M.Sc. in Mechatronics Engineering

System level rapid development in mechatronics
WS 2023/24

(Milestone-2)

Prepared by
Ganesh Dola

Table of Contents

Section1: State machines

1) Introduction:

To implement state machines for Milestone 2, the following toolboxes and support packages must be installed:

- Stateflow Toolbox – for modeling logic using state charts
- Embedded Coder and Simulink Coder – for code generation
- MATLAB/Simulink Compiler (Windows: MinGW-w64 C/C++)
- Arduino IDE – for hardware integration
- MATLAB/Simulink Support Packages – for interfacing with external hardware
- MecRoka S-Function Library – custom blocks for simulation
- Simscape Multibody Contact Force Library – for modeling physical interactions

These components enable the design, simulation, and deployment of embedded state-based control systems.

2) Simulink Stateflow basics:

a) Simple state flow

- A Stateflow chart is created to toggle between two states: "On" and "Off", every 2 seconds.
- The output signal is set to 1 when in the "On" state and 0 when in the "Off" state.
- The chart is connected to a Scope block to visualize the output, which appears as a square wave alternating between 1 and 0.

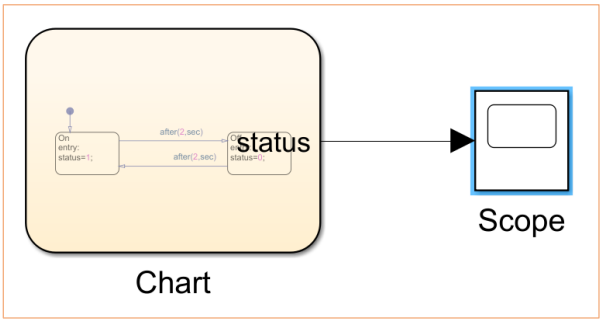


Fig: simple state chart

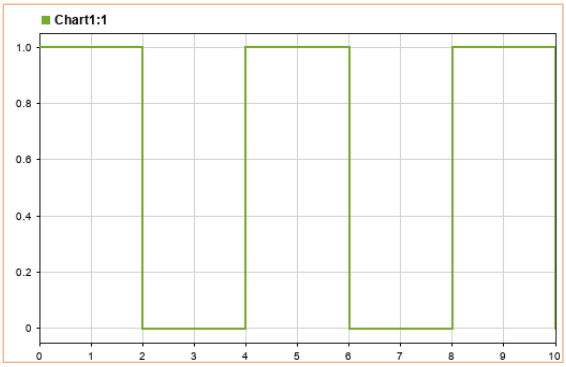


Fig: graph for simple state chart

b) charts with input & output ports

- A Constant block provides an input signal (myinput) with a value of 0, connected to the chart.
- The chart includes a new state called **error**, which is triggered when (myinput == 0).
- Upon entering the 'error state', the chart sets the **output signal status** to '**−1**', indicating a fault or invalid condition.
- The output is visualized using a **Scope block**, allowing real-time monitoring of the chart's response.

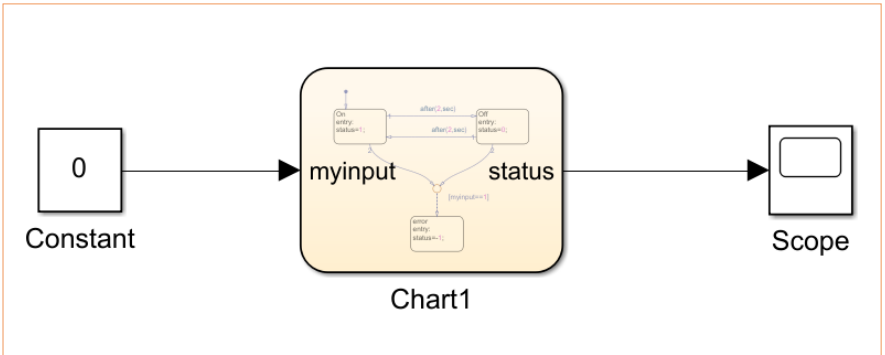


Fig: Chart with input & output ports

c) Chart combined with truth table

- To streamline complex decision logic, a **truth table** is used within the chart to evaluate input conditions and determine output actions.
- Each decision column (D1, D2, D3) maps to a specific row in the action table.
- The truth table outputs a flag that indicates whether the input into the truth table
    a. Input>0.5
    b. Input<=0.5

This logic ensures that the output flag is set based on boundary conditions of the input. The truth table simplifies branching logic and improves readability and maintainability of the control system.

Condition Table

	DESCRIPTION	CONDITION	D1	D2	D3
1	Example condition 1	my_input >0.5	T	F	-
2	Example condition 2	my_input <=0.5	F	T	-
		ACTIONS: SPECIFY A ROW FROM THE ACTION TABLE	1	2	3

Action Table

	DESCRIPTION	ACTION
1	Example action 1 called from D1 & D2 column in condition table	output_flag=1
2	Example action 2 called from D3 column in condition table	output_flag=-1
3	Example	output_flag=0

Fig: Truth table

- A sinus block with an amplitude of 1 is connected to the input of truth table.

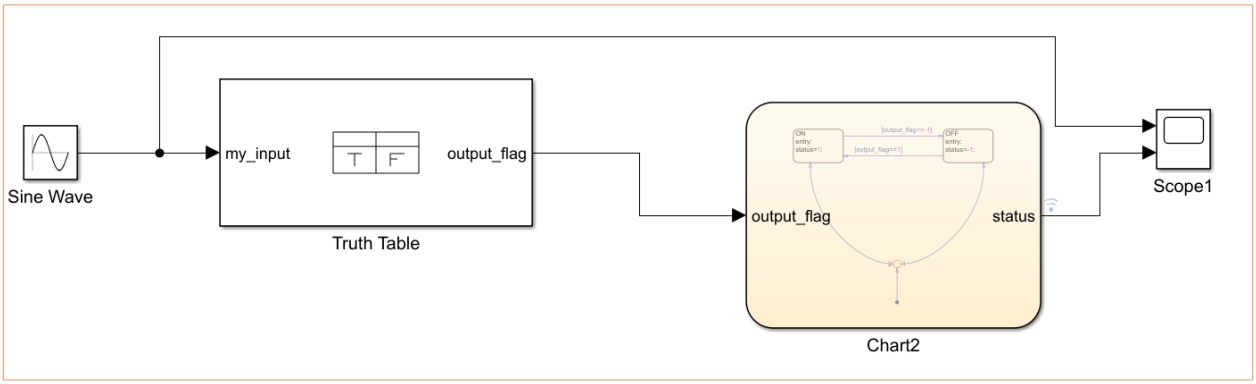


Fig: Chart combined with truth table

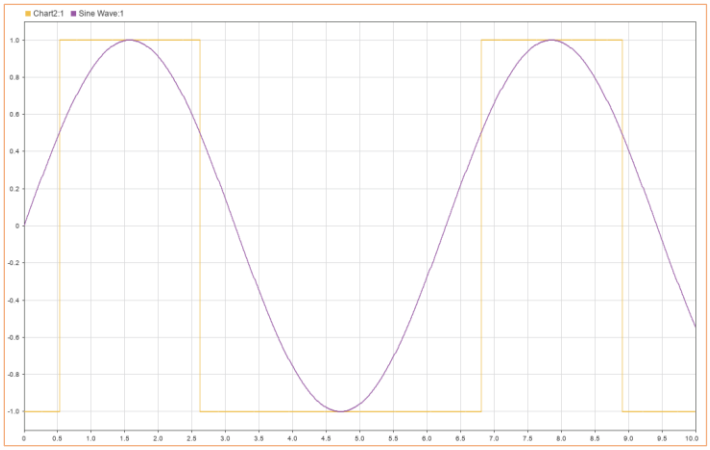


Fig: graph for chart combined with truth table

- From the above graph it is observed that the output status is showing ‘1’ if the input value is greater than 0.5 and ‘-1’ if it is less than or equal to 0.5.

d) Chart with Super state

This setup demonstrates how a super state can be used within a Stateflow chart to manage hierarchical logic and streamline control flow.

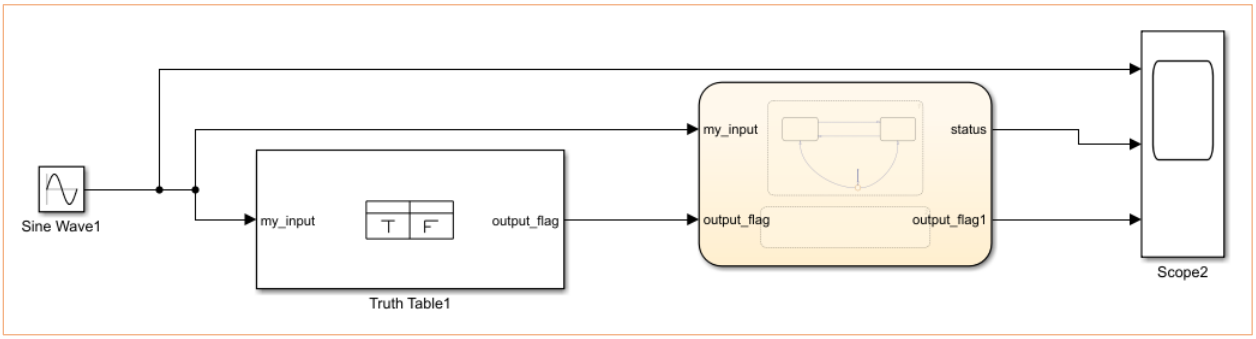


Fig: Chart with superstate inside

- A Sine Wave block generates a continuous input signal.
- The signal is passed through a Truth Table block to evaluate boundary conditions.
- The processed input (my_input) and decision flag (output_flag) are fed into a Stateflow chart containing a superstate, which outputs ‘status’ and ‘output_flag1’.
- A Scope block visualizes the output signals for analysis.

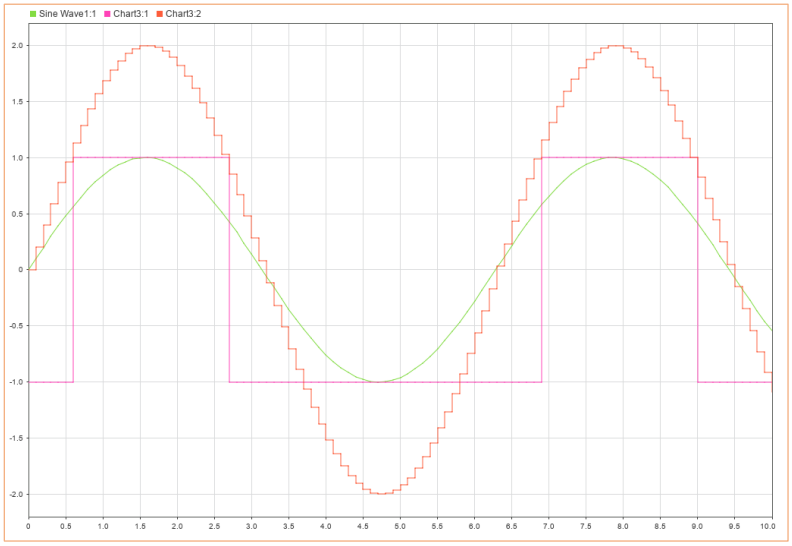


Fig: Graph for chart with superstate inside

- The sine wave output appears non-smooth due to solver settings.
- To improve signal resolution, the fixed-step size (fundamental sample time) should be adjusted to 0.01 seconds in the solver configuration.

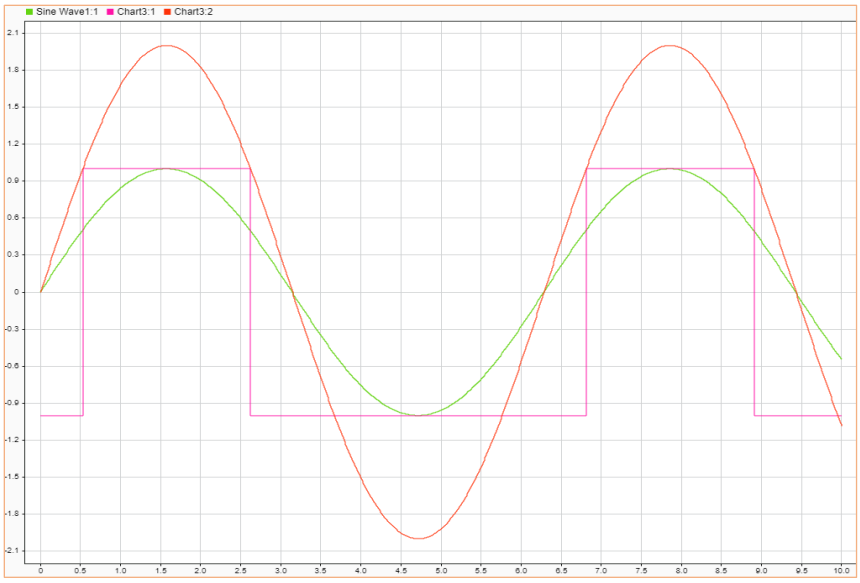


Fig: Smooth graph for chart with super state inside

e) Adding a simulink function

- A Simulink Function block is added to encapsulate reusable logic. In this example, the function multiplies the input value by two:
  - *my_output_value = multiply(my_input_value);*
- Inside the block, the function is defined as
  - *Simulink Function output = multiply(input)*
- This modular approach promotes clean design and simplifies function reuse across charts and models.

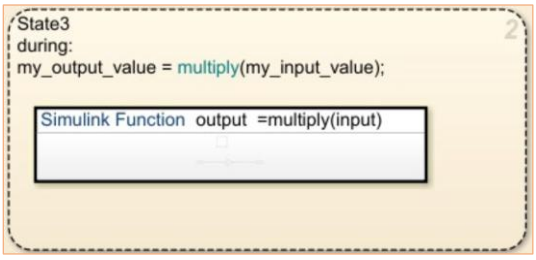


Fig: Simulink function

f) Time based trigger for chart execution

To enable periodic execution of the chart logic:

- A Function-Call Generator is connected to the chart, configured to trigger at 10 Hz.
- Input signals are provided via Sine Wave blocks, simulating dynamic conditions.
- The chart receives the trigger as an input event, allowing it to execute control logic at regular intervals.
- Output signals are visualized using a Scope block.

This setup allows for time-driven decision-making and supports iterative control loops within the chart.

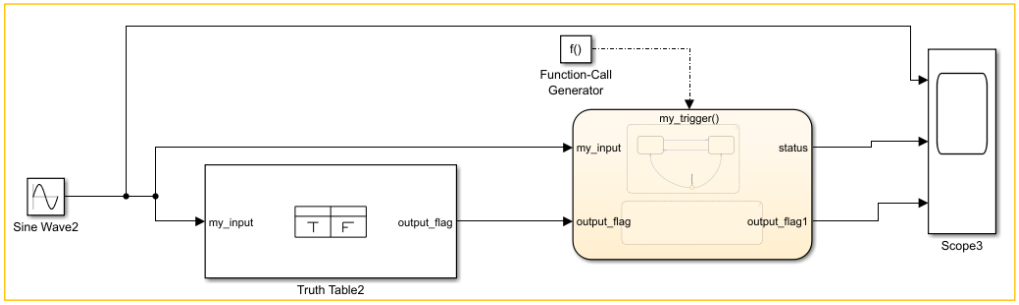


Fig: Chart with external trigger

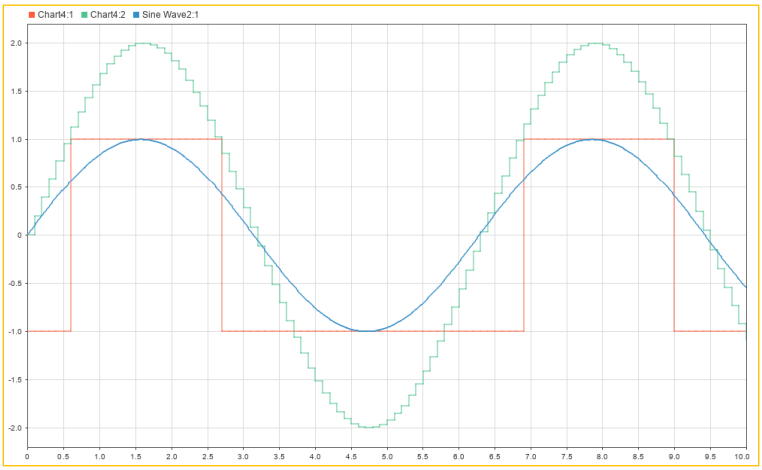


Fig: Graph for chart with external trigger

- The graph can be again smoothen by changing the fixed-step size in solver settings.

Section2: Code export

Arduino is an open-source electronics platform designed for easy hardware-software integration. It consists of:

- A physical microcontroller board (e.g., MKR1000)
- The Arduino IDE (Integrated Development Environment) used to write and upload code

1) Arduino IDE (Integrated Development Environment)

- Install Arduino IDE 2.1 and connect the **MKR1000** board to your computer.
- Test the setup using the "Blink" example:
    - Connect external LEDs on the two-wheel robot to pins 13 and 14 of the MKR1000.
- In Simulink, go to:
    - Model Settings → Hardware Implementation → Hardware Board
    - Select Arduino **MKR WiFi 1010** for compatibility.

2) Read sensor data

Blink Test Setup

- Use Pulse Generator blocks to send digital signals to:
  - Pin 13 → LED 1
  - Pin 14 → LED 2

This verifies digital output functionality.

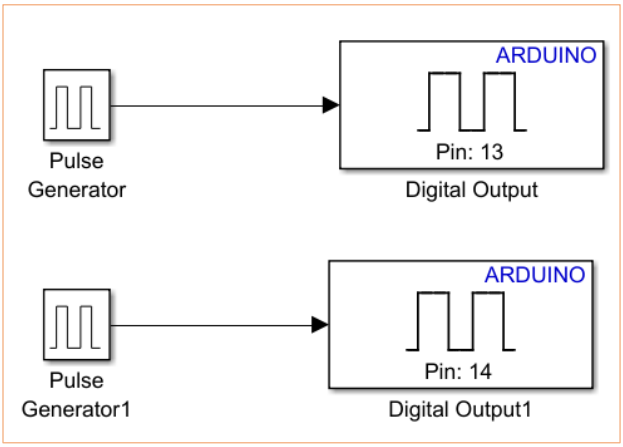


Fig: Connections for Blink test

Motor Driver Safety

To prevent unintended motor activation, set:

- Pin 6 and Pin 7 to 0 using Constant blocks.
- These pins control motor driver enable signals.

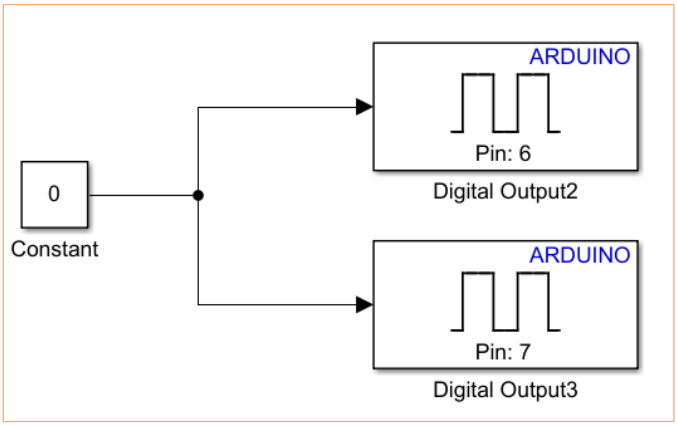


Fig: Connections to cut off the power supply

- Pin 8 is configured to read the line follower sensor status from the robot.

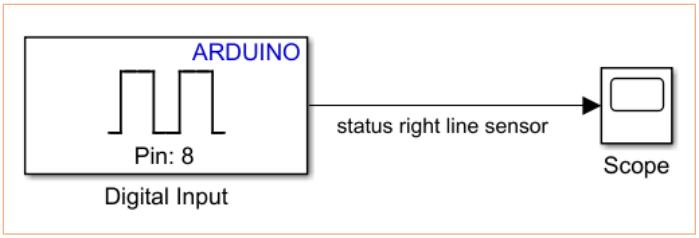


Fig: Connection for line follower sensor status

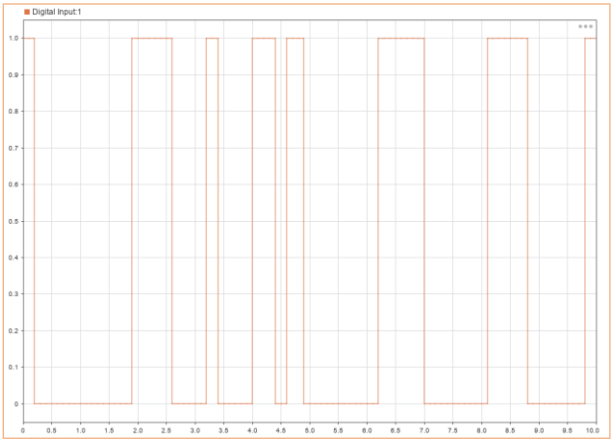


Fig: Reading data from h/w with Simulink

To acquire motion and orientation data from the robot, the MecRoka S-Function Library—developed by the Mechatronics and Robotics Group at Hochschule Kaiserslautern—is integrated into the Simulink environment.

.

These components enable seamless communication between Simulink and embedded hardware.

An Inertial Measurement Unit (IMU) typically includes:

- Gyroscopes – measure angular velocity
- Accelerometers – measure linear acceleration

- A Simulink model is created using MecRoka blocks to interface with the inbuilt IMU.
- The diagram illustrates how sensor data flows from the IMU into Simulink for processing and visualization.

- The IMU readings are plotted over time, showing dynamic changes in motion and orientation.
- This graph helps validate sensor performance and supports further control system development.

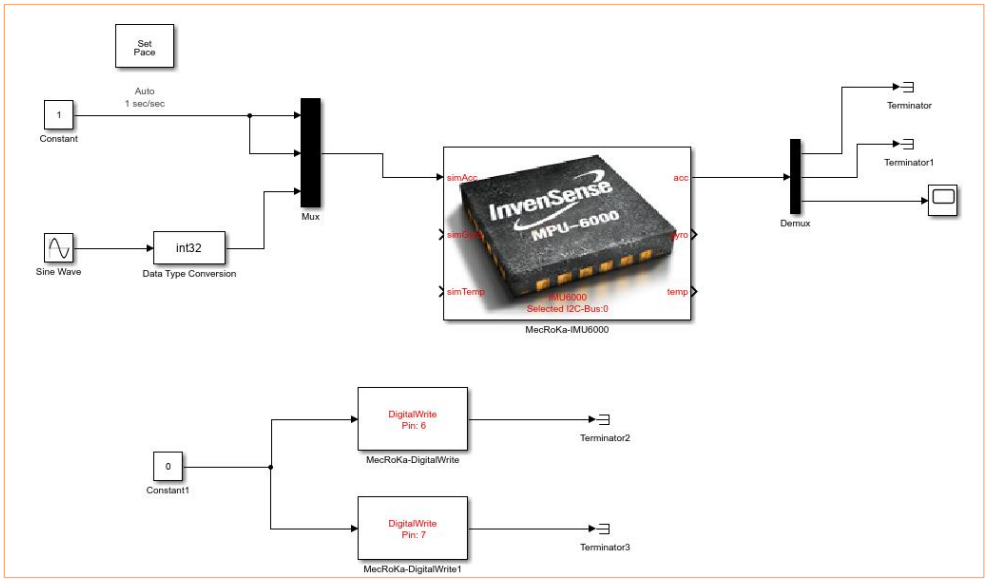


Fig: Reading IMU data using the MecRoka library

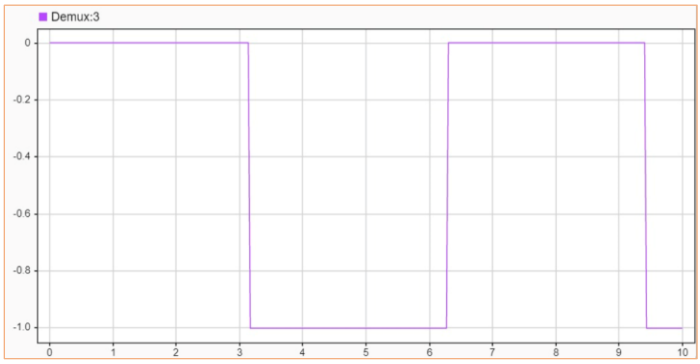


Fig: Graph-IMU data

Section3: Setting up and testing the two-wheel robot

The objective of this session is to successfully run the two-wheel robot model in both simulation and real-time environments.

Required Component:

- Install the Simscape Multibody Contact Forces Library via MATLAB Add-Ons.
- Download the simulation package:
  MECROKA_TWR_BLDC_v2.1d.zip, which contains:
  - MECROKA_TWR_BLDC_01_main_model_v2.1.slx – for multibody simulation
  - MECROKA_TWR_BLDC_02_controller_v2.1.slx – for controller logic

Solver Configuration:

Two separate solver types are used to accommodate different simulation needs:

- **Continuous Solver** (used in the main model):
  - Computes continuous states using numerical integration
  - Suitable for physical modelling and multibody dynamics
- **Fixed-Step Solver** (used in the controller model):
  - Solves the model at uniform time intervals
  - Ideal for real-time execution and code generation

Model Architecture:

The simulation model consists of two main components:

1. **Virtual Robot**
   - Represents the physical robot in simulation
   - Inspired by the inverted pendulum setup from Milestone 1
   - The base block is replaced by **wheels**, which are driven by **angular displacements** instead of force inputs
2. Controller
   - Represents the control logic running on the robot's microcontroller (**MKR1000**)
   - Interfaces with sensors and actuators to manage robot behavior

This dual-model approach ensures accurate physical simulation while maintaining compatibility with embedded hardware for real-world deployment.

1) Virtual robot:

This section outlines the modeling and testing of the two-wheel robot in both virtual and physical contexts using Simulink.

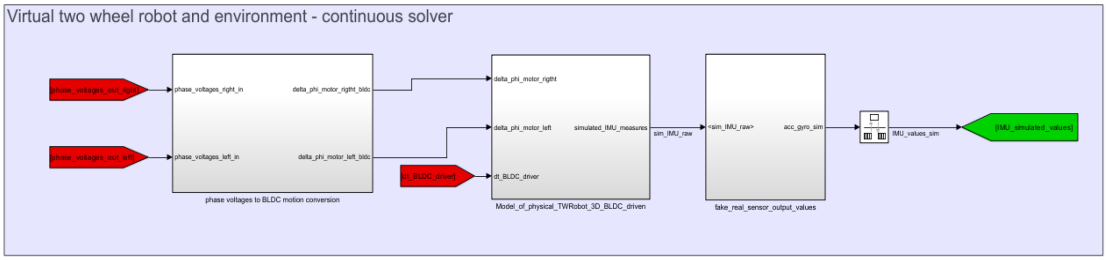


Fig: The simulink model of the real two-wheel robot H/W

- The virtual robot is based on the inverted pendulum concept from Milestone 1, but with key modifications:
  - The base block is replaced by wheels, allowing motion through angular displacement rather than force.
  - These angular steps serve as the input signals to drive the robot's movement.
- The model uses a continuous solver to simulate real-world dynamics with high fidelity

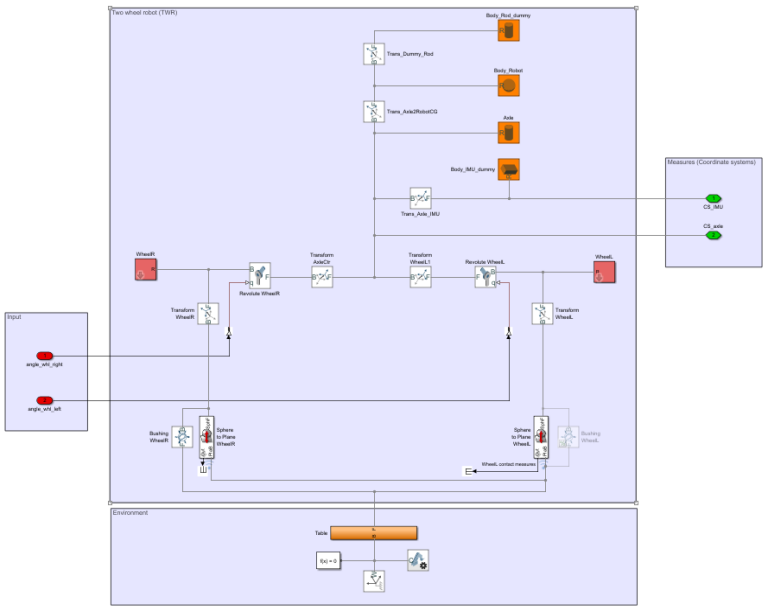


Fig: Physical robot

Physical Robot Mode:

To simulate the actual hardware behavior, an additional block is introduced before the multibody model:

- This block calculates the motor position based on the voltage of the three BLDC motor phases.
- This setup bridges the gap between electrical input and mechanical motion, enabling realistic control simulation.

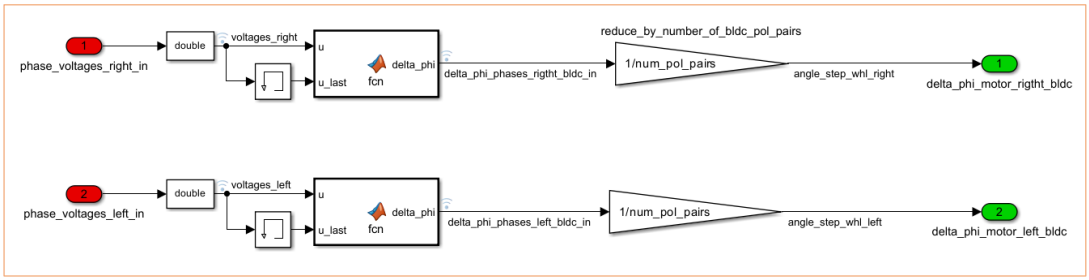


Fig: Phase voltage to BLDC conversion connections

Phase Voltage to BLDC Conversion:

- A dedicated block is added after the multibody simulation to compute the real IMU output.
- This allows for accurate sensor feedback and supports closed-loop control strategies

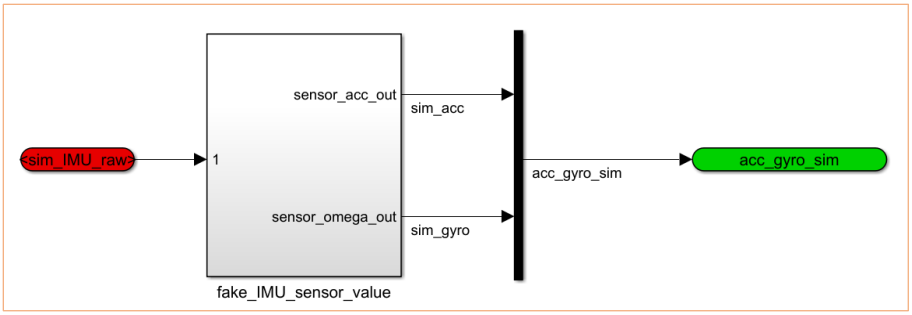
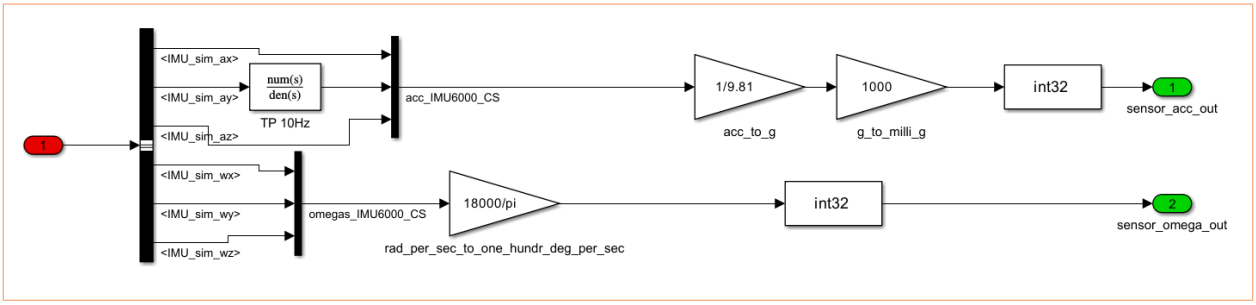


Fig: Real sensor output values



- From this, the simulated acceleration values are delivered as an input to the controller.

2) The controller

The controller is designed to manage real-time execution, sensor feedback, state estimation, and motor control for the two-wheel robot. It consists of six key components:

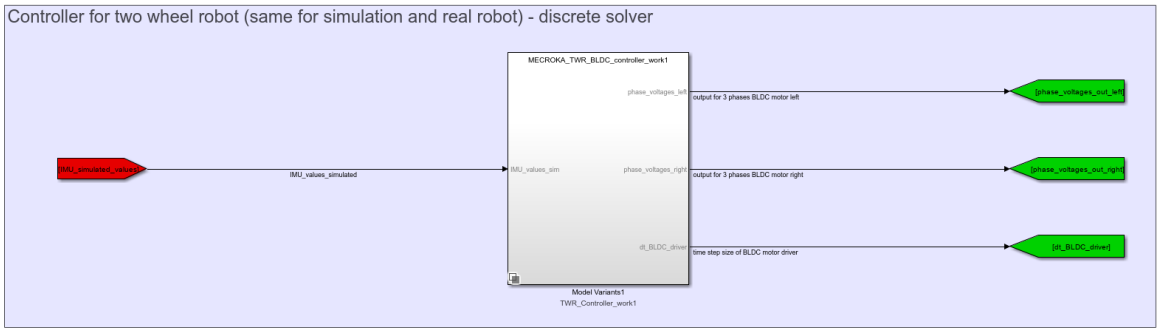


Fig: Controller for a two wheel robot

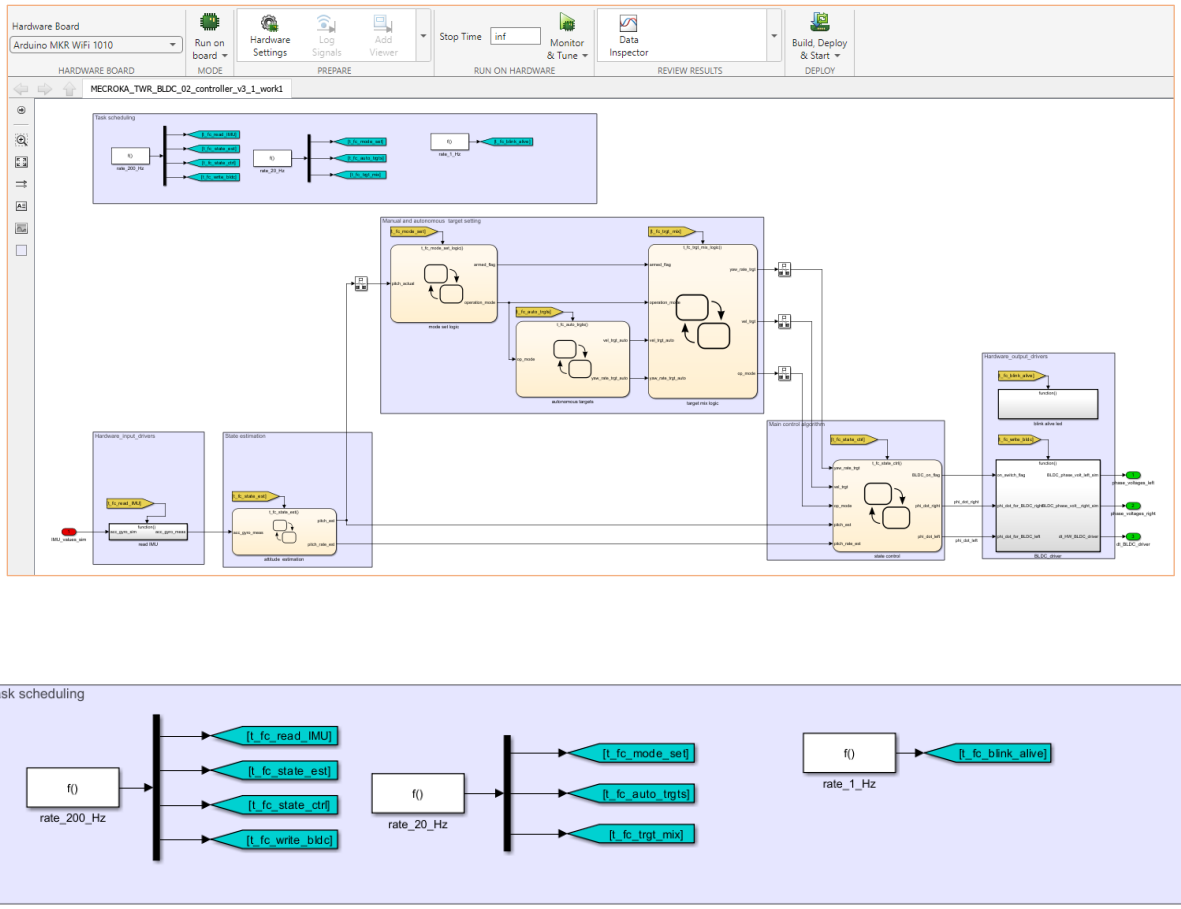


Fig: Generation of trigger signals

Task Scheduling and Trigger Management:

- A **triggering mechanism** coordinates the execution of multiple charts and tasks.
- Events are dispatched based on system requirements using a **Function-Call Generator**.
- This ensures synchronized operation across control modules.

IMU Sensor Driver:

- Interfaces with the **Inertial Measurement Unit (IMU)** to retrieve sensor data such as acceleration and angular velocity.
- In simulation, input values are passed through directly to mimic real sensor behaviour.

State Estimation:

- Estimates the robot's **pose** using sensor data.
- A **complementary filter** is applied to calculate the **pitch angle**, which cannot be measured directly.
- **Velocity** is obtained from the IMU and used as part of the estimation process.

Control Algorithm:

- Computes the required **motor control signals** based on the robot's current state and target behaviour.
- Implements a sequence of **mathematical and logical operations** to ensure stability and responsiveness.
- The algorithm is modular and adaptable to different control strategies

Motor and LED Drivers:

- Manages output signals to **motors and LEDs**.
- In simulation, values are passed through to represent hardware interaction.

Target Setting Block:

- Defines **task-specific targets** for the controller.
- Targets are configured via **Stateflow charts**, allowing flexible behaviour based on mission objectives.

This modular controller design supports both simulation and real-time deployment, enabling robust control of the two-wheel robot.