Cyber-Physical Systems Design Project Assignment

Welcome to the project assignment for Cyber-Physical Systems! This is your opportunity to apply the concepts we've explored throughout the course in a hands-on, real-world setting. You'll work on designing, developing, and implementing a system that integrates computational and physical processes.

You will select one of the following four project options, each presenting unique challenges and learning opportunities. You can use your creativity in solving the project and apply knowledge acquired throughout the course. Your submissions must include **technical report** and **implemented code**. Technical report has to be in PDF format and it will include explanation of the methods you used, implementation details and achieved results. Your code must be clearly structured and commented so it is clear for others. You can get maximum of 20 points based on quality of your technical report and code. Minimum number of points is 10 in order to get a credit and finish this course. Deadline for project submission is **9. May 2025, 23:59** in E-learning. In case, you want to work on another relevant topic in CPS, please consult with me, so I can approve your project proposal.

A: Drone Control

In this project, your task will be to design and implement a control system for a multirotor drone. The drone must be able to takeoff, hover and land at different location. You should have full attitude control (roll, pitch, yaw). Model of the drone is in the supplemented files. You should first run *initialization.m* to load parameters into the working directory. You can then open *DroneModel.slx* and run it. You should be able to see visualization of the drone simulation with spinning propellers. For your control system, you can use any technique you want like Proportional, Integral, Derivative (PID) control, Linear Quadratic Control (LQR) or Model Predictive Control (MPC). You should evaluate your final design on the Simscape model and report time-domain characteristics of your control system such as overshoot, rise-time, settling-time and steady-state error. There are a lot of resources for drone modeling and

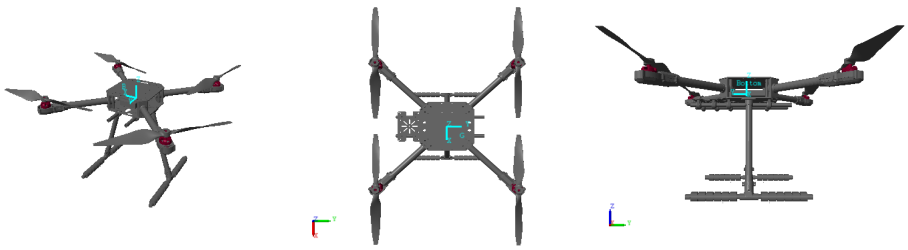

Figure 1: Virtual drone model in three views

control system design. You will find some of them in the project folder. Generally, you should first use a mathematical representation (linear/nonlinear differential equations) of the drone to design your controller. In case you are confident with your result, the code from MATLAB® & Simulink® can be loaded on real Pixhawk controller.

Required software: MATLAB® & Simulink®; SimscapeMultibody™

B: Robotic Arm Control

In this project, your task will be to design and implement a control system for a robotic arm. The robotic arm should be able to pick up a reference object (box) and move it to a different location. Try to design your system such that the arm and the payload avoids collision with itself. You should first run *initialization.m* to load parameters into the working directory. You can then open *RoboticArmModel.slx* and run it. You should be able to see visualization of the arm and the payload. To make this project a little easier, **Denavit-Hartneberg** parameters are provided in the *initialization.m* file named *dhparams*. This project was simplified such that the joint motion behaves according to double integrator ordinary differential equations. The control inputs of this system are therefore the joint accelerations. You will first need to understand **inverse kinematics** to design your end-effector trajectory. The end-effector is equipped with four vacuum cups. The code in SimscapeMultibody™ is designed, such that the vacuum cups stick to the payload after they first come to contact.

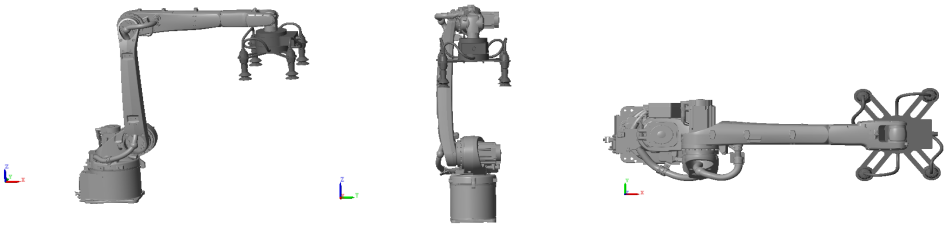

Figure 2: Robotic arm model in three views

Afterwards, you will have to design a controller for the joint angle accelerations to control the joint angles. Your controller should be able to follow a specified trajectory with the end-effector. At each point of the trajectory, you should be able to control the position and orientation of the end effector. Please report your achieved **Root Mean Squared error (RMSE)** accuracy in the trajectory following problem.

Required software: MATLAB® & Simulink®; SimscapeMultibody™

C: Vehicle Detection & Tracking

In this project, your task will be to implement vehicle detection and tracking algorithm in a traffic scenario generated in CARLA driving simulator. The supplemented files include a one minute recording sampled at 15 frames per second. Each frame has its unique ID corresponding to the filename. Each frame has also associated ground truth 2D bounding box JSON file with vertices of the bounding box in image coordinates and 3D bounding box JSON file including vertices of the box in world coordinates and also projected vertices to the image coordinates. The folder also contains camera intrinsic and extrinsic parameters. You can choose to track vehicles in **2D image coordinates or 3D world coordinates**. Try to address common issues in tracking such as static and dynamic occlusion. You should be able to assess your object detector and tracker using the supplemented ground truth bounding boxes. Typical metric for object localization is the **Intersection over Union (IoU)** metric. Please report the IoU in your technical report. You can use any technique you will find suitable. However, neural network based detector such as **You Only Look Once (YOLO)** will be necessary for robust detection of the vehicles. One of the most used tracking algorithm is the **Simple Online and Realtime Tracking (SORT)** which is very fast even in presence of many objects.

Figure 3: Vehicle detection and tracking in CARLA simulator

You can work in any programming language you want but I can recommend a Stone Soup library in Python which helps in designing the tracking algorithm. There is also a lot of examples in topics such as Kalman filtering and object association techniques.

Recommended software: Python; Stone-Soup

D: Flight Simulation & Autopilot Modes

In this project, your task will be to implement a functioning flight simulation with two autopilot modes, **heading change (HDG)** and **altitude hold (ALT)**. When both autopilot modes are active, the aircraft should not loose altitude when changing heading. You will first need to setup a **Flight Dynamic Model (FDM)** based on Equations of Motion (EoM). You can choose any aircraft model, but keep in mind that you will need to find the aerodynamic coefficients to properly simulate motion of that aircraft. For reference, you can find derivation of linearized EoM for a single flight condition and all the required parameters for a Boeing 747-200 in the project folder. You can work in any programming language, but you should be able to visualize the results of your flight simulation in software such as FlighGear. If you use Simulink® and AerospaceBlockset™, you can do this directly from Simulink®. When designing your

Figure 4: Flight simulator

control system, you should first focus on deriving the **State-Space model** (multidimensional linear model) of longitudinal and lateral directional dynamics. For your control system, you can use any technique you want such as Proportional, Integral, Derivative (PID) control, Linear Quadratic Control (LQR) or Model Predictive Control (MPC). After you are done, please report time-domain characteristics of your HDG and ALT autopilot modes in your technical report. These should include rise time, settling time, overshoot and steady-state error.

Recommended software: MATLAB® & Simulink®, AerospaceBlockset™, FlightGear