During the second semester (12 weeks), complete the hardware assembly, testing, and integration of the robot's data analysis systems.

2 Specification Requirements

2.1 Appearance and Size

In terms of appearance and size, the design specifications of the project's bionic butterfly robot include a fuselage length of 15 cm, a wingspan of 30 cm, and a weight of 80 grams. In order to ensure the robot's lightness and stability in flight, the fuselage material selected 3D printed high-strength plastic and carbon fiber brackets. The wings are made of P31N fabric, which has good durability and flexibility and is suitable for the dynamic flapping required for bionic design.

Several critical factors were considered in determining the robot’s size and design. First, wing loading, which is the ratio of body weight to wing area, was a key consideration. A higher wing loading would result in the robot needing to consume more energy to maintain flight, which would reduce the overall flight duration [3]. Second, the wing-to-body mass ratio significantly impacts flight stability. If this ratio is too high, the robot could experience excessive vertical and horizontal oscillations during flight, severely affecting stability [4]. Finally, the aspect ratio (the ratio of wingspan to wing chord length) was another essential factor. The robot's wide wings result in a low aspect ratio, which mimics the natural slow flight of butterflies. Although a low aspect ratio can decrease flight speed [5], it also reduces energy consumption, making the robot more efficient. By incorporating advanced control algorithms, the robot's flight agility is enhanced, compensating for the slower flight speed due to the low aspect ratio.

This design ensures that the robot balances energy efficiency, and stability, making it well-suited for its intended testing environment.

2.2 The Sensor

The robot is equipped with a 1080p resolution camera for environment awareness and navigation. The purpose of having a single sensor is to make the robot lighter and reduce the power consumption during take-off. In addition, the design of a single sensor not only reduces hardware complexity, but also effectively reduces power consumption, making the butterfly robot more flexible and stable during operation. The camera is installed on the head of the robot butterfly to ensure that it can collect clear image data in real time during flight to meet the needs of navigation. In future in-depth studies, high-definition cameras can also provide environmental monitoring functions and increase the robustness of the butterfly robot.

2.3 The Flight Performance

In the aspect of flight performance, the robot has the ability to complete automatic take-off, hovering, flight for 10 seconds and automatic landing. Its take-off time is controlled within 5 seconds and the flight altitude is not more than 3 meters. The robot will be tested indoors in a windless environment. The robot has three basic flight modes of autonomous take-off, hovering and landing, which can ensure the stability and flexibility of the flight process.

2.4 Dynamics System

The design of power system is very important for the stability of robot flight. This project selected two GDW DS1906B steering engines as the core drive. They provide the butterfly robot with a wing-beating frequency of 5 Hertz per second. This frequency is close to the butterfly's natural flight pattern, helping to simulate its efficient aerodynamics. In addition, the robot is equipped with a battery with a capacity of 200mAh, which can support its continuous flight for 10 minutes. Such a battery design not only ensures the smooth completion of the flight mission, but also meets the needs of long-term data acquisition missions in the future. This greatly improves the practical application value of the robot.

In the design of the power system, the decision to use a steering gear instead of an ordinary motor is driven by the specific mechanical requirements of the bionic butterfly robot’s wing movements. The wings of a butterfly generate both thrust and lift through an up-and-down flapping motion, while an ordinary motor is limited to rotational movements, which do not align with the natural flapping and "clap-and-fling" dynamics of a butterfly's wings. To use an ordinary motor, a mechanical linkage would be needed to convert rotational motion into flapping. In this case, the complexity of the structure will increase, and the maintenance challenges will occur. These challenges become more pronounced under high-frequency movements, where the durability and reliability of linkage systems may degrade more quickly.

Moreover, another limitation of using a single motor and linkage system is the difficulty in controlling the speed differential between the two wings during turning maneuvers. In nature, butterflies adjust their turning by modulating the frequency and angle of each wing independently. However, a single mechanical linkage system cannot effectively provide such differential control, which restricts the flexibility and agility of the robot’s flight dynamics. In contrast, using two separate servos (steering gears) to control the left and right wings allows for precise control over the wing's angle and flapping speed. This not only enables differential control, facilitating turning and manoeuvrability, but also simplifies the design

while improving system stability and ease of maintenance. As a result, the choice of steering gear over an ordinary motor is optimal for meeting the requirements of a bionic design.

2.5 Control System

The control system is the core of the whole robot design. Through adaptive control algorithm, the robot can take off, hover and land safely. The real-time camera data will constantly update the control algorithm, enabling the robot to adjust itself to changes in the surrounding environment to ensure smooth and accurate flight. In addition, the control system supports autonomous path planning, enabling obstacle avoidance and flight decisions in complex environments.

2.6 Environmental adaptability

The design of the butterfly robot does not include wind resistance. Therefore, all tests will be conducted in a windless indoor environment. The purpose of this project is to study the flight behaviour pattern of butterfly and design a bionic robot by imitating its physiological structure. The basic flight mode of the butterfly will be referred to, while functions such as wind resistance are not considered in the current design. In addition, the operating temperature range of the robot is 0°C to 40°C, and the humidity adaptation range is 30 to 60 percent. These design considerations allow it to operate safely in most indoor environments. The design ensures the stability of the robot in a variety of environmental conditions, while providing operational space for future expansion and upgrades.

2.7 Data acquisition and processing

Data acquisition and processing is one of the key functions of bionic butterfly robot to realize environmental monitoring. The processing speed of the robot camera is 3 frames per second, which can meet the basic real-time environmental data acquisition needs. In addition, the equipped internal storage function enables the robot to store camera data without an external connection. The wireless data transmission function further expands the robot's use scenario, allowing real-time monitoring and remote data processing. This design takes into account not only the efficiency of data processing, but also the practicality of the robot.

2.8 Safety Design

In order to avoid accidental failure resulting in equipment damage or injury to the surrounding environment and personnel, the robot has been designed with an emergency shutdown mechanism that can quickly respond and stop operation when there is a problem. Such a safe design ensures the safety of the robot and provides an effective guarantee for risk control in practical applications. This capability is particularly important for the stable operation of the robot in complex environments, especially when it is tested and applied in more complex external environments in the future.

3 Robot Design and Model

Nectar Wings is a robotic butterfly designed to mimic the natural flight of a butterfly, with a focus on lightweight construction and agile movement. The objective is to develop an autonomous biomimetic robot capable of take-off, hovering and landing with adaptive flight control. The design process is covered in this section including design objectives, key components, material choices, simulation plans, and the software used.

3.1 Design Objectives

The main goal is to build an aerial robot that mimics butterfly flight mechanics, focusing on stability and low power consumption. A camera will be used for real time environmental awareness to achieve adaptive path planning and navigation in confined indoor spaces.

3.2 Key Components

Wings and Holders: The wings are constructed from P31N fabric for flexibility and durability, and wing holders are used to provide structural support to maximise lift during the clap and fling wing motion. The model of wings is shown in Fig.3.1 below.

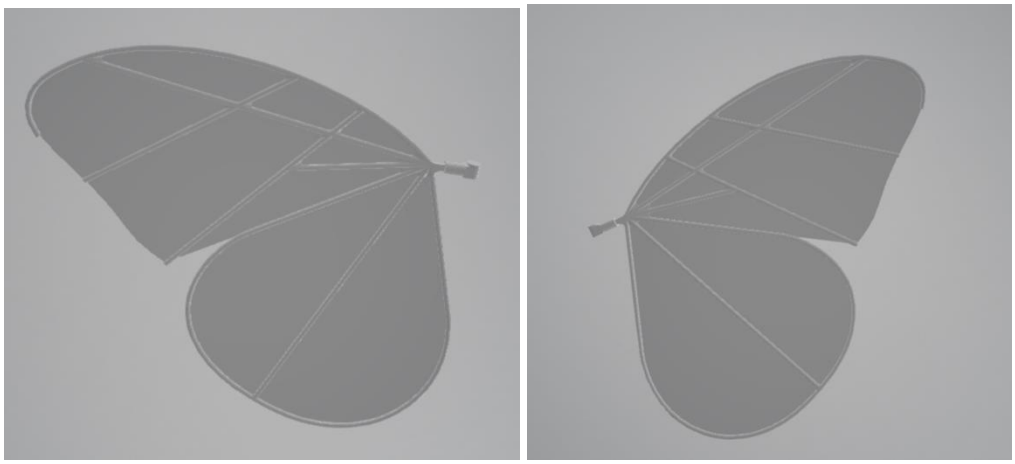

Fig.3.1 The Wings of the Butterfly Robot.

Body and Frame: The frame is constructed from carbon fibre and high strength plastic and integrates the core components while keeping weight to a minimum. It provides structural rigidity necessary for stable flight. The model of the body is shown in Fig.3.2 below.

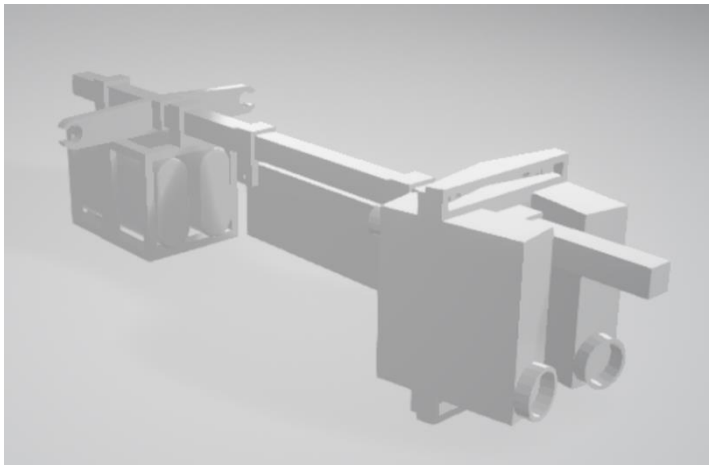

Fig.3.2 The Base of the Butterfly Robot

Motors and Motor Holders: The flapping mechanism is driven by two GDW DS1906B motors at a frequency of 5 Hz. They are mounted securely in motor holders, with precise alignment and adequate range of motion.

Battery and Battery Holder: The butterfly is powered by two 200mAh batteries in a custom holder, which give it up to 10 minutes of flight and balanced weight distribution.

Control Unit and Holder: Mounted in a holder that reduces added weight and secures the component, the control unit processes signals and executes flight commands.

Camera: A 1080p camera at the head of the robot provides real time visual data for navigation and adaptive control, used for path planning and obstacle avoidance.

3.3 Material Selection

The frame and wing supports are chosen to be made of materials such as carbon fibre and high strength plastic to achieve a target weight of about 100 grammes. The wings are made from P31N fabric, which offers the necessary balance of flexibility and strength for dynamic wing movements.

3.4. Simulation, Testing, and Software

The robot was 3D modelled using SolidWorks and Fusion 360 to achieve a precise design and visualisation of each component. The first semester will be spent conducting simulations in Gazebo to optimise the flight dynamics, including take off, hover, and landing performance. Weight distribution, motor positioning and wing configurations will be adjusted based on simulation feedback. In the second semester, we will learn about hardware assembly and real-world testing.

3.5 Challenges and Adjustments

A major design challenge was to ensure a balanced weight distribution while maintaining structural integrity. Optimal placement of the two motors and batteries was found to achieve stability. These challenges were addressed by lightweight materials like carbon fibre which improved manoeuvrability.

4 Model Simulation in Gazebo

In this section, the module described above will be divided into three assembly parts including Basebody, Leftwing and Rightwing. This distinction facilitates subsequent urdf file generation, and the main components of the model can be more intuitively understood. The whole model of the butterfly robot is shown in Fig.4.1. More specifically, in order to precisely configure these parts, a SolidWorks plug-in tool named sw2urdf is used to configure the joints, links, coordinates, and key points of the model. By defining these key parameters, the plug-in generates an URDF (Unified Robot Description Format) file that describes the

physical structure and motion capabilities of the robot. As for the below subsections, we will describe two main parts including module part and URDF file design and launch file design.

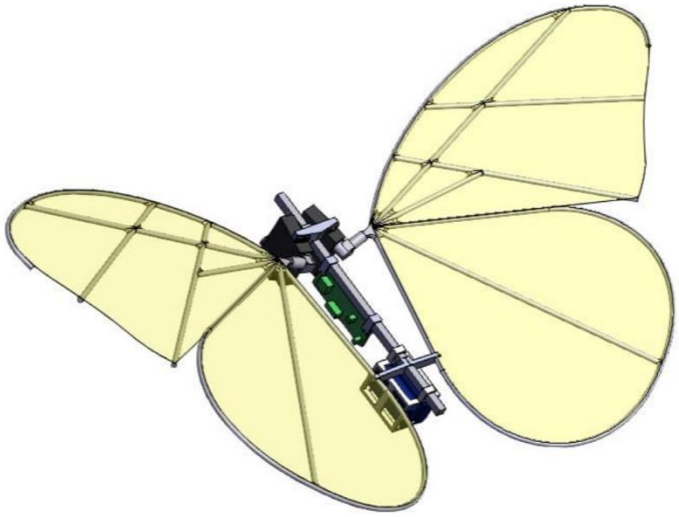

Fig.4.1 The Model of the Butterfly Robot

4.1 Module Part

4.1.1 Basebody part

This is the main structural part that contains the core systems as shown in Fig. 3.2 including two steering engines, steering gear stand, four bolts, one mobile jib, one development board and one battery holder. It acts as a connecting centre for the rest of the butterfly. The joint setting of the central torso is crucial for controlling the overall direction and movement of the wings relative to the torso.

4.1.2 Leftwing and Rightwing parts

As shown in Fig. 3.3, the wings are attached to the central torso and require precise joint Settings to simulate natural flapping and rotating motion. The joint of the wing must be flexible and capable of up and down flapping and rotating motion to ensure synchronous and controlled wing movement. Proper linkage Settings are also required to define the range and limits of these movements.

4.2 URDF File Design

We designed three coordinate systems on the main part of the robot butterfly and the joints where the two wings connect with the main body. These coordinate systems are used to accurately locate the movement of the wings. In addition, we also designed two reference axes at the inlay between the two steering engines and the wings as shown in Fig. 4.2, which serve as reference points for connection and rotation, ensuring that the wings can have precise Angle control during movement. The design of these coordinate systems and reference axes helps to define the relative position relationship between the wings and the main body and the motion characteristics.

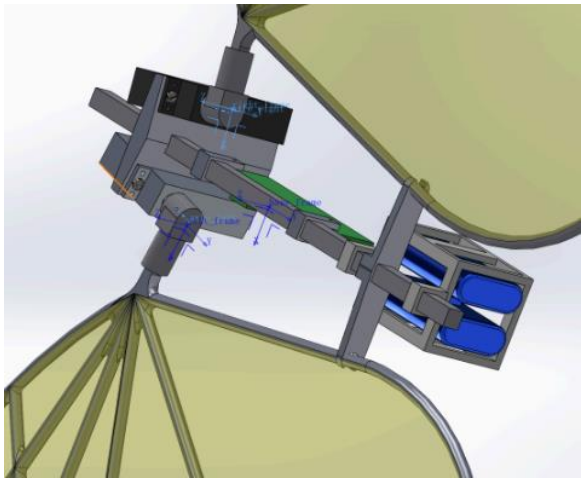

Fig.4.2: Design diagram of coordinate system and reference axis of the joint.

With these designs in place, we can use the SolidWorks plug-in sw2urdf to select the corresponding link and joint. This plugin will generate the URDF file based on the coordinate system and the reference axis we set, ensuring that the individual links and joints of the model have the correct physical properties and motion capabilities in the robot operating system (ROS). This process not only makes the structure of the model more accurate, but also provides a stable basis for subsequent simulation and control.

Moreover, in order to mimic the vibration effect of butterfly wings, we set an Angle limit on the reference axis, with a range of plus or minus 0.698132 radians. This limitation ensures that the wings can simulate the natural vibration amplitude during movement and avoid exceeding the reasonable rotation range, thus maintaining the stability and authenticity of the simulation process. After the setup, we exported the URDF file through the SolidWorks

plugin and used it for verification of the subsequent simulation platform. On these simulation sites such as Garrett Johnson, we can import URDF files to check the motion of the current model as shown in Fig.4.3, the vibration characteristics and the overall structure of the rationality, to ensure that the individual parts are set up as expected.

Fig.4.3: The wing movement simulation of robot butterfly by Garrett Johnson simulation platform

4.3 Launch File Design

However, the tool above section introduced can just output URDF suitable for ROS system instead of ROS2, but our Linux PC uses ROS2 Jazzy and Gazebo Harmonic. Thus, we need to use another tool named “sw2urdf_ros2” to help us transfer the ROS URDF file into ROS2. But there will be another question, this tool can just output launch file used in the RVIZ. Therefore, we need to write the launch file used in Gazebo by ourselves. The next section will discuss the details of this launch file following the flow chart shown in Fig. 4.4.

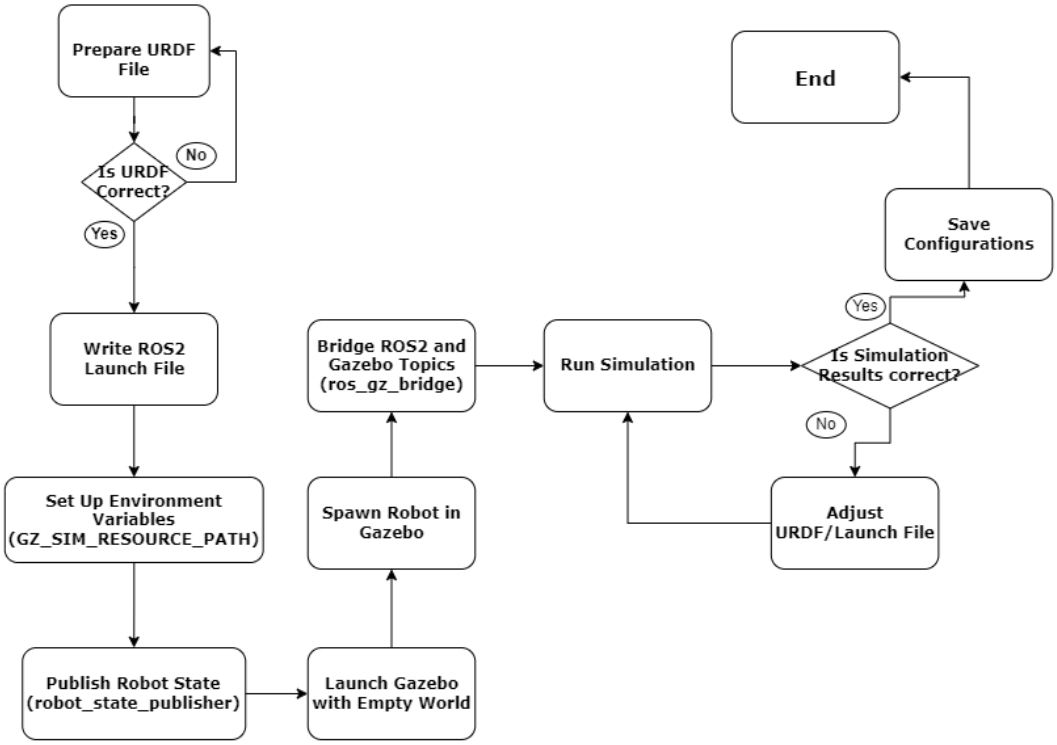

Fig.4.4: Gazebo simulation flow chart

- **Environment variable Settings**

  Firstly, we set the environment variable “GZ_SIM_RESOURCE_PATH” to indicate the path that Gazebo should take to find the model resource. This step ensures Gazebo is able to correctly locate the URDF file and its associated resources when loading the robot model.

- **Specify the URDF file path**

  In the startup file, we specify the urdf file path for the butterfly model (for example, 'butterfly.urdf') and store it as a variable for use by subsequent nodes. This document defines the physical structure and motion capabilities of the robot and is the core document of the simulation.

- **Declaration of startup parameters**

  We declare a startup parameter named “use_sim_time” that determines whether to enable emulation time. By default, the parameter is false, which means that the simulation time is not used. This allows you to flexibly adjust the simulation environment as needed.

- **Start the robot status publishing node**

  Use the “robot_state_publisher” node to load the URDF file into the ROS2 network. This node is responsible for publishing information about the robot's status (such as joint status and coordinate transformations), ensuring that other nodes can access and use this information to control and monitor the robot.

- **Gazebo simulation environment startup**

  Contains a Gazebo startup file that starts the simulation environment and loads an empty scene (empty.sdf). Here, we set up detailed logging so that we can get a clear picture of how the simulation environment is running during debugging.

- **Robot entity generation**

  We used the create executable from the “ros_gz_sim” package to import the robot model into the Gazebo simulation environment. The name of the robot (such as "robot1") and the initial position (coordinates x, y, z) are set here to ensure that the robot loads correctly and is in the desired position.

- **ROS2 and Gazebo's topic bridge:**

  Using the “ros_gz_bridge” node, we bridge data and commands between ROS2 and Gazebo. For example, I bridge the “/joint_states” topic to transfer the joint state and the “/cmd_vel” topic to pass the speed command. This design ensures that the control commands sent in ROS2 are properly acted upon by the robots in Gazebo, while the sensor data in Gazebo is fed back to ROS2 in real time.

- **Combine and return the full boot description**

  Finally, we put all the settings, parameters, and nodes together into a complete startup description. The purpose of this is to ensure that all nodes and the simulation environment load and interact smoothly at startup, thus achieving the correct operation and control of the robot model on the simulation platform.

5 Discussion and Reflections

5.1 Discussion

The design and parameter configuration of the biomimetic butterfly robot "Nectar Wings" in Gazebo represent significant progress in the autonomous flying robot project. By closely mimicking the flight dynamics of real butterflies, this project utilizes biomimetic principles to enhance its flight performance, making it particularly suitable for educational development tools and simple survey flight needs. The selection of lightweight materials such as carbon fiber and high-strength plastics, along with the biomimetic wing design, enhances the robot's agility and energy efficiency, which is crucial for sustaining longer flight durations.

A key challenge during the design process was ensuring a balanced weight distribution while maintaining structural integrity. Based on the research findings from UST Butterfly-II [6], the project decided to introduce two servos to independently drive the left and right wings, achieving tailless control of the robotic butterfly. This allows for differential control of wing motion, simulating the natural butterfly flight [6]. This is a crucial improvement, as traditional motor setups with linkages often compromise manoeuvrability, which not only adds mass but also significantly increases design complexity. Biomimetic butterfly robots typically emulate the natural butterfly wing structure with innovative designs like split wings to optimize lift and reduce weight, reflecting a delicate balance between structural integrity and functional performance. However, the overall structural design's reliability, imitating the lightweight and fragile structure of real butterfly wings, may also impact the robot's durability and reliability under different environmental conditions. Therefore, high-strength yet lightweight carbon fibre components were selected as the core elements of the robot in this project. The adaptive control algorithm implemented in the robot's flight system further enhances its ability to navigate in dynamic environments, showcasing potential applications in complex scenarios.

Moreover, the decision to use a single miniature camera or other distance sensors for navigation and obstacle avoidance simplifies the design and reduces power consumption, but it also limits sensor redundancy. Future iterations of the robot could explore integrating more sensors to enhance environmental awareness and improve navigation and obstacle avoidance accuracy under different conditions. Safety mechanisms, including remote control operations for testing flight attitudes and emergency shutdown features, highlight the project's commitment to operational safety, which is crucial in both experimental and practical applications.

5.2 Conclusions

The Nectar Wings project has successfully achieved the main objectives of this phase, namely, developing a structural model of an autonomous biomimetic butterfly robot capable of taking off, flying, and landing. Additionally, the configuration of model parameters and the loading of simulations into the Gazebo environment have been completed. This sets a solid foundation for further simulation development and subsequent physical research and improvements.

The plan not only aims to highlight the potential for robots to operate autonomously in real-world environments but also emphasizes the importance of lightweight design and agile movement. The use of advanced materials and careful consideration of weight distribution are crucial for ensuring the performance and stability of the robot during flight. Ultimately, by mimicking the efficient flapping of butterfly wings, the project gains valuable experience that will inform the development of physical prototypes.

Furthermore, the meticulous approach adopted in the subsequent design process, including extensive simulations in Gazebo, has laid a solid foundation for the hardware assembly and testing phases. The orderly transition from virtual models to physical implementations is crucial for minimizing risks and enhancing the reliability of the robot in real-world scenarios. The collaborative efforts of the project team have also fostered a rich environment of creativity and problem-solving, contributing to the overall success of the project.

In summary, the Nectar Wings project exemplifies the convergence of robotics technology and biology, demonstrating the power of nature-inspired design in creating effective and autonomous systems. As the team continues to refine and test the robot, these advancements will not only contribute to the field of robotics but also offer promising avenues for future applications in environmental monitoring and other areas.

References

[1] Tooling Ideas, “What Are The Key Components And Features Of A Robot,” 25 01 2024. [Online]. Available: https://toolingideas.com/what-are-the-key-components-and-features-of-a-robot/. [Accessed 20 10 2024].

[2] H. Huang, W. He, Z. Chen, T. Niu, and Q. Fu, “Development and experimental characterization of a robotic butterfly with a mass shifter mechanism,” *Biomimetic Intelligence and Robotics*, vol. 2, p. 100076, Oct. 2022, doi: 10.1016/j.birob.2022.100076.

[3] K. Suzuki, M. Nakamura, M. Kouji, and M. Yoshino, "Revisiting the flight dynamics of take-off of a butterfly: experiments and CFD simulations for a cabbage white butterfly," *Biology Open*, vol. 11, no. 3, Mar. 2022, doi: 10.1242/bio.059136.

[4] R. Xu, X. Zhang, and H. Liu, "Effects of wing-to-body mass ratio on insect flapping flights," *Physics of Fluids*, vol. 33, no. 2, p. 021902, Feb. 2021, doi: 10.1063/5.0034806.

[5] K. Sukvichai and K. Yajai, "Design of a Flapping Wings Butterfly Robot based on Aerodynamics Force," *Adv. Sci. Technol. Eng. Syst. J.*, vol. 5, no. 4, pp. 667-675, 2020, doi: 10.25046/aj050480.

[6] H. Huang, W. He, Y. Zou, and Q. Fu, “USTButterfly: a Servo-Driven Biomimetic Robotic Butterfly,” IEEE Transactions on Industrial Electronics, vol. 71, no. 2, pp. 1758–1767, Mar. 2023, doi: 10.1109/tie.2023.3260355.

Appendix

Group Member Contributions

- Alvaro – Modeling work of motor, control systems.etc.
- Gideon – Writing the executive summary, organising all the notes, resources and records in the group’s OneDrive folder.
- Junyang – Writing assignments report and Assisting Yifan in modeling.
- Yanzhang - The setup of ROS2 and simulation in Gazebo.
- Yifan - Modeling work of wings, wings holders and motor holders.
- Zijin - Writing assignments report and Assisting Yanzhang in simulation set.

Code and Programmes

Butterfly.urdf

<?xml version="1.0" encoding="utf-8"?>

<!-- This URDF was automatically created by SolidWorks to URDF Exporter! Originally created by Stephen Brawner (brawner@gmail.com)

```
    Commit Version: 1.6.0-4-g7f85cfe  Build Version: 1.6.7995.38578
    For more information, please see http://wiki.ros.org/sw_urdf_exporter -->
<robot
  name="butterfly">
<link name="base_footprint">
   <visual>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
         <box size="0.001 0.001 0.001" />
      </geometry>
   </visual>
</link>

<joint name="base_footprint_joint" type="fixed">
   <origin xyz="0 0 0.001" rpy="0 0 0" />
   <parent link="base_footprint"/>
   <child link="base_link" />
</joint>
  <link
   name="base_link">
   <inertial>
     <origin
       xyz="-6.9885731202049E-05 -0.00416139324316582 -0.00869330215629166"
       rpy="0 0 0" />
     <mass
       value="0.0597211231742216" />
     <inertia
       ixx="8.75969708228123E-06"
       ixy="4.52866635883372E-09"
       ixz="6.08505015303817E-10"
       iyy="1.34837706447811E-06"
```

```
      iyz="-2.27118460944342E-08"
      izz="7.78827401439364E-06" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://meshes/visual/base_link.STL" />
      </geometry>
      <material
        name="">
        <color
          rgba="0.752941176470588 0.133322334 0.5553333 1" />
      </material>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://meshes/visual/base_link.STL" />
      </geometry>
    </collision>
  </link>
  <link
    name="Right_Link">
    <inertial>
     <origin
```

```
      xyz="-0.055723 0.024002 0.00028555"
      rpy="0 0 0" />
    <mass
      value="0.035164" />
    <inertia
      ixx="4.3042E-05"
      ixy="-1.2425E-05"
      ixz="-1.1392E-11"
      iyy="4.237E-05"
      iyz="6.8486E-11"
      izz="8.5395E-05" />
  </inertial>
```

The launch.py

```
import os
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.substitutions import LaunchConfiguration, Command, PathJoinSubstitution, FindExecutable
from launch_ros.substitutions import FindPackageShare
from launch.actions import DeclareLaunchArgument, SetEnvironmentVariable
from launch_ros.actions import Node
from launch_ros.parameter_descriptions import ParameterValue
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource
from ament_index_python.packages import get_package_share_directory


# Function to generate the complete launch description, which includes setting up environment variables,
```

```
# launching Gazebo, spawning the robot, and bridging communication between ROS 2 and
Gazebo.


def generate_launch_description():


    # Defines whether simulation time should be used. This is important for synchronizing the
robot

    # with the simulated clock in Gazebo, ensuring consistent timing for topics and services.

    use_sim_time = LaunchConfiguration('use_sim_time')


    # Specifies the path to the URDF file that describes the robot's structure and joints.

    # This file defines the physical and kinematic properties of the robot for both visualization and
simulation.

    urdf_file_name = 'butterfly.urdf'

    path_to_urdf = os.path.join(

        get_package_share_directory('butterfly'),  # Locate the 'butterfly' package containing the
URDF

        'urdf',  # Folder containing the URDF files

        urdf_file_name  # The name of the URDF file to be used

    )


    # Set environment variable for Gazebo to locate the necessary robot resources.

    # This ensures that Gazebo knows where to find model and mesh files for the simulation.

    resource_path = get_package_share_directory('butterfly')

    set_gz_sim_resource_path = SetEnvironmentVariable(

        name='GZ_SIM_RESOURCE_PATH',

        value=resource_path

    )


    # Node to publish the robot's state (joint angles, transformations) to the ROS system.

    # This allows for visualizing the robot's movement and updating its pose information.
```

```
    node_robot_state_publisher = Node(
        package='robot_state_publisher',
        executable='robot_state_publisher',
        name='robot_state_publisher',
        output='screen',  # Prints the node output to the terminal for debugging
        parameters=[{
            'robot_description': ParameterValue(Command(['xacro ', str(path_to_urdf)]),
value_type=str)
        }]
    )

    # Include Gazebo launch with an empty world to start the simulation environment.
    # This sets up Gazebo with basic settings and loads an empty simulation world.
    gz_sim = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            os.path.join(
                get_package_share_directory("ros_gz_sim"),  # Package providing the Gazebo-ROS
integration
                "launch",
                "gz_sim.launch.py"
            )
        ),
        launch_arguments={"gz_args": "-r -v 4 empty.sdf"}.items(),  # Load an empty Gazebo world
with verbose logging
    )

    # Spawn the robot in Gazebo using the 'create' service, providing its initial position in the world.
    spawn_entity = Node(
        package="ros_gz_sim",
        executable="create",
        arguments=[
```

```
            "-name", "robot1",  # Name the robot instance as 'robot1'
            "-file", path_to_urdf,  # Load the URDF file into the simulation
            "-x", "0", "-y", "0", "-z", "1.4"  # Set the initial position of the robot in the Gazebo world
        ],
        output="screen"
    )

    # Bridge node to connect ROS 2 topics (e.g., joint states, velocity commands) with Gazebo
messages.
    # This allows for communication and control between ROS 2 and the Gazebo simulation.
    bridge = Node(
        package='ros_gz_bridge',
        executable='parameter_bridge',
        name='ros_gz_bridge',
        arguments=[
            '/joint_states@sensor_msgs/msg/JointState[gz.msgs.Model',  # Bridge for joint states
            '/cmd_vel@geometry_msgs/msg/Twist[gz.msgs.Twist'  # Bridge for velocity commands
        ],
        output='screen'
    )

    # Return the full launch description
    return LaunchDescription([
        DeclareLaunchArgument(
            'use_sim_time',  # Declare whether to use simulation time or real-world time
            default_value='false',  # Default to real-world time
            description='Use simulation time if true'  # Description of the argument
        ),
        set_gz_sim_resource_path,  # Set the environment variable for Gazebo resources
        node_robot_state_publisher,  # Publish the robot's joint states and transformations
```

```
        gz_sim,  # Launch Gazebo with an empty world
        spawn_entity,  # Spawn the robot model in the Gazebo world
        bridge  # Bridge ROS 2 topics to Gazebo for communication
    ])
```

Multimedia

Access the project team's photos and videos via the QR Code below:

If unable to scan the QR Code, or if it does not function as expected, kindly access the group's multimedia via the following link:

ELEC330 - Group 5 Team Butterfly - Assignment 1 Multimedia