UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Wheeled-Legged Balancing Robot

Electrical & Computer Engineering

Team 3
Gabriel Gao, Jerry Wang, Tony Yuan

2023/12/05

Introduction

Facing the challenge of terrain navigation in traditional wheeled robots and the speed limitations in legged robots, our project is focused on creating a hybrid wheel-legged robot. This innovative design aims to overcome the shortcomings of both types in urban environments.

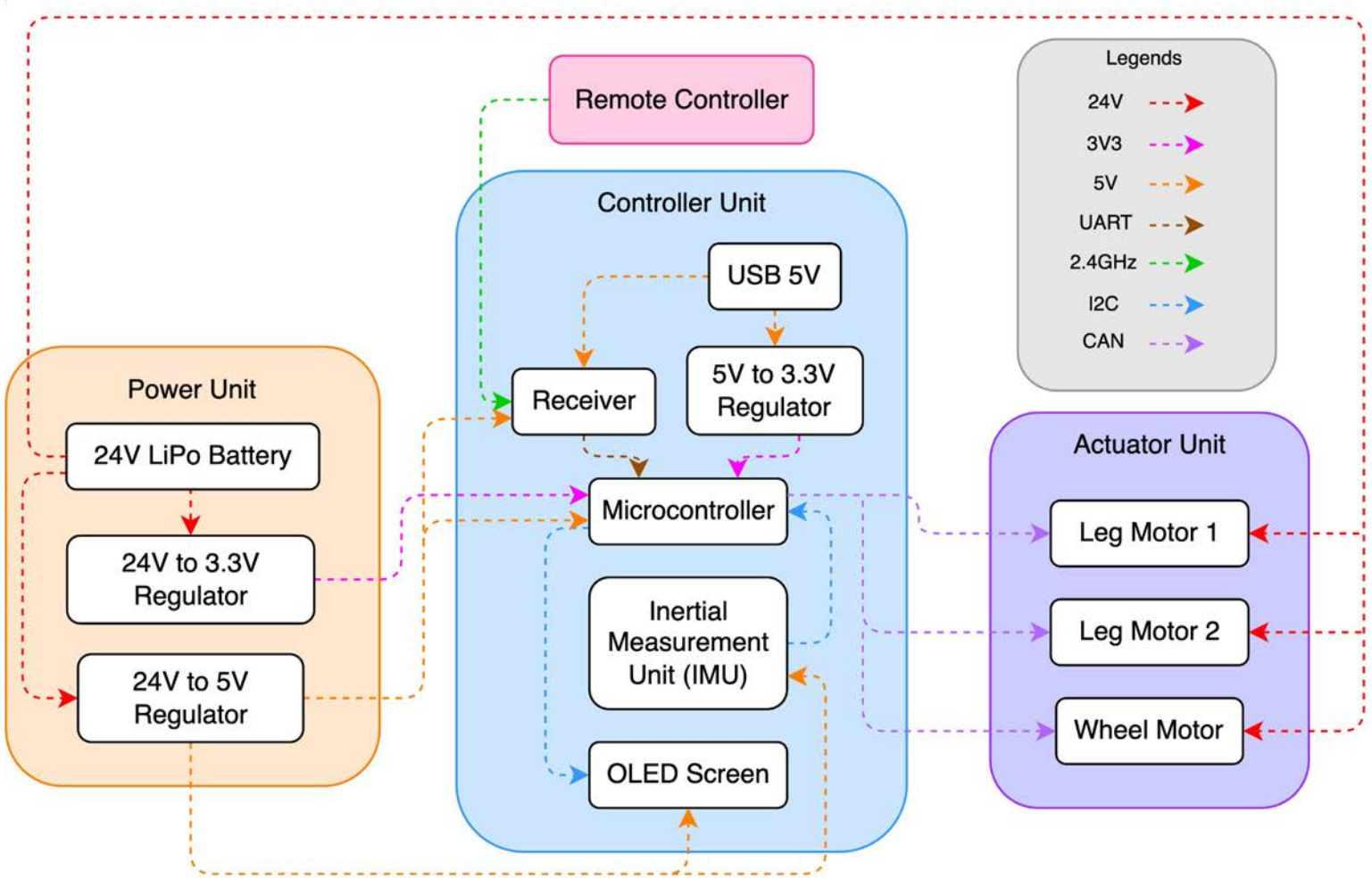
Type of platform	Ability to overcome terrains	Moving Speed
Wheeled robot	Low	High
Legged robot	High	Low

Objective

Our project introduces the hybrid wheel-legged robot as a solution to these challenges. This design combines the speed and efficiency of wheeled robots with the adaptability and terrain-navigating ability of legged robots.

With a unique leg mechanism acting as a dynamic suspension system, our robot is aim to efficiently traverse diverse urban terrains, providing a robust platform suitable for a wide range of applications

System Block Diagram



Actuator Legs

Requirement	Verification
• The motor should be able to supply a continuous torque of at least 4Nm ± 5% at 120RPM ± 5%.	• Using a script to read the torque value from the data given by the motors or Using the rotary torque sensor to measure the torque. Check whether the values from either method are larger than 4 Nm ± 5%. • Using a tachometer to measure the motor's RPM to confirm it can achieve 120 RPM ± 5% under the specified torque load.
• The motors should be able to be powered by 20 - 27V.	• Using an adjustable power supply to test the motors at various voltages within the range of 20-27V to ensure consistent and efficient performance.
• The motors must maintain a communication rate of 1kHz ± 10HZ with the STM32 through the CAN bus.	• Use a CAN bus analyzer or other appropriate testing tools to measure the communication rate between the motors and the STM32. Ensure the communication rate remains stable at 1kHz ± 10HZ.
• The encoder of the motor should have an angle resolution of less than 1 degrees • Able to remember the correct motor zero point after power cutoff.	• Run multiple commands to rotate the motor to a certain angle, and measure whether each angle rotation matches the control command. Rotate the motor at a certain angle and read the motor angle feedback, comparing whether the angle feedback is within 1 degree of the actual difference. • After setting the motor to zero point, power off and rotate the motor angle. After powering on, check if the motor remembers the correct zero position

Wheel Drive

Requirement	Verification
• M3508 motor maintains a communication rate of 1kHz ± 10Hz with the STM32.	• Use a CAN bus analyzer or other appropriate testing tools to measure the communication rate between the motors and the STM32. Confirm the communication rate remains stable at 1kHz ± 10HZ.
• The motors should be able to be powered by 20 - 27V.	• Using an adjustable power supply to test the motors at various voltages within the range of 20-27V to ensure consistent and efficient performance.
• M3508 motor delivers a continuous torque of at least 0.1Nm and achieves a speed of at least 2000 RPM. • M3508 motor can move the robot under a load of around 2.5lb.	• Using a script to read the torque value from the data given by the motors or Using the rotary torque sensor to measure the torque. Check whether the values from either method are larger than 0.1 Nm ± 5% • Using a tachometer to measure the motor's RPM to confirm it can achieve 2000 RPM ± 5% under the specified torque load. • Load the robot with a weight of 2.5lb and verify the motor moves the robot effectively.
• The encoder of the motor should have an angle resolution of less than 1 degrees	• Run multiple commands to rotate the motor to a certain angle, and measure whether each angle rotation matches the control command. Rotate the motor at a certain angle and read the motor angle feedback, comparing whether the angle feedback is within 1 degree of the actual difference.

PCB & Microcontroller

Requirement	Verification
• Capable of processing inputs efficiently and directing outputs to various peripherals.	• Perform an on-off test and voltage drop test on the entire PCB board using a multimeter. After the first two tests are successful, power up the PCB board and measure the voltage of the circuit components to see if they are regular. Then using scripts to test the functionality of each interface.
• Must have protection mechanisms against power surges or short circuits.	• Check the input and output capacitors before powering the PCB board to ensure they can filter voltage fluctuations. • Using the adjustable DC power supply to test the design under various voltage and current conditions to ensure protection mechanisms work effectively.
• STM32F103 should run at 72MHZ ± 5MHZ.	• Use a frequency meter to confirm the operating frequency of the STM32F103 during operation.
• PCB should consume $\leq$ 1W to ensure efficient power consumption.	• Using the adjustable DC power supply, which can show the input voltage, input current, and input power, to measure the power consumption of the design to ensure it does not exceed 1W.
• Maintaining Can bus load under 80% between the IMU and the motors to ensure stable communication.	• Run a script and Use a CAN bus analyzer to monitor the CAN bus load, ensuring it stays below 80% for stable communication.
• Capable of processing and interpreting signals from the IMU accurately.	• Put the PCB board on a horizontal table. Then, power the PCB board and run the script to check whether the IMU returns the zero point to the microcontroller. Then, rotate the PCB board 45 degrees counterclockwise 8 times and check whether the IMU returns the correct data for each rotation.
• Minimize the time used in signal processing for real-time applications to 5ms± 5%.	• Using the inner clock to record signal processing time during the operation.
• Must provide precise control signals to the motors via the CAN bus.	• Connect the PCB board and the motors and use the scripts to send the rotation signal to the motors. Check whether the motors rotate to the desired position.

Attitude Sensing

Requirement	Verification
• Low bias error and drift. • Low noise levels, High resolution	• Read the data from IMU in a normal environment and Compare it with the data from the reference sensor. • Let the PCB work at a high electromagnetic interference environment and check the return data with the reference sensor.
• Suitable range and high sampling rate to get reasonable data.	• In a stable state of the IMU, connect the data output pins of the IMU to an oscilloscope and observe the waveform of the output signal. By measuring the time interval between two consecutive peaks, we can estimate the sampling rate.
•Ability to be calibrated and temperature-compensated	• Use a script to send the calibrating signal to IMU and use the return data to check whether the IMU is calibrated. Then, Keep monitoring the temperature data from IMU during the operation to check whether it can be temperature-compensated.
•Low power consumption	• Using the adjustable DC power supply, which can show the input voltage, input current, and input power, to measure the power consumption during operation.
•Common interfaces (I2C, SPI, UART, or CAN)	• Using simple scrips to check whether the microcontroller can send the signals to IMU and receive the IMU signals.

Remote Control

Requirement	Verification
• The user can use this remote control to send commands to the robot successfully.	• Using the script on the microcontroller to print the commands sent by the remote controller so that we can visualize the commands on the computer screen.
• The remote control must maintain reliable communication with delay $\leq$ 10ms.	• When the microcontroller receives the data from the remote controller, we can add the time stamp at that moment. Then we can let the remote controller send the same signal continuously. The time interval between the time stamps is the communication delay. Check whether the delay $\leq$ 10ms.

Requirement	Verification
• The remote control must maintain an emergency kill switch to stop the robot in 2s.	• Let the robot run at a low speed and then toggle the emergency kill switch. Use a timer to record the stop time and check whether the robot is killed $\leq$ 2s.

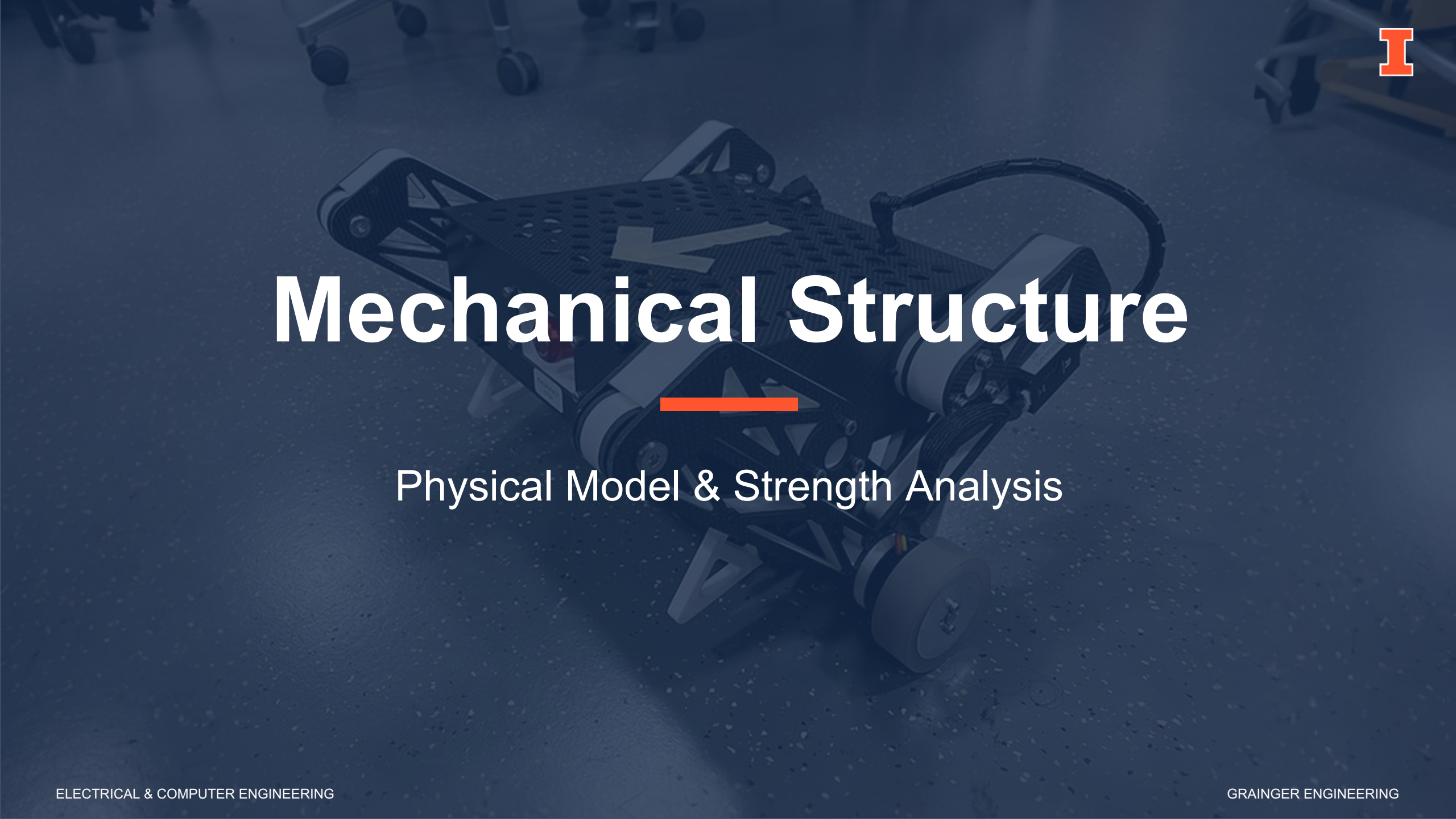
Power

Requirement	Verification
DC-DC Converter 1: • Input: 20-27V Output: 5V ± 0.5V, ≥ 1A.	• Use a multimeter to measure input and output voltage and current to ensure they are within specified limits.
DC-DC Converter 2: • Input from Converter 1 Output: 3.3V ± 0.3V, 500mA ± 5%.	• Use a multimeter to measure input and output voltage and current to ensure they are within specified limits.
Battery: • Supply: 20-27V, 20A peak, 5A continuous Runtime: ≥ 30 minutes at 5A.	• Use a multimeter to measure the output voltage and current from the battery under load. Use a timer to ensure it lasts for at least 30 minutes at a 5A load.

Requirement	Verification
Safety and Flexibility: • Quick power cut-off: ≤ 200ms Switch between battery and wired power.	• Test the cut-off response time with a stopwatch or electronic timer. Check the seamless switching between power sources under operation.

Outline

- Mechanical Structure
- Electrical System
- Embedded Software
- Control System

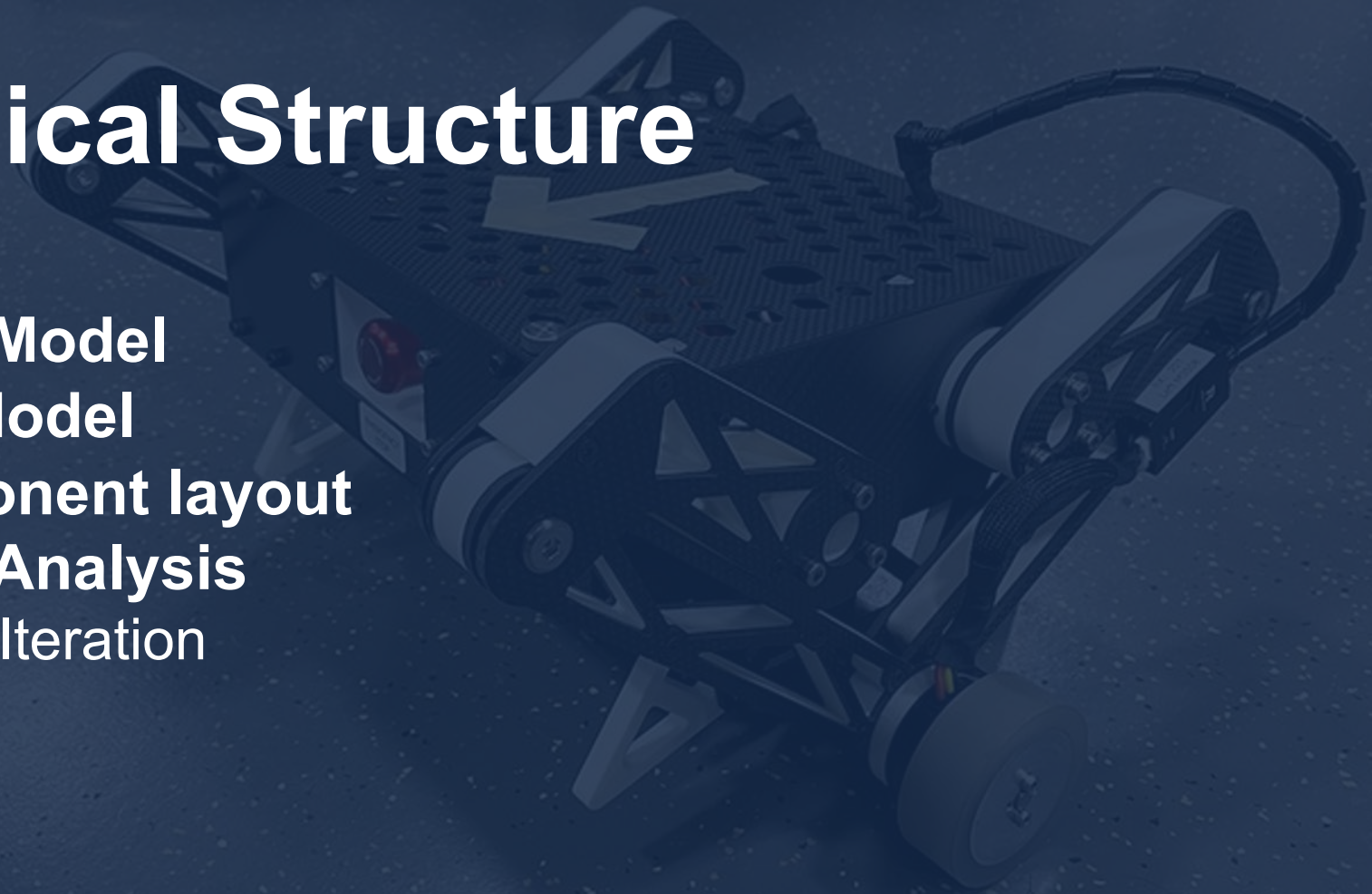


Mechanical Structure

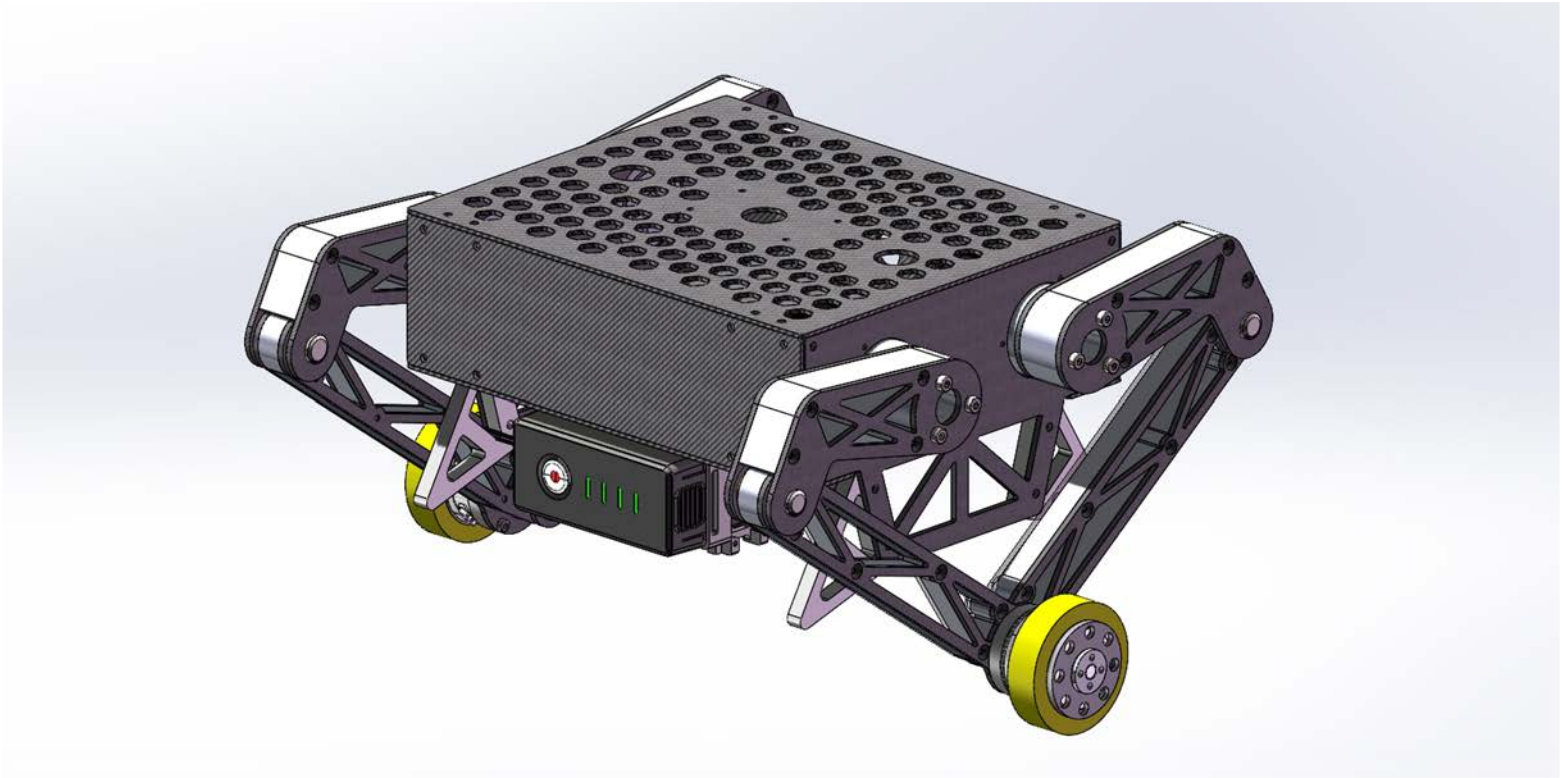
Physical Model & Strength Analysis

Mechanical Structure

- **Physical Model**
  - **CAD Model**
  - **Component layout**
- **Strength Analysis**
  - Design Iteration

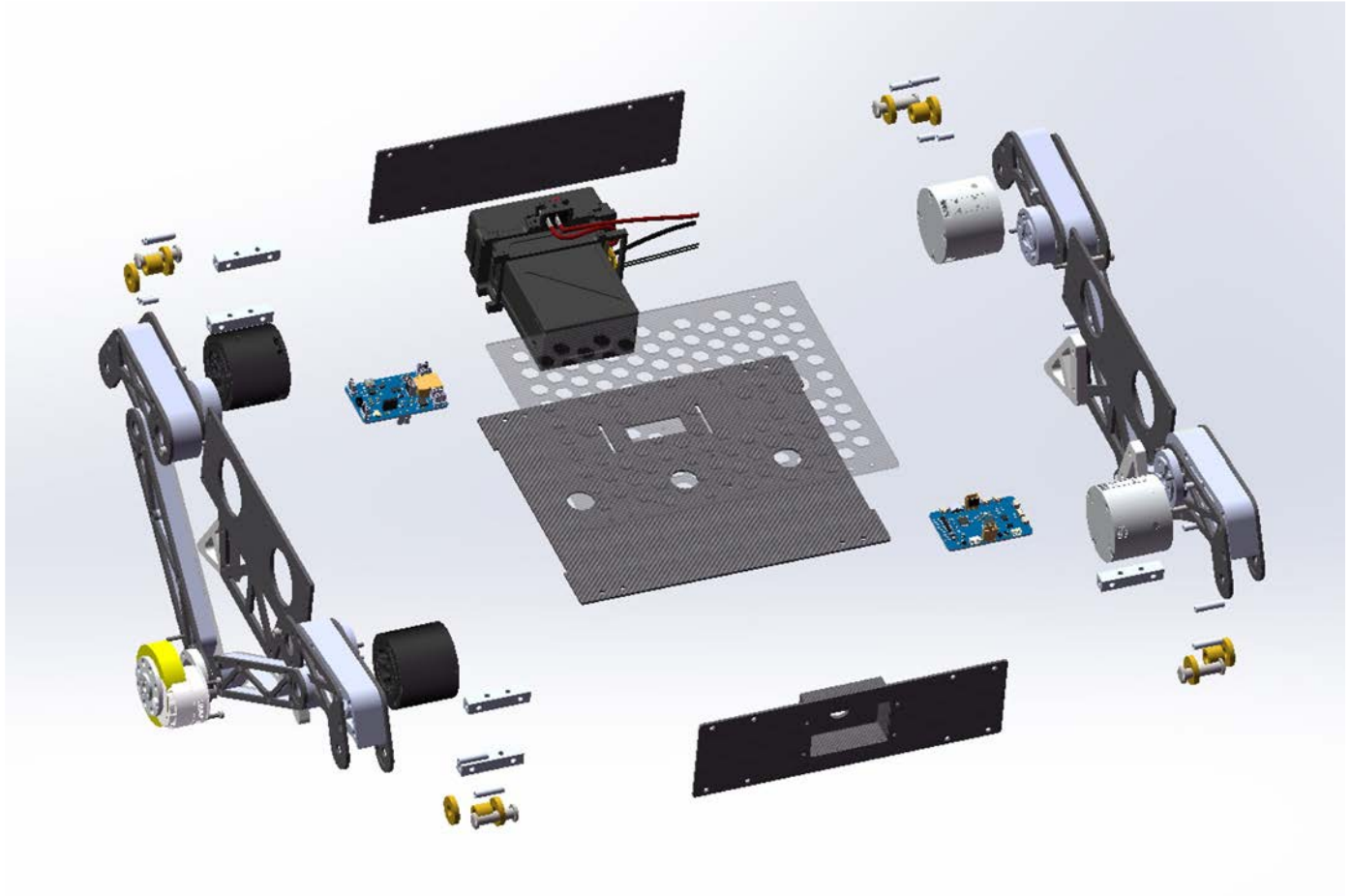


CAD Model



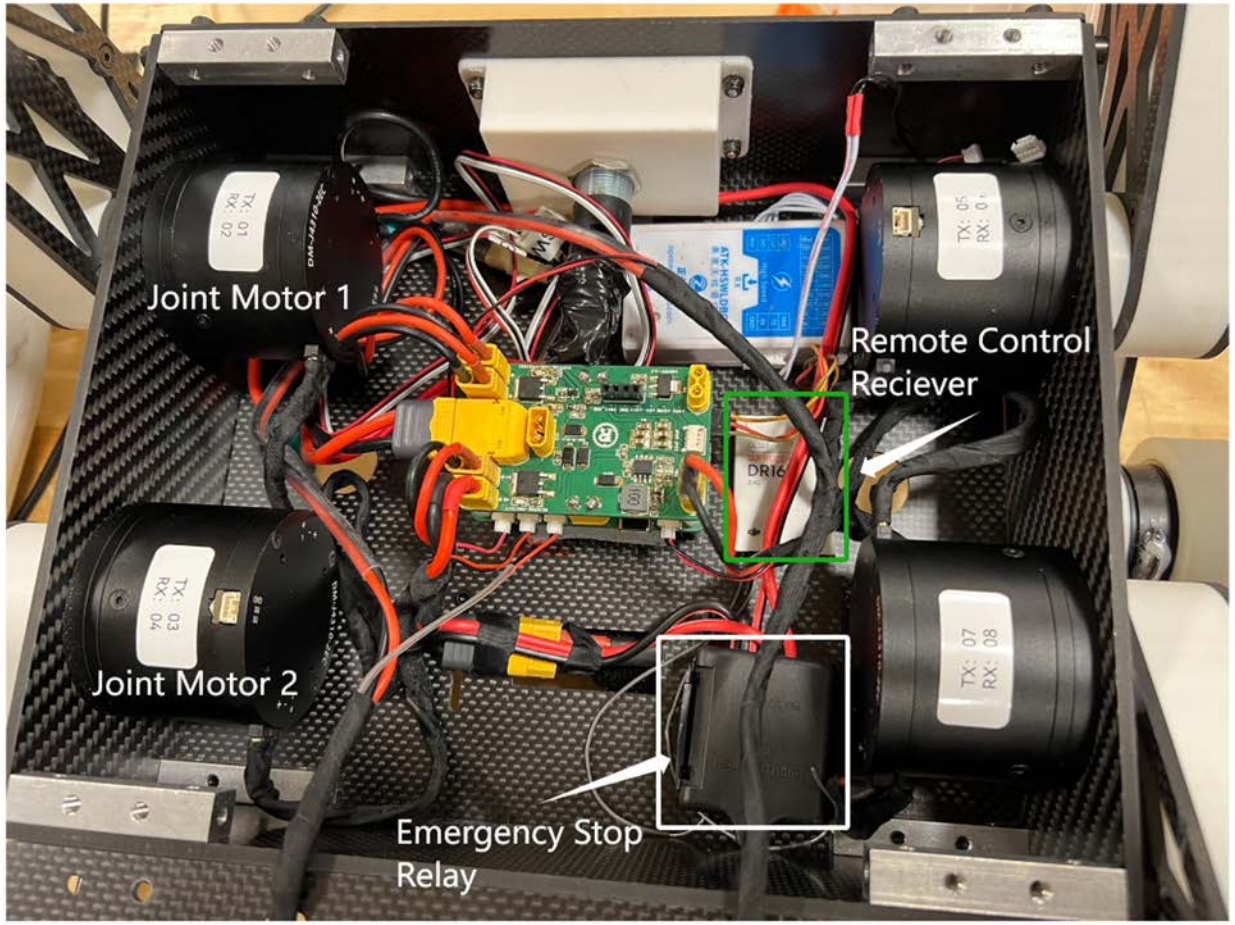
Assembly of robot

CAD Model



Exploded View

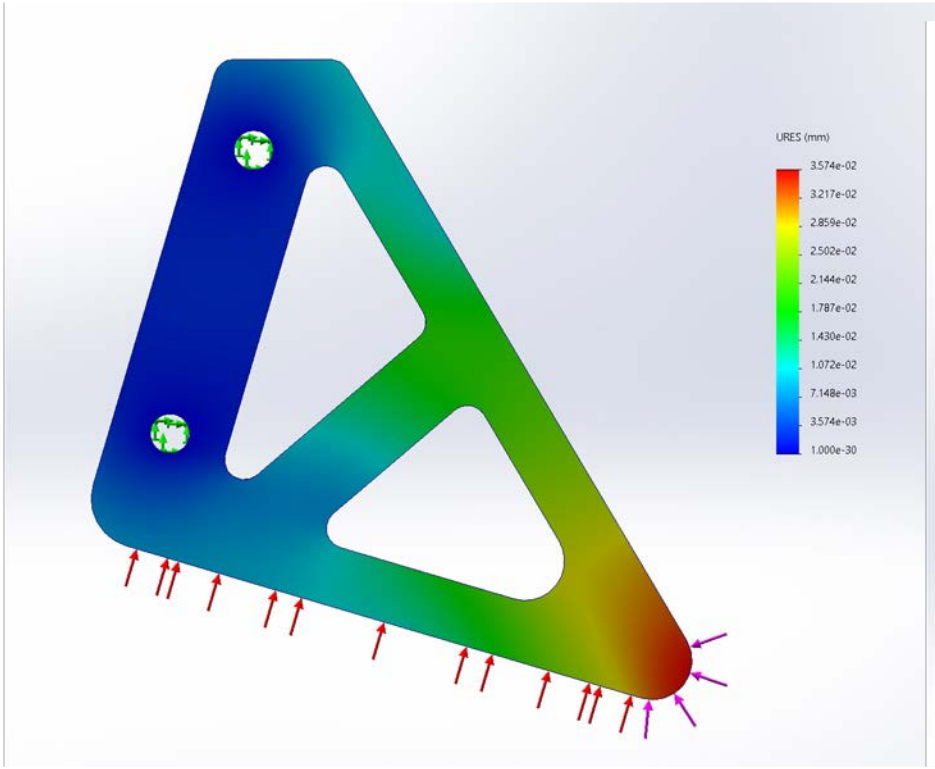
Component Layout

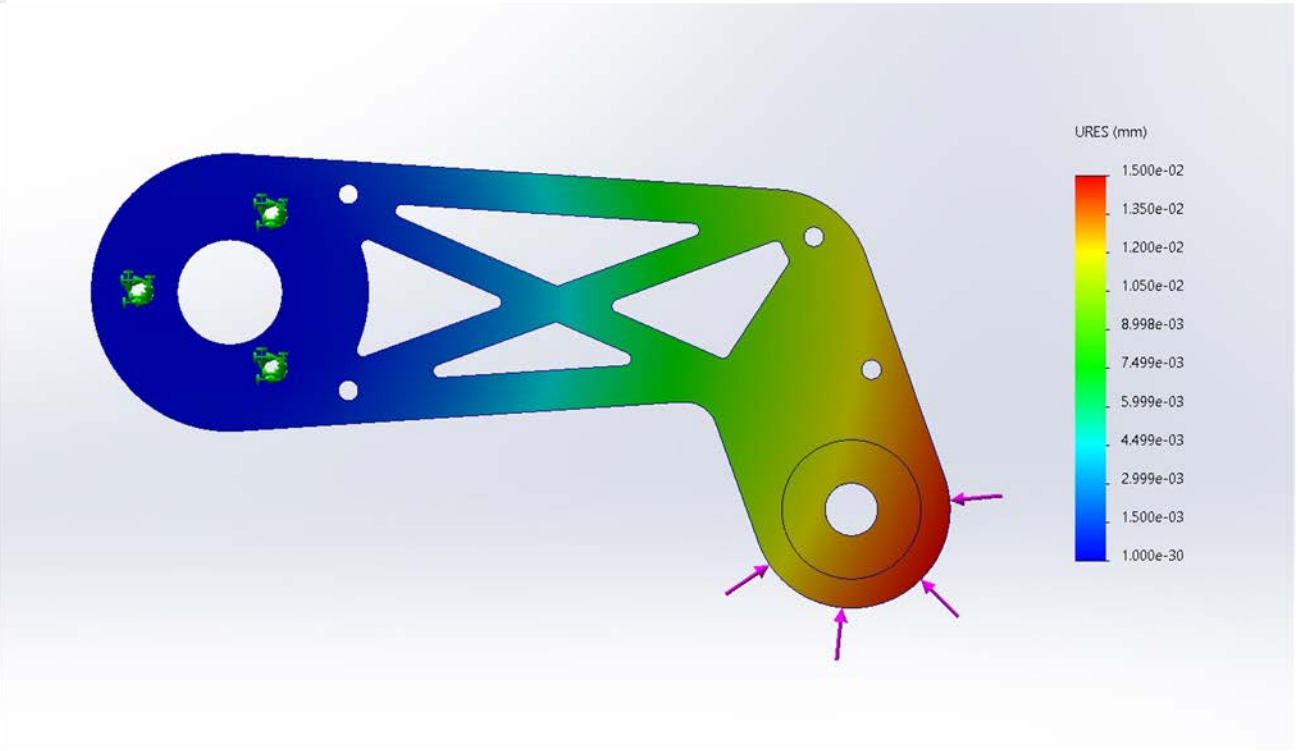


Inside the robot

Safety Component

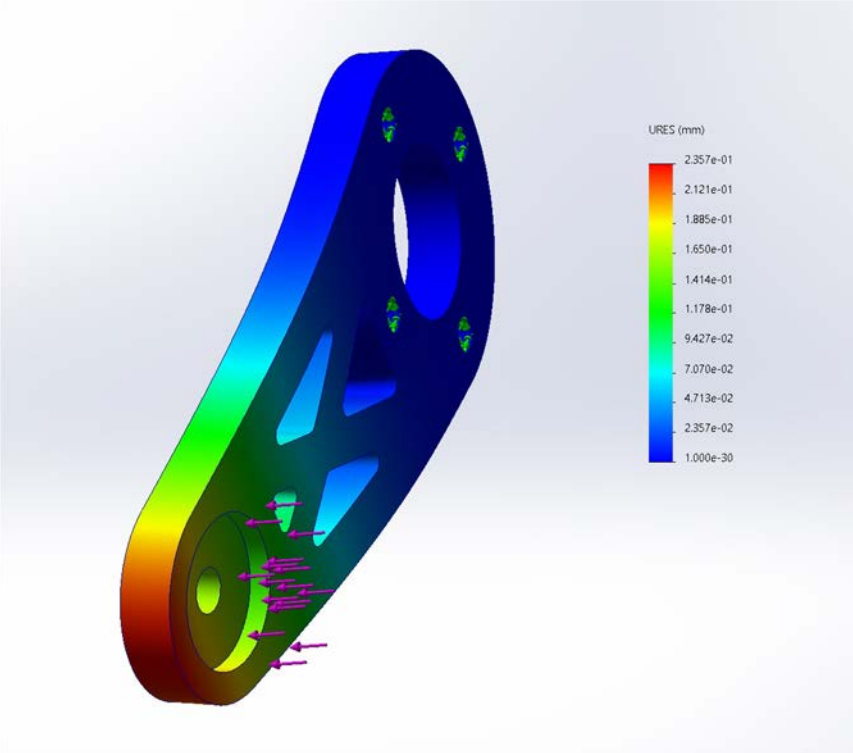
Strength Analysis

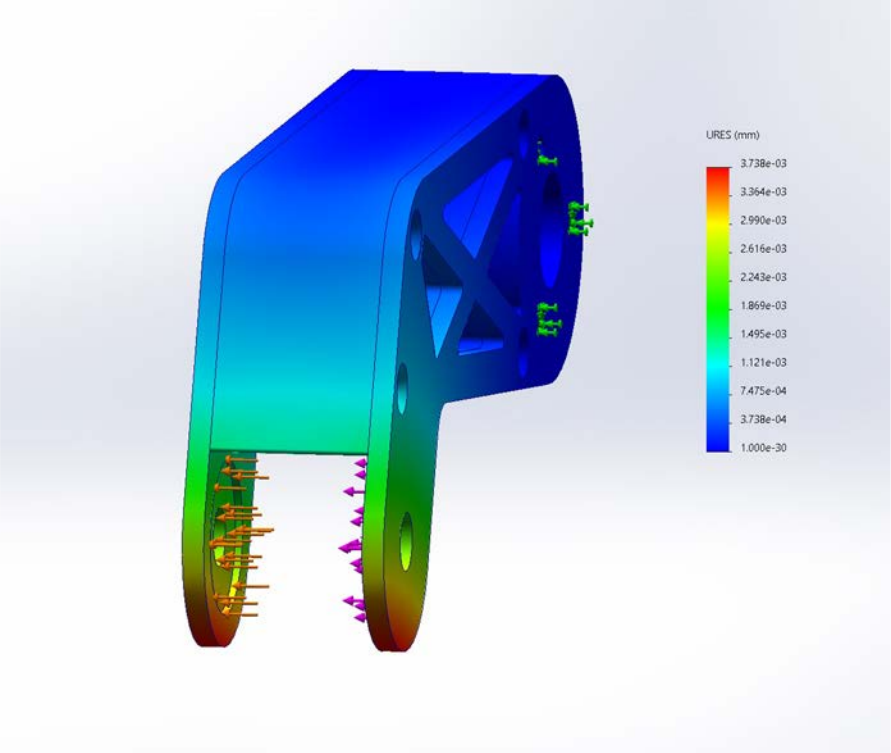




Implement static force analysis to assess the structural integrity and performance of components under various loads and conditions.

Design Iteration





First Version

Design: Single-layer thick aluminum plate.

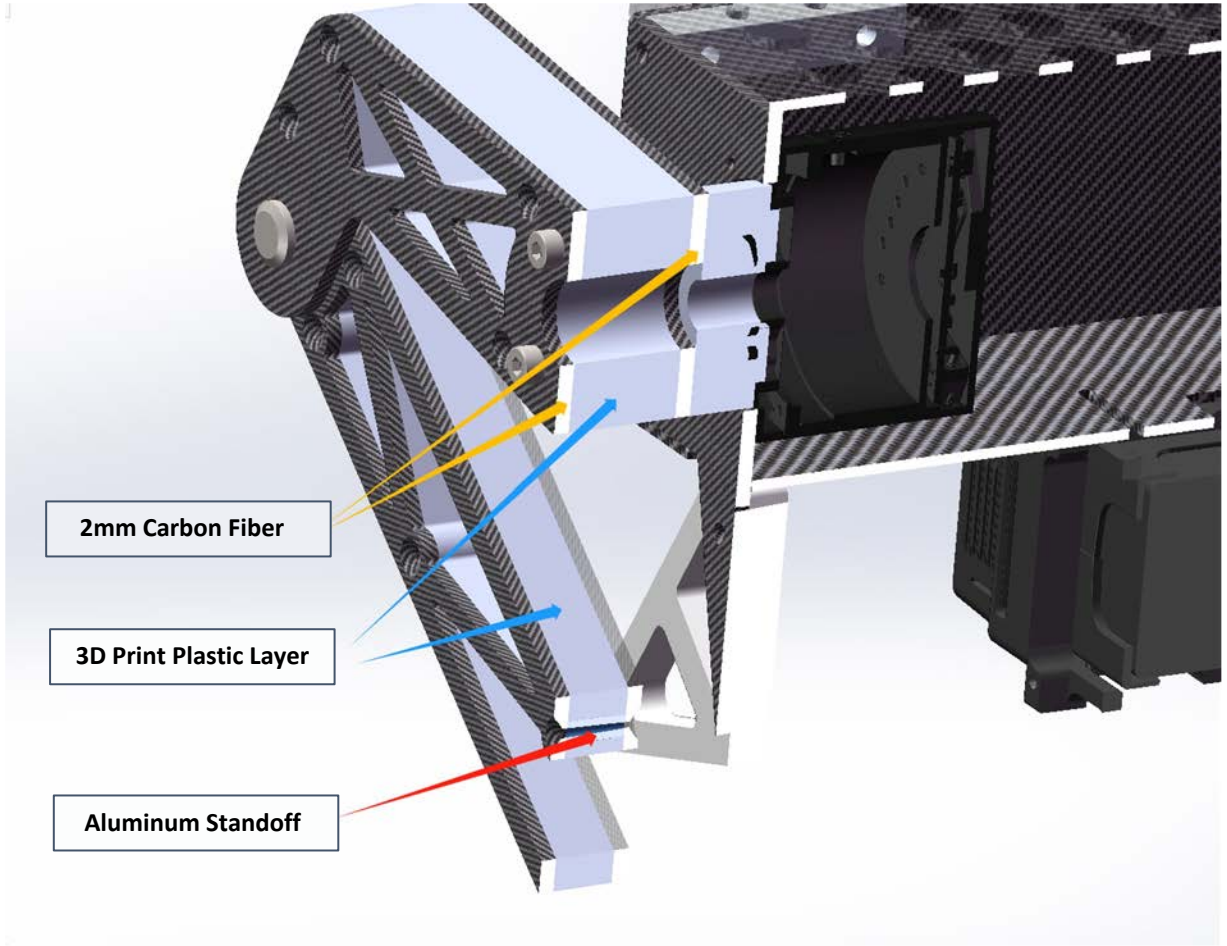
Challenge: Offers limited support for lateral forces, such as those experienced during side collisions on a leg component

Enhanced Version:

Design Upgrade: two layers of carbon fiber with an inner layer sandwiched by a low-density, 3D-printed material.

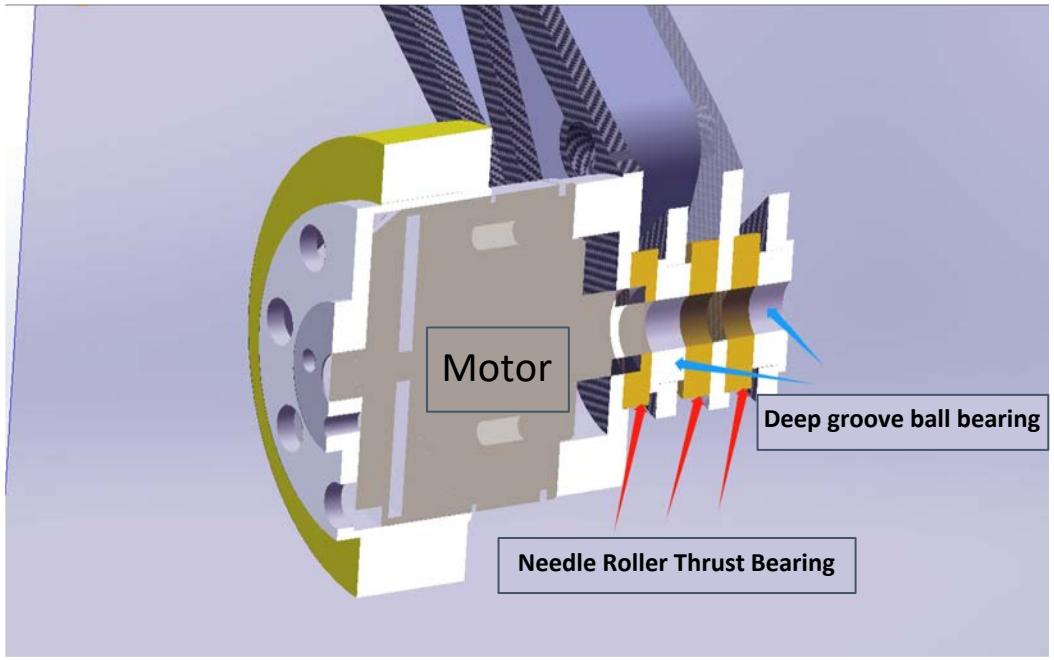
Performance Improvement: Significant increase in strength

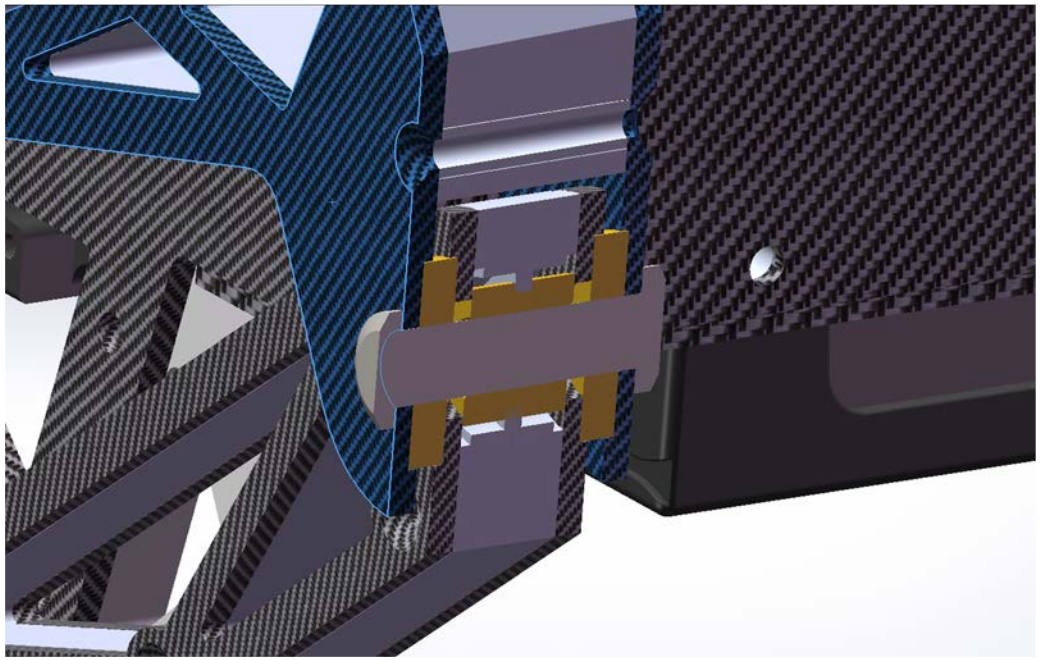
Detailed cross section



Cross section of leg structure

Design Iteration



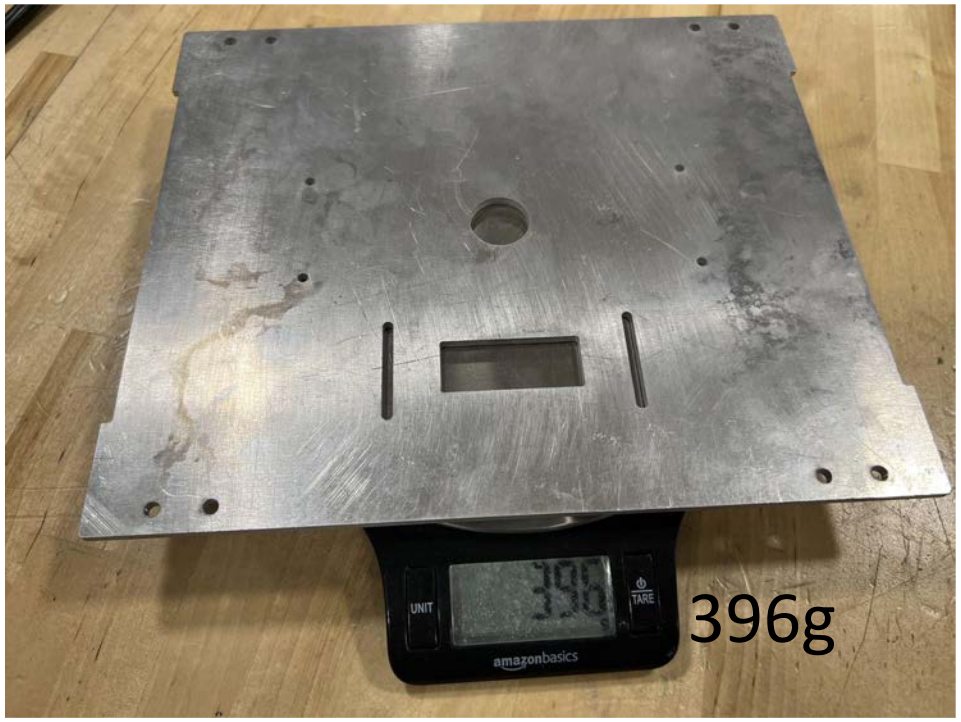


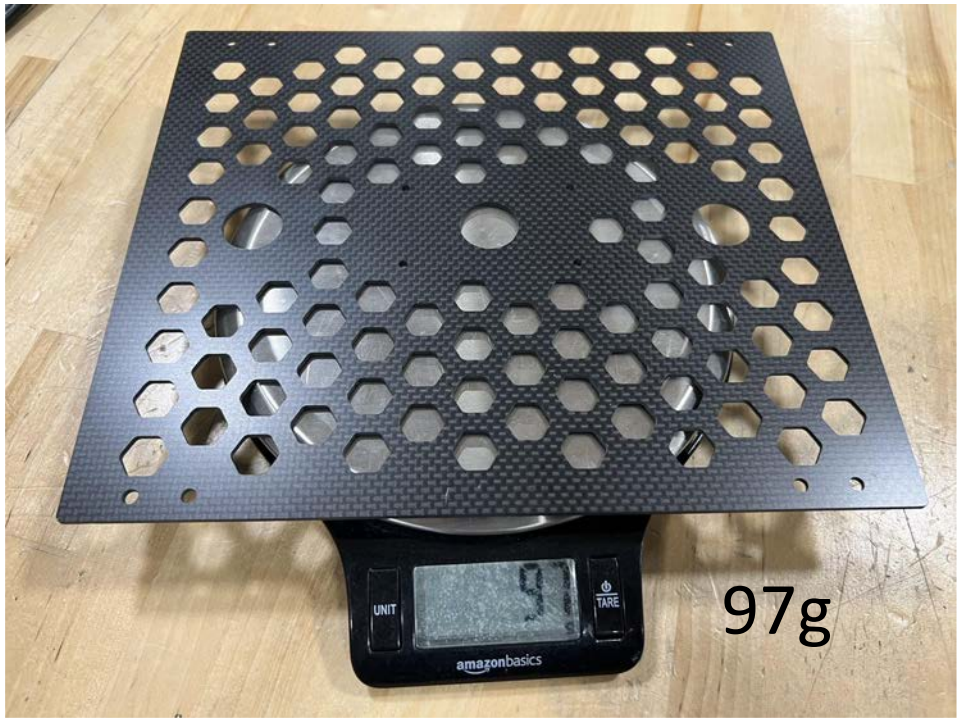
Design: Completely restricted shaft system.
Support Mechanism: Needle roller bearings combine with flat thrust bearings.
Performance Improvement: This new design considerably enhances bearing strength and reduces rotational resistance, addressing the issues identified in the initial design.

Design Iteration

Comparison before and after optimization

Design Iteration





Comparison of weight before and after optimization & change material

Electrical System

Power & Microcontroller Boards

Electrical System

- DC-DC Power Board
  - Hardware Components
  - Protection Circuit
- Microcontroller Board
  - Hardware Components
  - Design Considerations

Electrical System

- DC-DC Power Board
  - Hardware Components
  - Protection Circuit
- Microcontroller Board
  - Hardware Components
  - Design Considerations

DC-DC Power Board

Core Components:

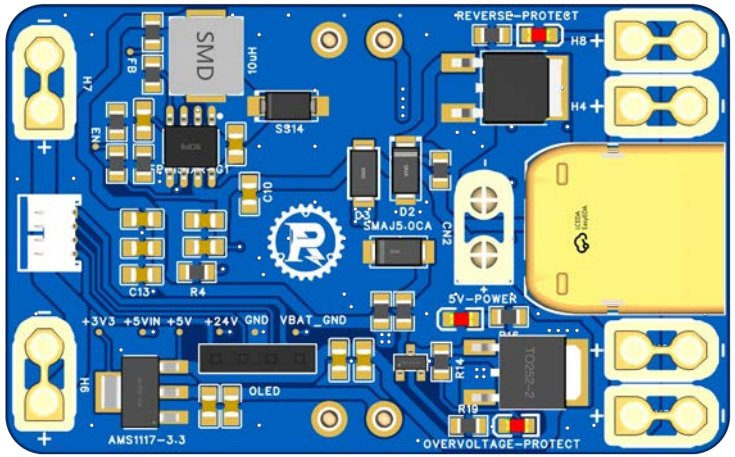
FP6151 Buck Converter with up to 90% efficiency.
AMS1117 Voltage Regulator: Voltage level regulation.
Overvoltage and Reverse Polarity Protection Circuit: Safety and durability enhancement.

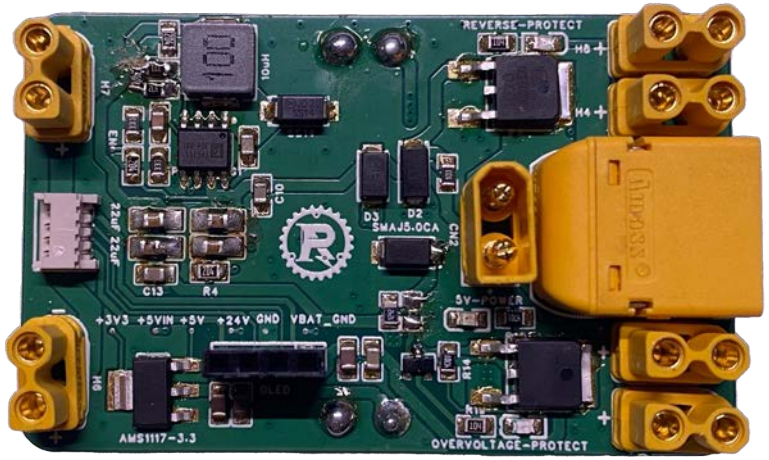
Functional Specifications:

Input Voltage Range: 10-36V.
Output Voltage and Current: 5V up to 5A, and 3.3V up to 1A.
Protection: Input reverse polarity protection, output overvoltage protection

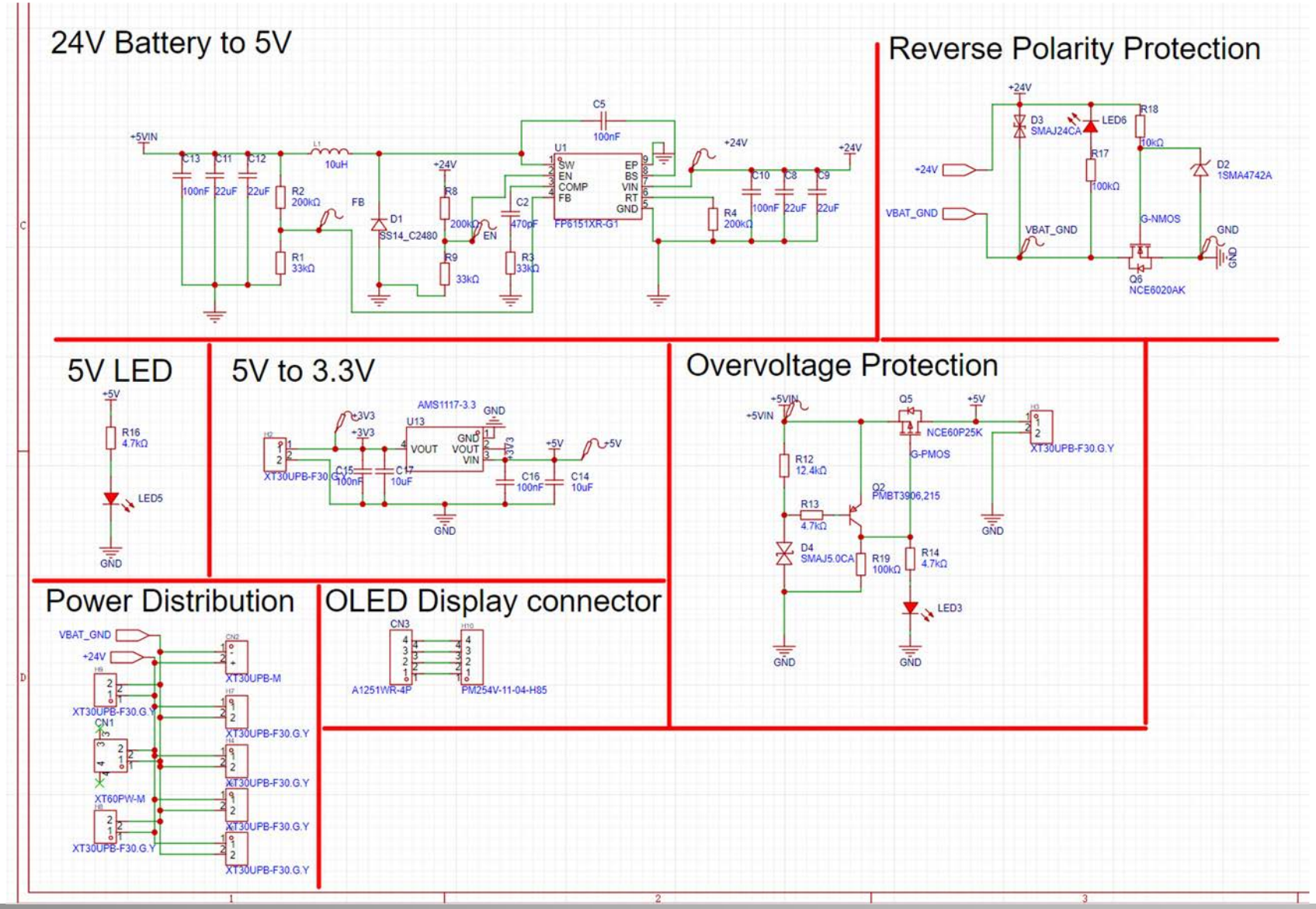
Additional Features:

Test Points: For monitoring all voltage levels.
Status Indicators: 3 LEDs to indicate circuit status.
Motor Power distribution(6*XT30)
OLED screen connector

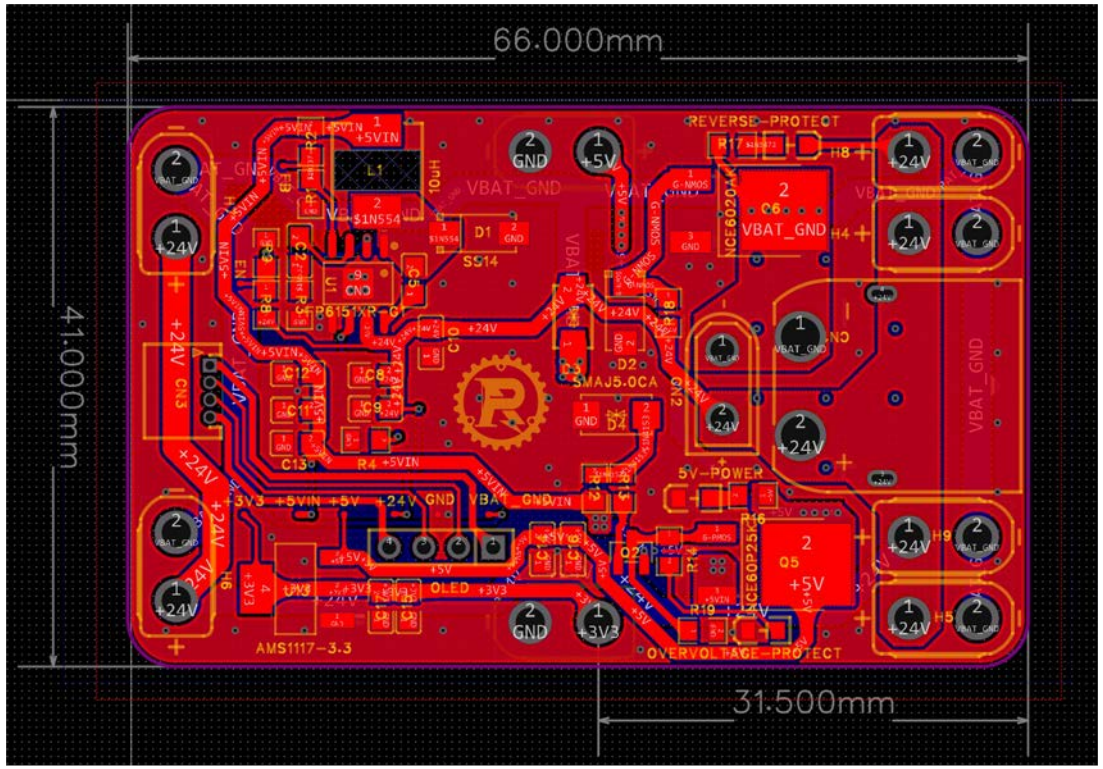


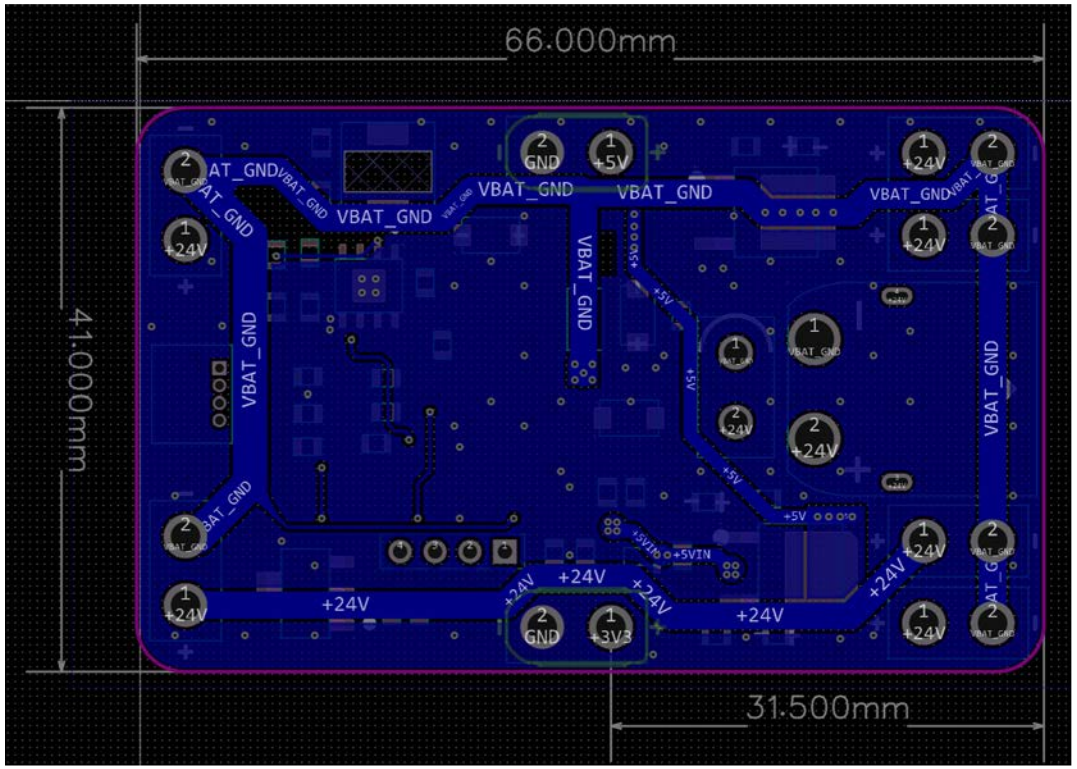


DC-DC Power Board Schematic



DC-DC Power Board Layout





Electrical System

- DC-DC Power Board
  - Hardware Components
  - Protection Circuit
- Microcontroller Board
  - Hardware Components
  - Design Considerations

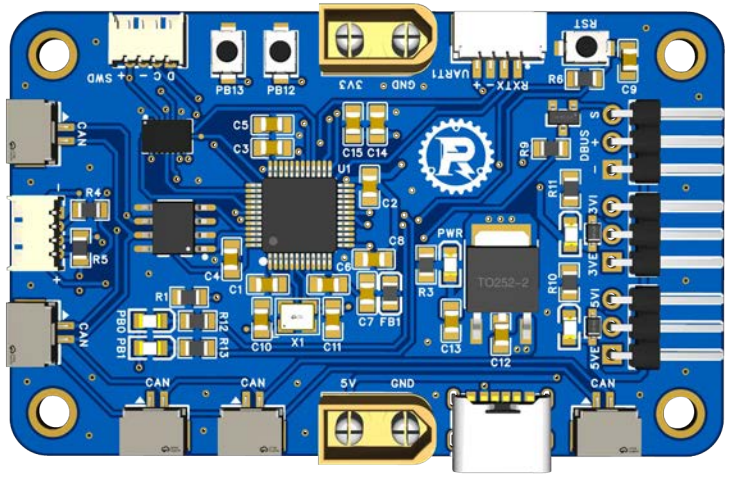
Hardware Components

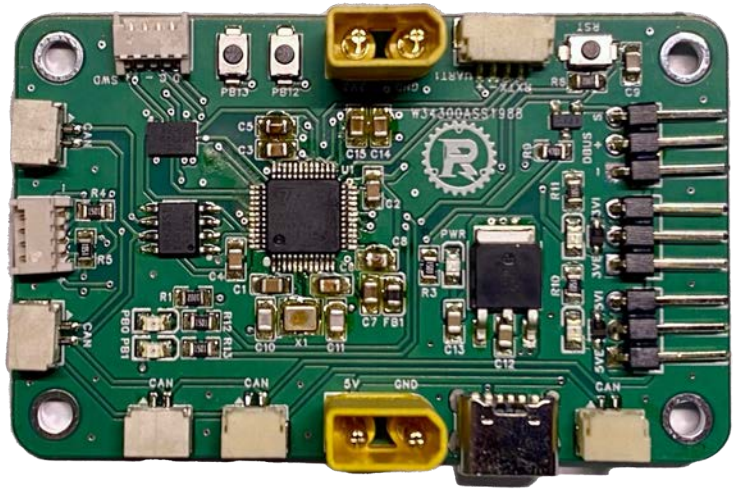
Main Hardware Components

- STM32F103C8T6 microcontroller
- AMS1117 voltage regulator
- MAX3051 CAN Transceiver
- BMI088 IMU

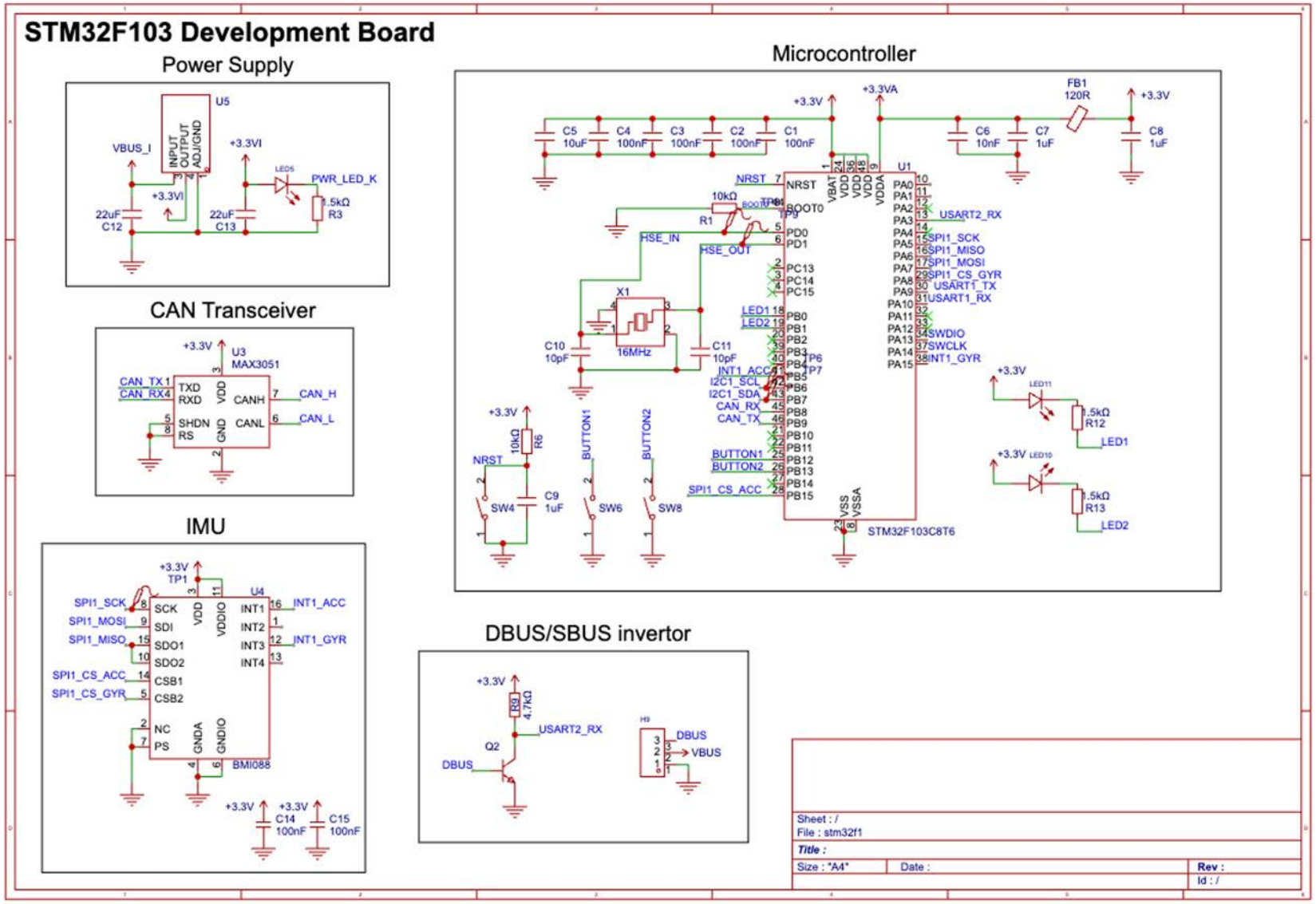
Connectors

- 2 XT30 connectors for power input
- 1 USB-C connector for power input
- 5 CAN connectors
- 1 DBUS port
- SWD for serial wire debugging
- UART for printing messages through serial port

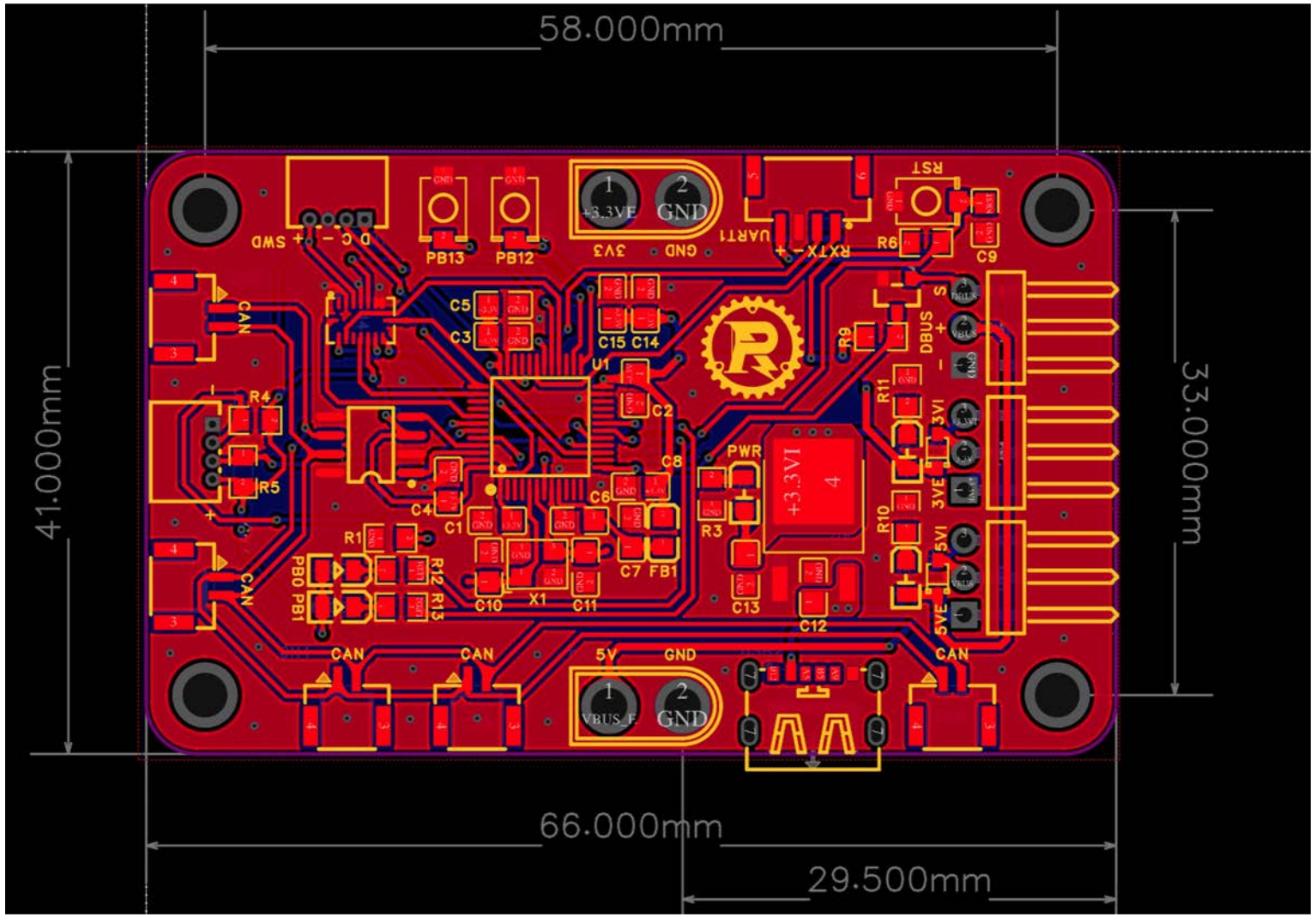




Microcontroller Board Schematic



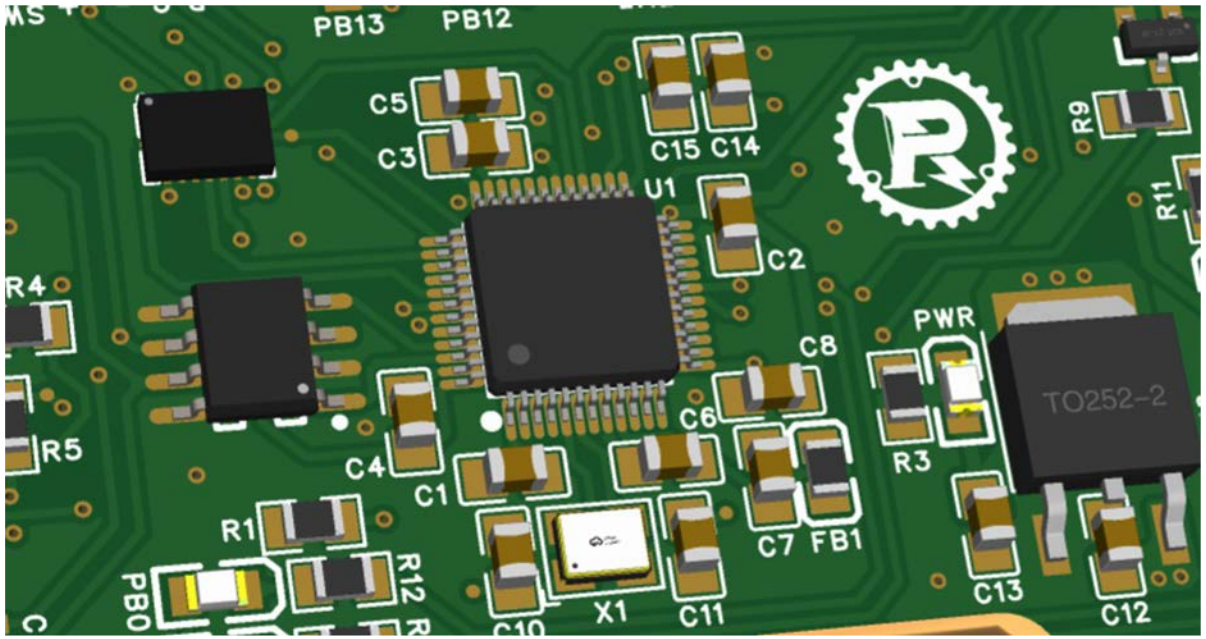
Microcontroller Board Layout



Design Considerations

Component Placement

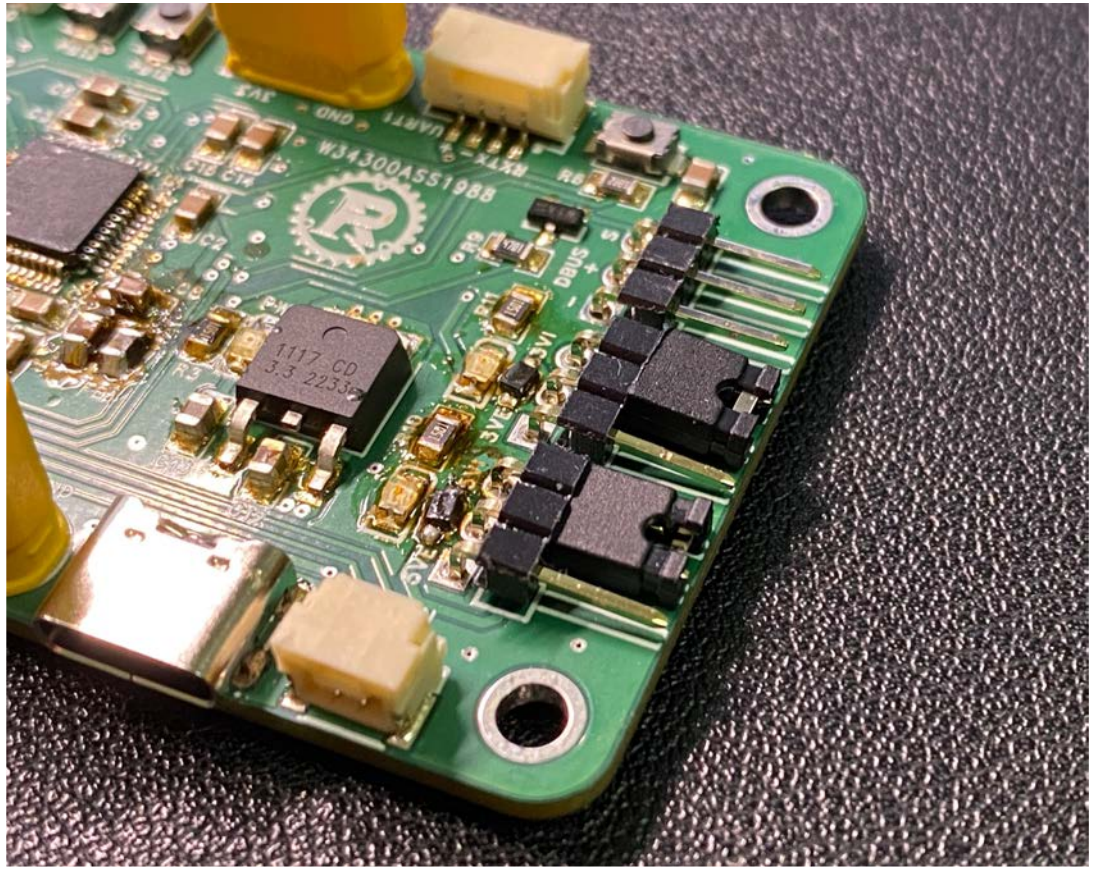
- Decoupling capacitors are placed closed to the microcontroller to minimize impedance and inductance
- The IMU is placed further away from the power circuit to minimize the heat generated by the linear regulator.



Design Considerations

Power Sources

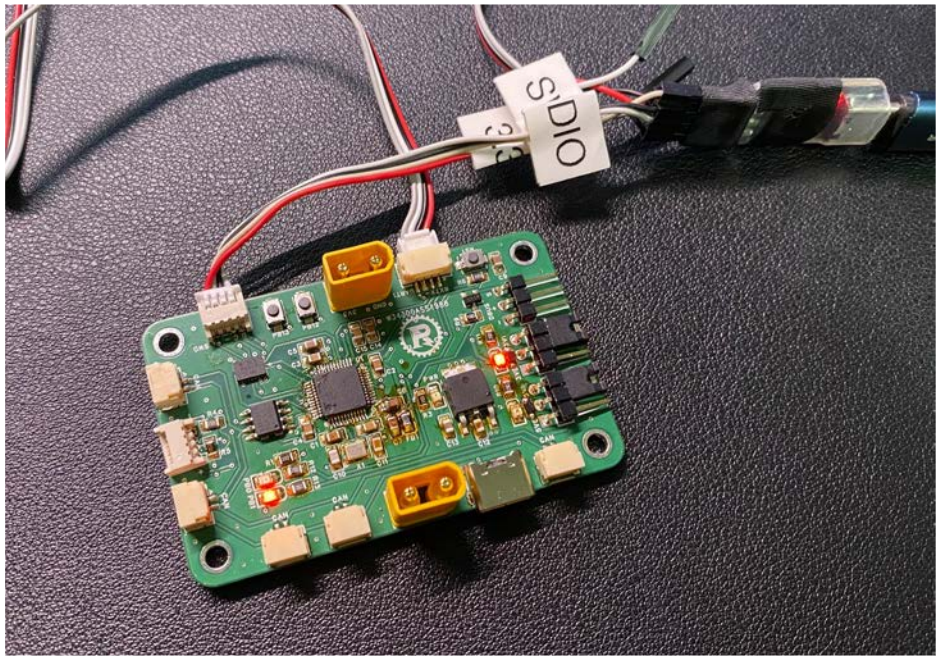
- The board can be powered by several different power sources.
- When testing the board on its own, we can power it with the SWD port or the USB-C port.
- When mounted on the robot, the board can be powered through the two XT30 connectors. The power source can be easily selected using jumper hats.



Design Considerations

Debug Purposes

- The board contains two LEDs and two buttons, which can be configured for debugging purposes.
- In the robot program, an LED is configured as a breathing LED to indicate the status of the board.



Embedded Software

Low-level Software, Peripherals, and Sensor Fusion

Embedded Software

- Low-level Software
  - Hardware Abstraction Layer (HAL)
  - Real-time Operating System (RTOS)
- Peripherals
  - CAN
  - SPI
  - UART
- Sensor Fusion
  - The Mahony Filter

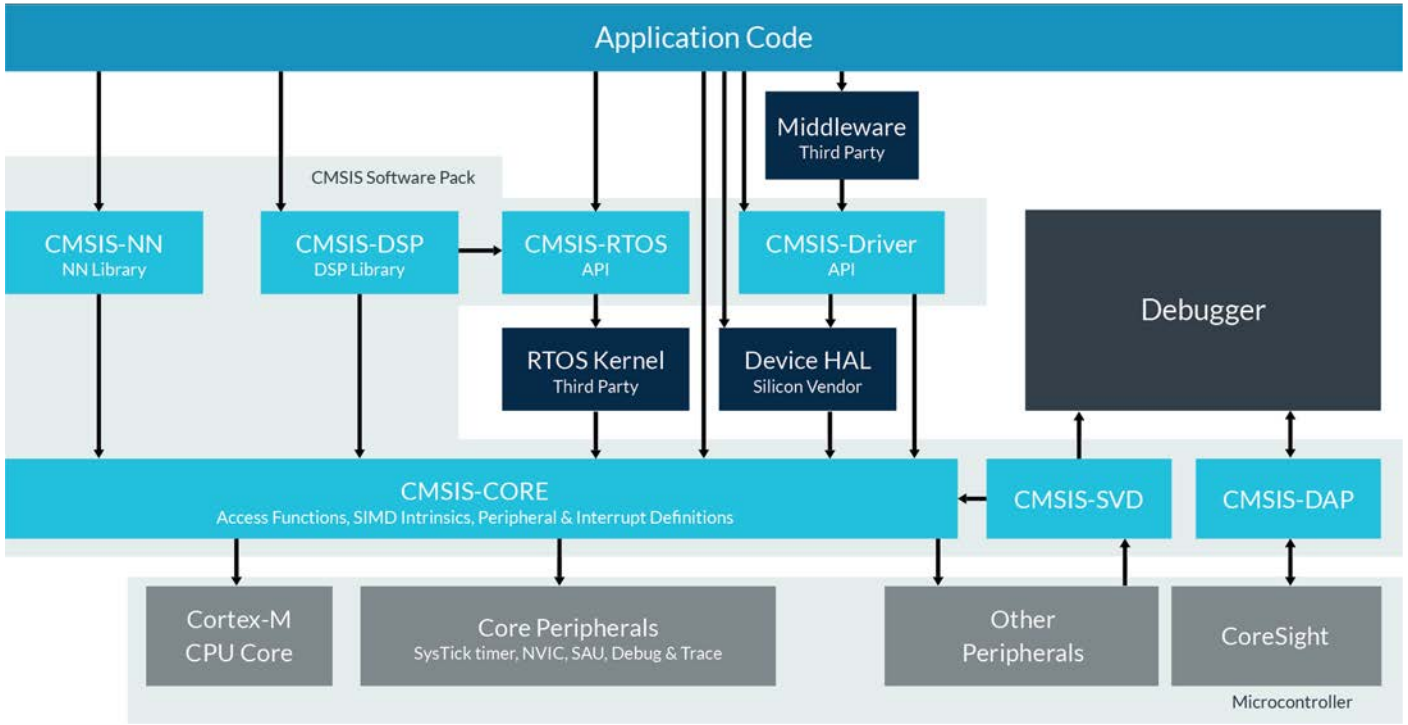
Embedded Software

- Low-level Software
  - Hardware Abstraction Layer (HAL)
  - Real-time Operating System (RTOS)
- Peripherals
  - CAN
  - SPI
  - UART
- Sensor Fusion
  - The Mahony Filter

Low-level Software

Hardware Abstraction Layer (HAL)

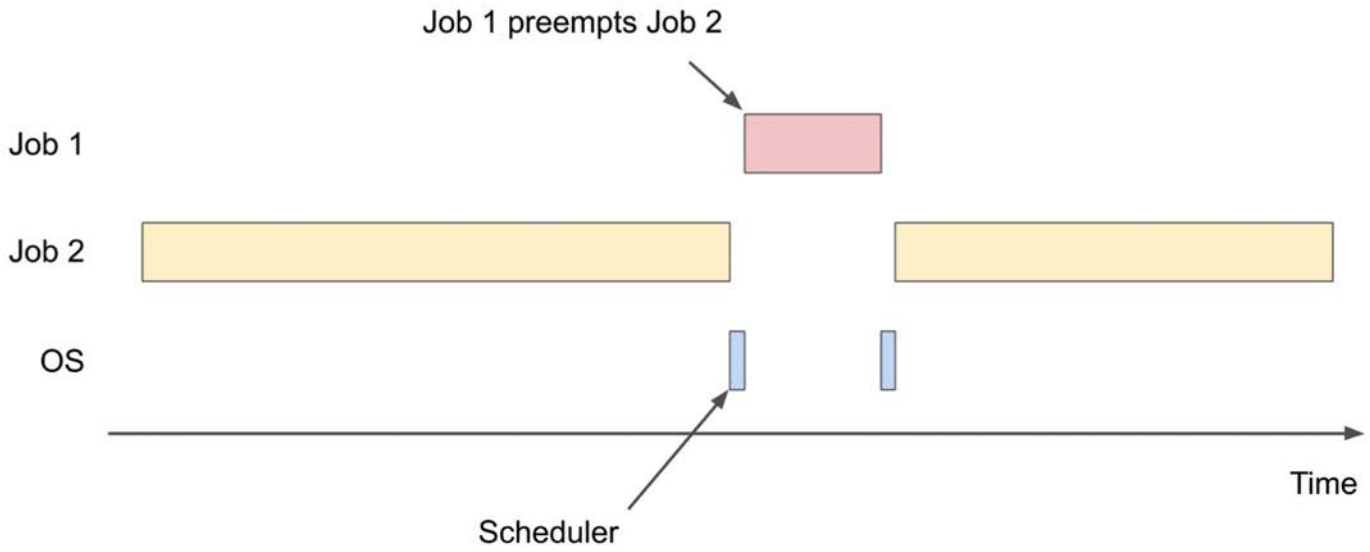
The Hardware Abstraction Layer (HAL) provides a high-level API that allows easier access to the hardware layer. In our case, the HAL is provided by the chip manufacturer, STMicroelectronics.



Low-level Software

Real-time Operating System (RTOS)

The RTOS allows us to create multiple threads on a single core, where its scheduler is responsible for switching tasks, scheduling, etc. The RTOS kernel we are using is **FreeRTOS**, a popular open source RTOS kernel. However, we are not directly using the APIs from FreeRTOS. Instead, we are using the **CMSIS-RTOS** library developed by ARM, which acts as an abstraction layer to FreeRTOS. We can assign priority to each task, where the task with higher priority can preempt the ones with lower priority.



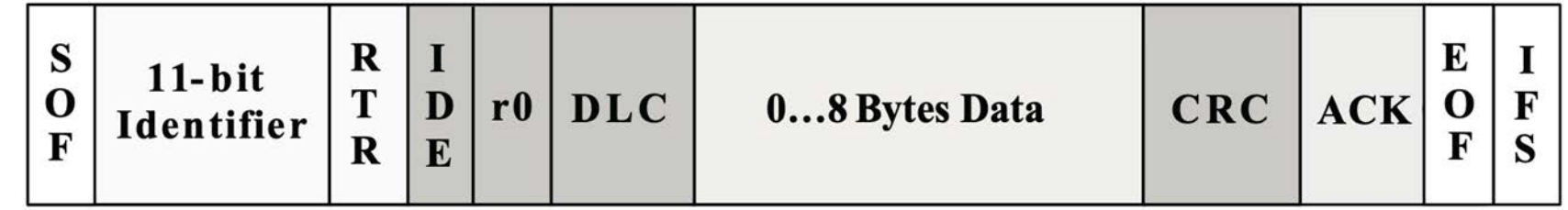
Embedded Software

- Low-level Software
  - Hardware Abstraction Layer (HAL)
  - Real-time Operating System (RTOS)
- Peripherals
  - CAN
  - SPI
  - UART
- Sensor Fusion
  - The Mahony Filter

Peripherals

CAN (Controller Area Network)

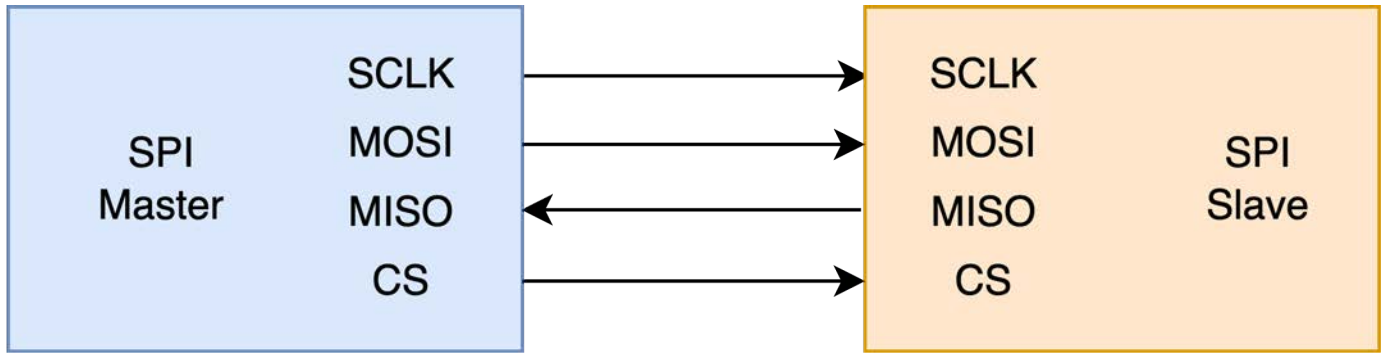
- Protocol for communicating with all of our motors.
- Allows each device to communicate with each other without the need of a central hub.
- When the microcontroller transmits a message, all devices receives the same message but only responds if the identifier attached to the message matches its own.



Peripherals

SPI (Serial Peripheral Interface)

- Protocol for communicating with the IMU
- A high speed full-duplex communication protocol, which is important for the IMU as we need real-time data for pose estimation.



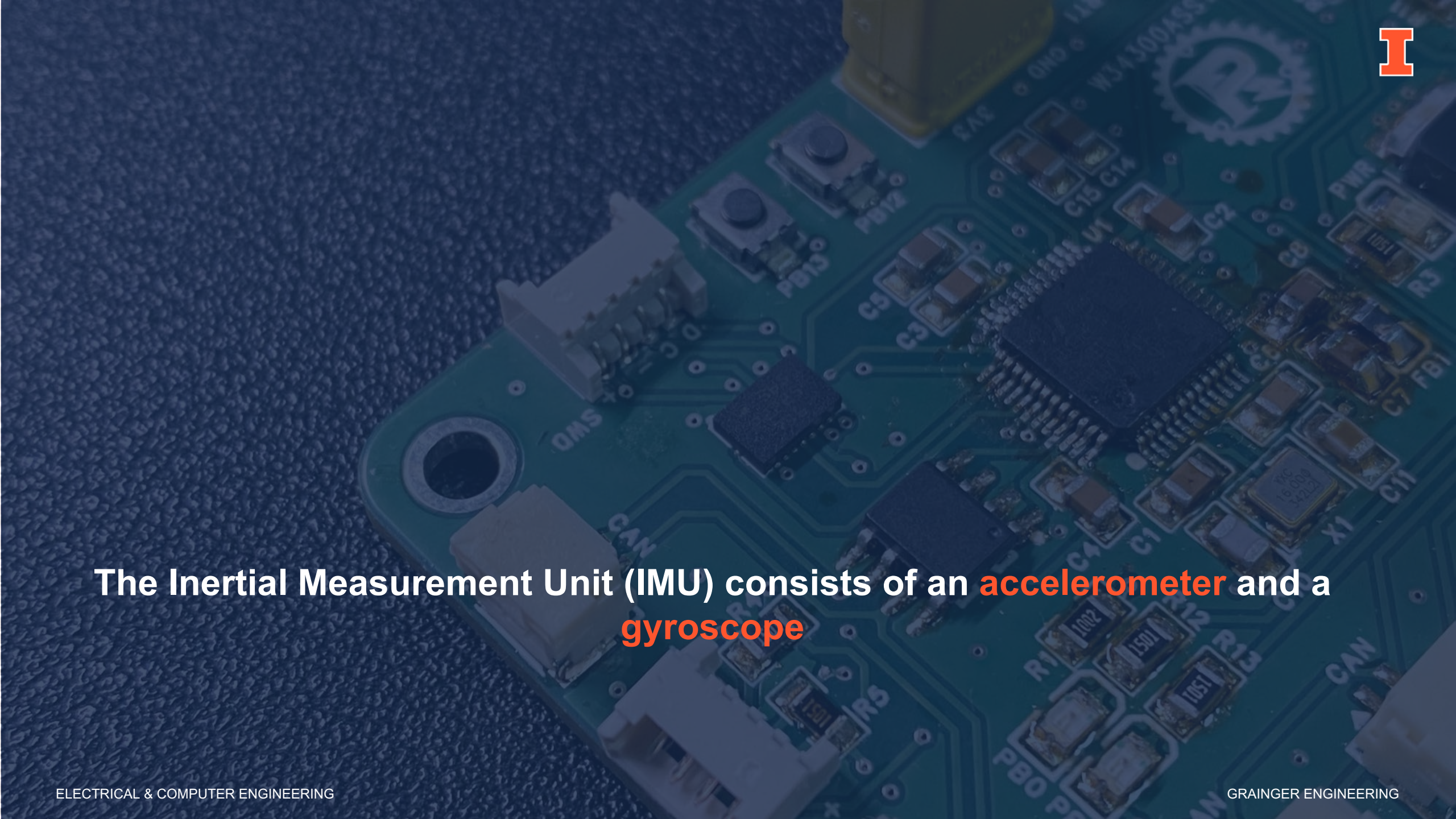
Peripherals

UART (Universal Asynchronous Receiver/Transmitter)

- Two purposes: transmission from DBUS receiver and debugging.
- The remote control signals received by the receiver are transmitted to the microcontroller through a DBUS protocol, which is essentially an inverted UART signal.
- For debugging, we use UART to transmit messages to our screen through a serial port.

Embedded Software

- Low-level Software
  - Hardware Abstraction Layer (HAL)
  - Real-time Operating System (RTOS)
- Peripherals
  - CAN
  - SPI
  - UART
- **Sensor Fusion**
  - **The Mahony Filter**

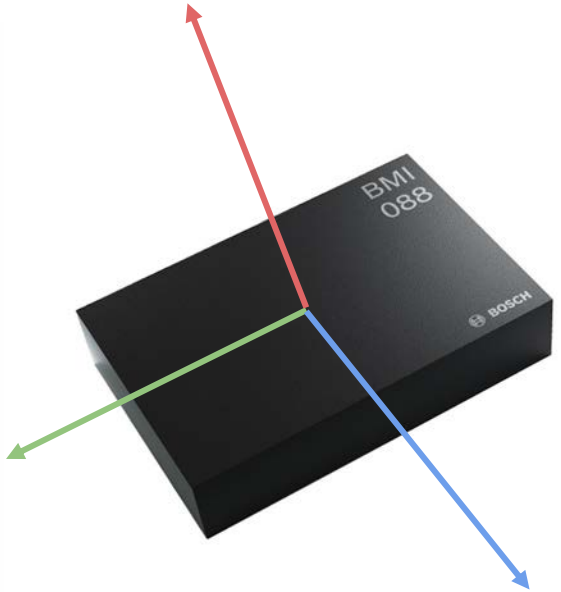


The Inertial Measurement Unit (IMU) consists of an accelerometer and a gyroscope

Sensor Fusion

Accelerometer

- The accelerometer measures **linear acceleration** in each axis.
- It measures the force of acceleration caused by gravity or by movement.
- When at rest, an accelerometer will measure the acceleration due to gravity.

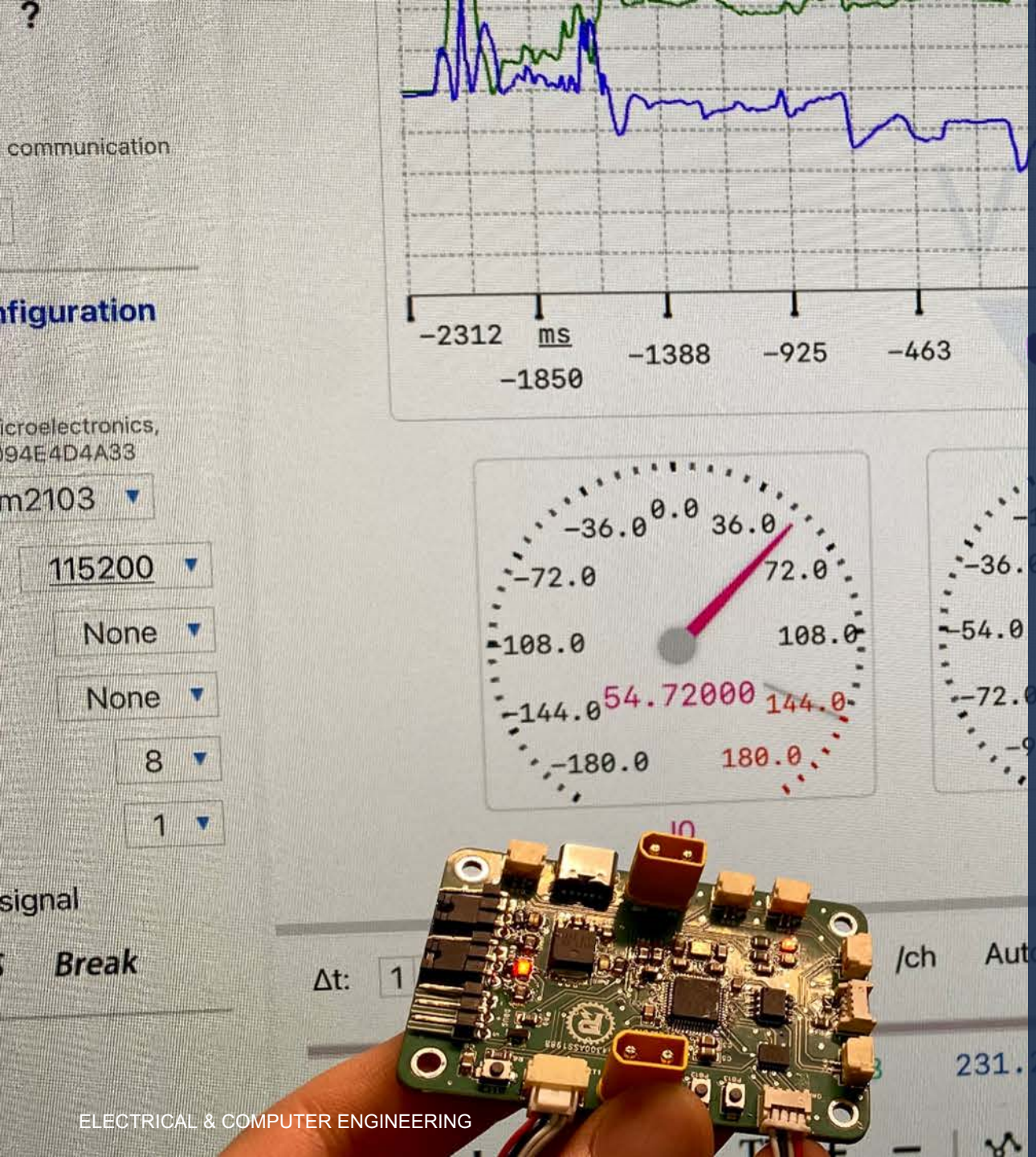


Gyroscope

- The gyroscope measures **angular rate** around each axis.
- The sensor measure the rate of a rotation by detecting a deviation from the initial orientation.

Why is sensor fusion necessary?

- Gyroscopes are prone to **drift** due to error accumulated over time.
- Accelerometer measurements contain **high frequency noise**
- We can combine the high-frequency, short-term accuracy of the gyroscope with the low-frequency, long-term stability of the accelerometer.



Sensor Fusion

The Mahony Filter

Step 1: Obtain sensor measurements

$^{I}\omega_t$: gyroscope measurements

$^{I}\mathbf{a}_t$: accelerometer measurements

$^{I}\hat{\mathbf{a}}_t$: normalized accelerometer measurements

Step 2: Orientation error using accelerometer and estimated gyroscope measurements

$$\mathbf{v}\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t}\right) = \begin{bmatrix} 2(q_2 q_4 - q_1 q_3) \\ 2(q_1 q_2 + q_3 q_4) \\ \left(q_1^2 - q_2^2 - q_3^2 + q_4^2\right) \end{bmatrix}$$

$$\mathbf{e}_{t+1} = {}^{I}\hat{\mathbf{a}}_{t+1} \times \mathbf{v}\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t}\right)$$

$$\mathbf{e}_{i,t+1} = \mathbf{e}_{i,t} + \mathbf{e}_{t+1}\Delta t$$

R. Mahony, 2008

Sensor Fusion

The Mahony Filter

Step 3: Update gyro measurements using PI controller (Fusion)

$$ {}^{I}\omega_{t+1} = {}^{I}\omega_{t+1} + \mathbf{K}_p\mathbf{e}_{t+1} + \mathbf{K}_i\mathbf{e}_{i,t+1} $$

Step 4: Calculate change in orientation based on the updated gyro measurements

$$ {}^{I}_{W}\dot{\mathbf{q}}_{\omega,t+1} = \frac{1}{2}{}^{I}_{W}\hat{\mathbf{q}}_{est,t} \otimes \left[0, {}^{I}\omega_{t+1}\right]^T $$

$\otimes$: quaternion multiplication

Step 5: Update orientation estimate with numerical integration

$$ {}^{I}_{W}\mathbf{q}_{est,t+1} = {}^{I}_{W}\hat{\mathbf{q}}_{est,t} + {}^{I}_{W}\dot{\mathbf{q}}_{\omega,t+1}\Delta t $$

R. Mahony, 2008

Sensor Fusion

The Mahony Filter

Integration of Gyroscope Data: The filter first integrates the angular rate measurements from the gyroscope over time to estimate the orientation. This orientation is subject to drift due to accumulating errors in the gyro data.

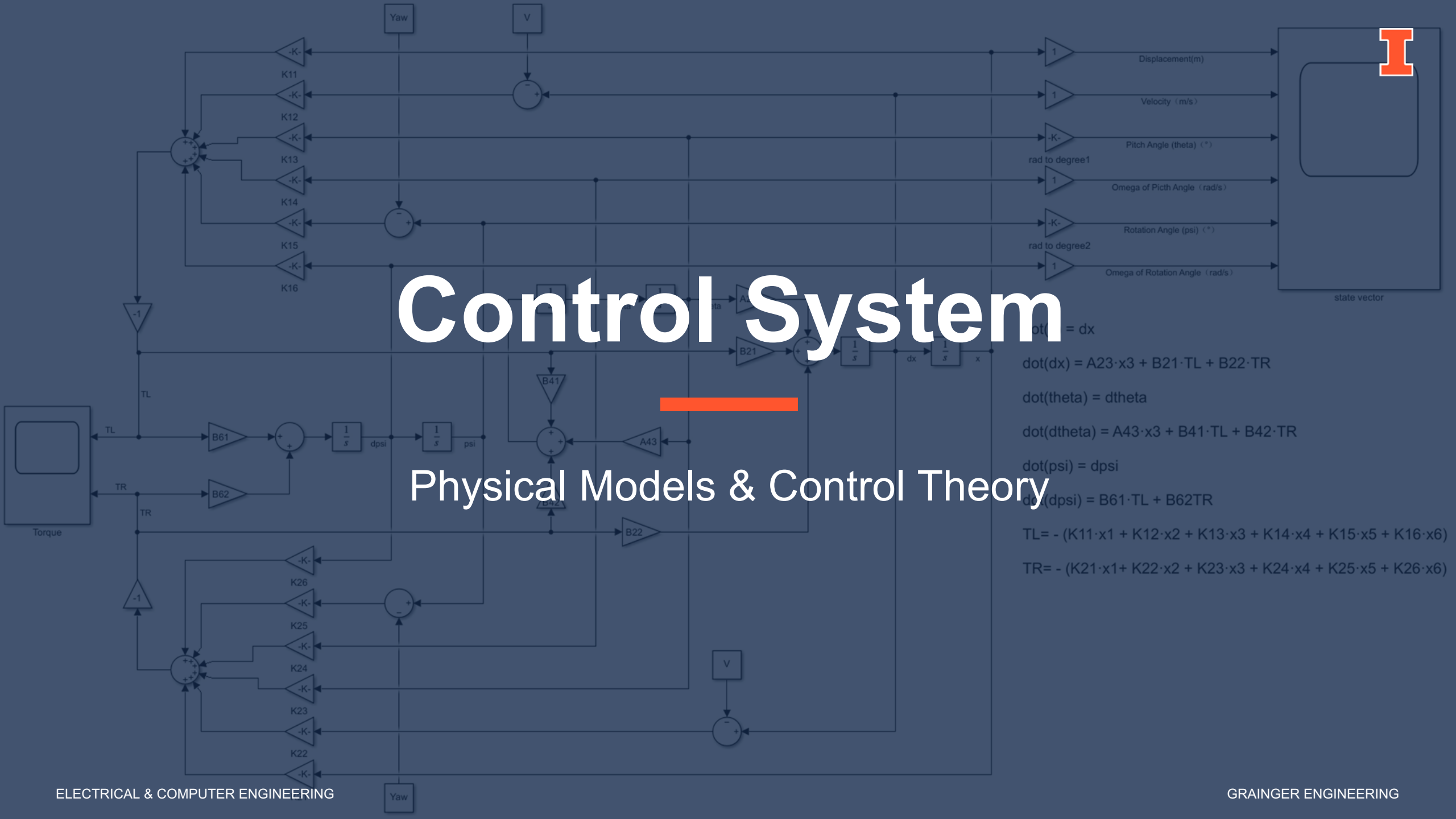
Accelerometer Correction: The filter then uses the accelerometer data to obtain an estimate of the gravitational vector in the device's frame of reference. This estimate is used to correct the drift in the gyroscope-based orientation estimate.

Error Estimation: The Mahony filter calculates the error between the estimated gravity direction (from the integrated gyroscope data) and the measured gravity direction (from the accelerometer).

Feedback Loop: This error is then fed back into the system to adjust the gyroscope integration in the next iteration, effectively reducing the drift.

Tuning: The filter includes parameters that can be tuned to balance the responsiveness of the filter to gyroscope and accelerometer data. These parameters control how quickly the filter corrects the gyroscope drift based on the accelerometer data.

R. Mahony, 2008



Control System

Physical Models & Control Theory

dot( ) = dx

dot(dx) = A23·x3 + B21·TL + B22·TR

dot(theta) = dtheta

dot(dtheta) = A43·x3 + B41·TL + B42·TR

dot(psi) = dpsi

dot(dpsi) = B61·TL + B62TR

TL= - (K11·x1 + K12·x2 + K13·x3 + K14·x4 + K15·x5 + K16·x6)

TR= - (K21·x1+ K22·x2 + K23·x3 + K24·x4 + K25·x5 + K26·x6)

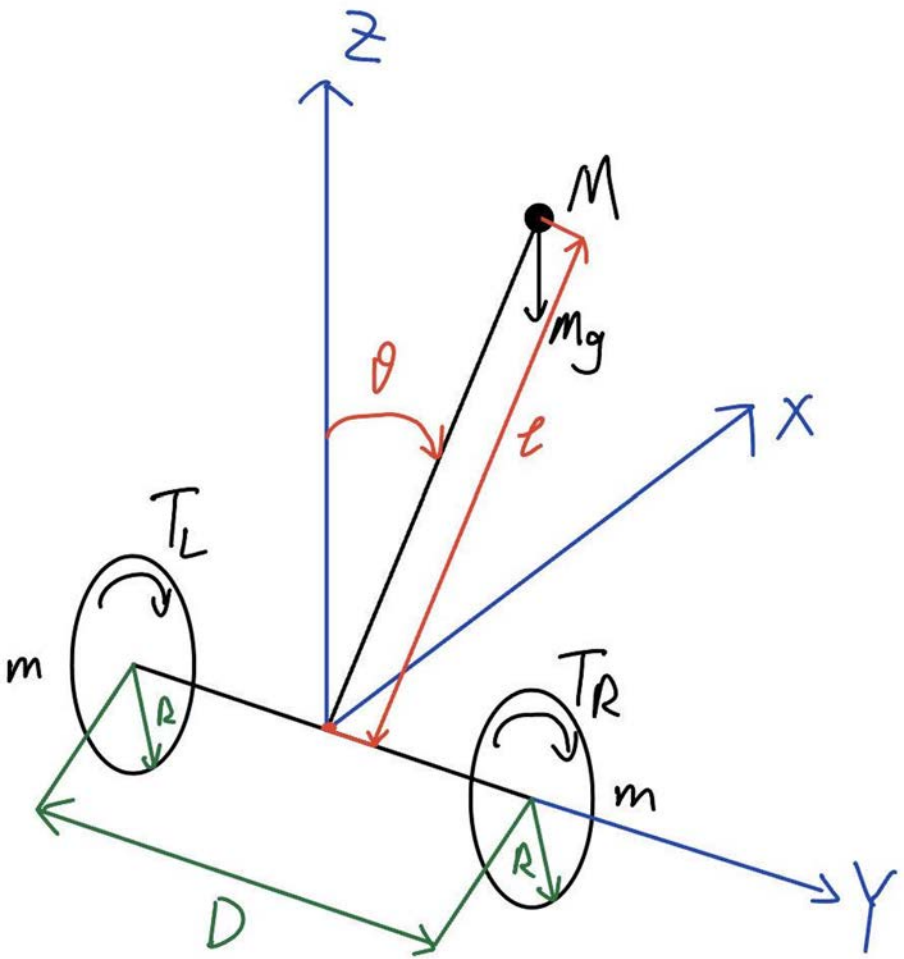
Control System

- Physical Models
  - Wheel Motion Model
  - Leg Motion Model
- Control Theory
  - Linear Quadratic Regulator(LQR) Algorithm
  - Virtual Model Control(VMC) Algorithm

Control System

- Physical Models
  - Wheel Motion Model
  - Leg Motion Model
- Control Theory
  - Linear Quadratic Regulator(LQR) Algorithm
  - Virtual Model Control(VMC) Algorithm

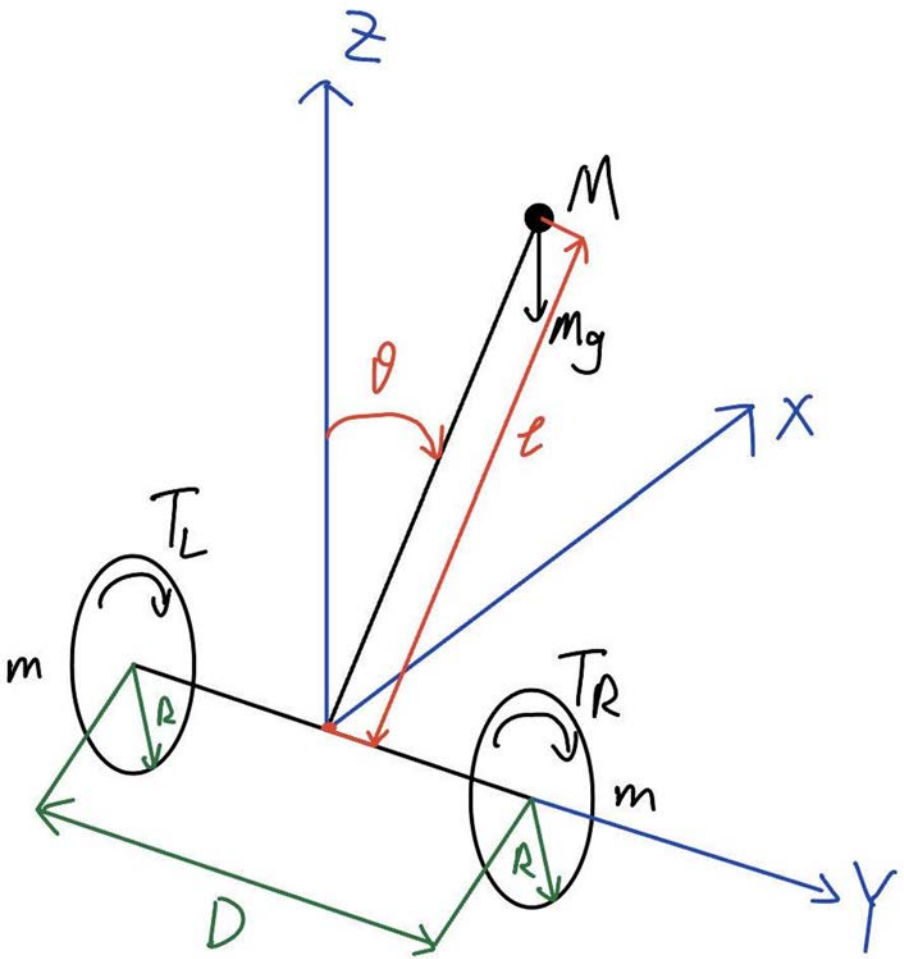
Wheel Motion Model



Variable Declaration

Label	Meaning	Unit
x_L, x_R	The displacement of the left and right wheels.	m
z	The distance between the body's center of mass and wheel motor rotation axis along the z-axis.	m
ϕ	The roll angle of the body.	rad
θ	The pitch angle of the body.	rad
ψ	The yaw angle of the body.	rad
T_L, T_R	The output torque of the left and right wheel motors.	$N \cdot m$
T	The output torque of the leg motors.	$N \cdot m$
N_L, N_R	The horizontal component of the force between wheels and the body (along the x-axis).	N
P_L, P_R	The vertical component of the force between wheels and the body (along the y-axis).	N
F_L, F_R	The frictions of the wheels when moving.	N

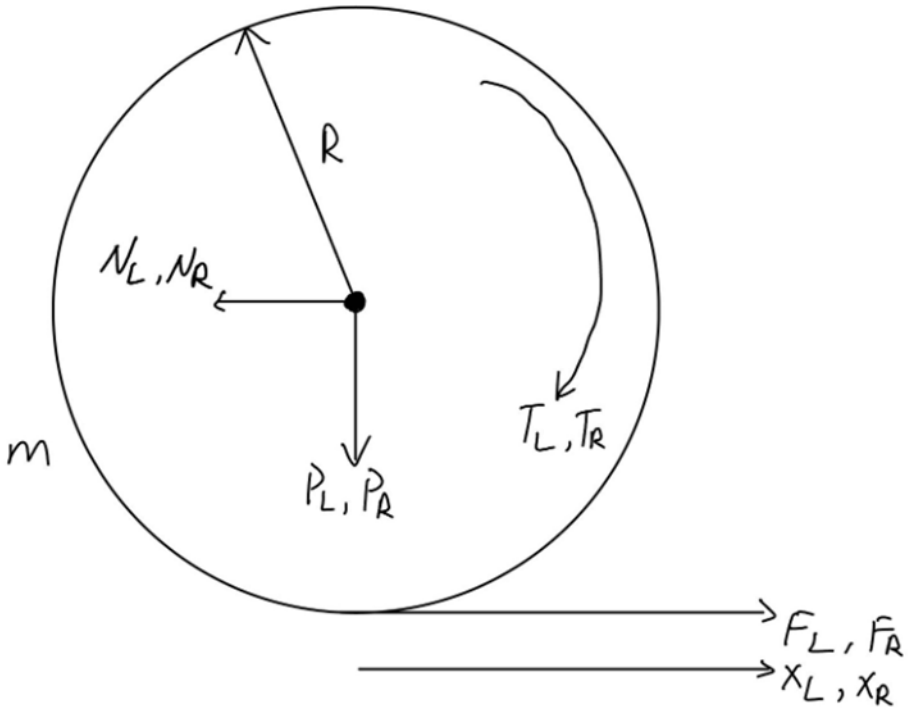
Wheel Motion Model



Parameter Declaration

Label	Meaning	Unit
m	The mass of the rotor in the wheel motors.	kg
M	The mass of the body.	kg
I_w	The moment of inertia of the rotor in the wheel motors.	$kg \cdot m^2$
I_x	The moment of inertia of the body rotated around the x-axis.	$kg \cdot m^2$
I_y	The moment of inertia of the body rotated around the y-axis.	$kg \cdot m^2$
I_z	The moment of inertia of the body rotated around the z-axis.	$kg \cdot m^2$
R	The radius of the wheel.	m
l	The distance between the body's center of mass and the rotation axis of the wheel motor.	m
D	The distance between the left and right wheels.	m
g	The acceleration due to the gravity measured.	m/s^2

Wheel Motion Model - Move Forward and Backward



For Wheels

Net force and torque analysis:

$$F_L - N_L = m\ddot{x}_L \tag{1}$$

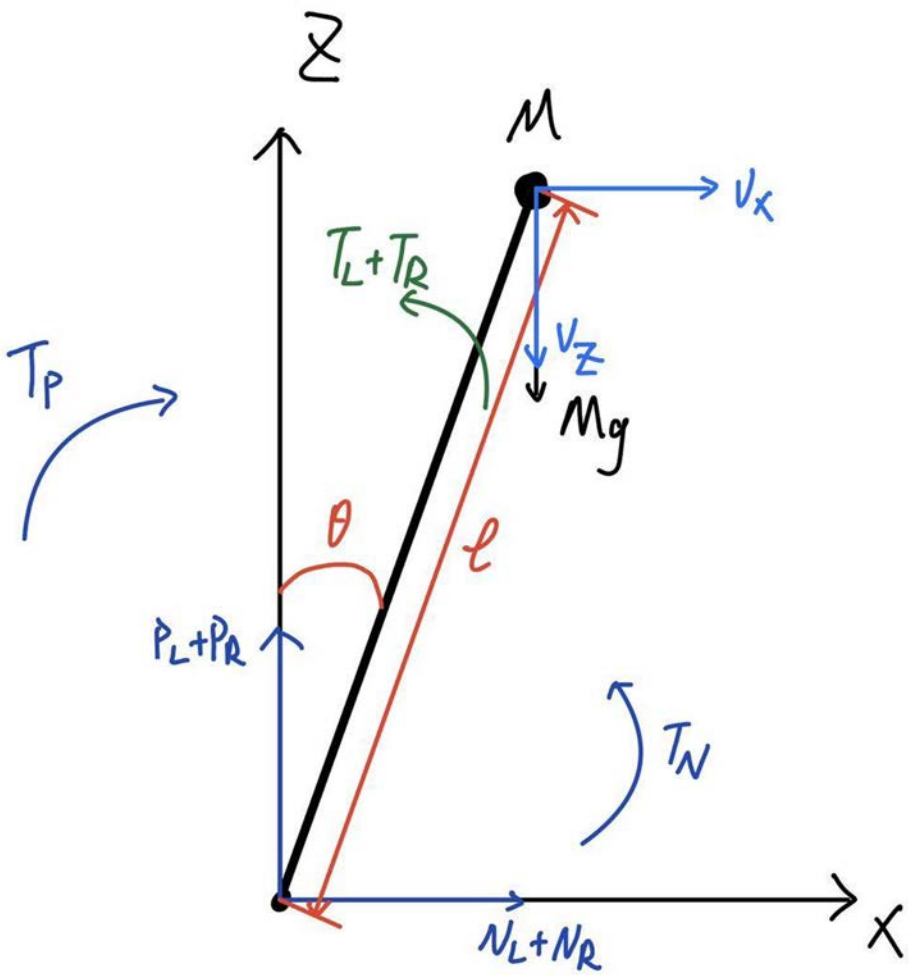
$$T_L - F_L * R = I_w \frac{\ddot{x}_L}{R} \tag{2}$$

Combine (1) and (2):

$$\ddot{x}_L = \frac{T_L - N_L R}{\frac{I_w}{R} + mR} \qquad \ddot{x}_R = \frac{T_R - N_R R}{\frac{I_w}{R} + mR} \tag{3}$$

$$\ddot{x} = \frac{\ddot{x}_L + \ddot{x}_R}{2} = \frac{T_L + T_R - (N_L + N_R)R}{2(\frac{I_w}{R} + mR)} \tag{4}$$

Wheel Motion Model - Self Balance (Pt.1)



Velocity decomposition:

$$v_x = \frac{\partial}{\partial t}(x + lsin(\theta)) = \dot{x} + l * cos(\theta)\dot{\theta} \quad (5)$$

$$v_z = \frac{\partial}{\partial t}(l - l * cos(\theta)) = l * sin(\theta) * \dot{\theta} \quad (6)$$

Net force and torque analysis:

$$N_L + N_R = M(\ddot{x} + l * cos(\theta)\ddot{\theta} - l * sin(\theta)\dot{\theta}^2) \quad (7)$$

$$P_L + P_R = Mg - M(l * sin(\theta)\ddot{\theta} + l * cos(\theta)\dot{\theta}^2) \quad (8)$$

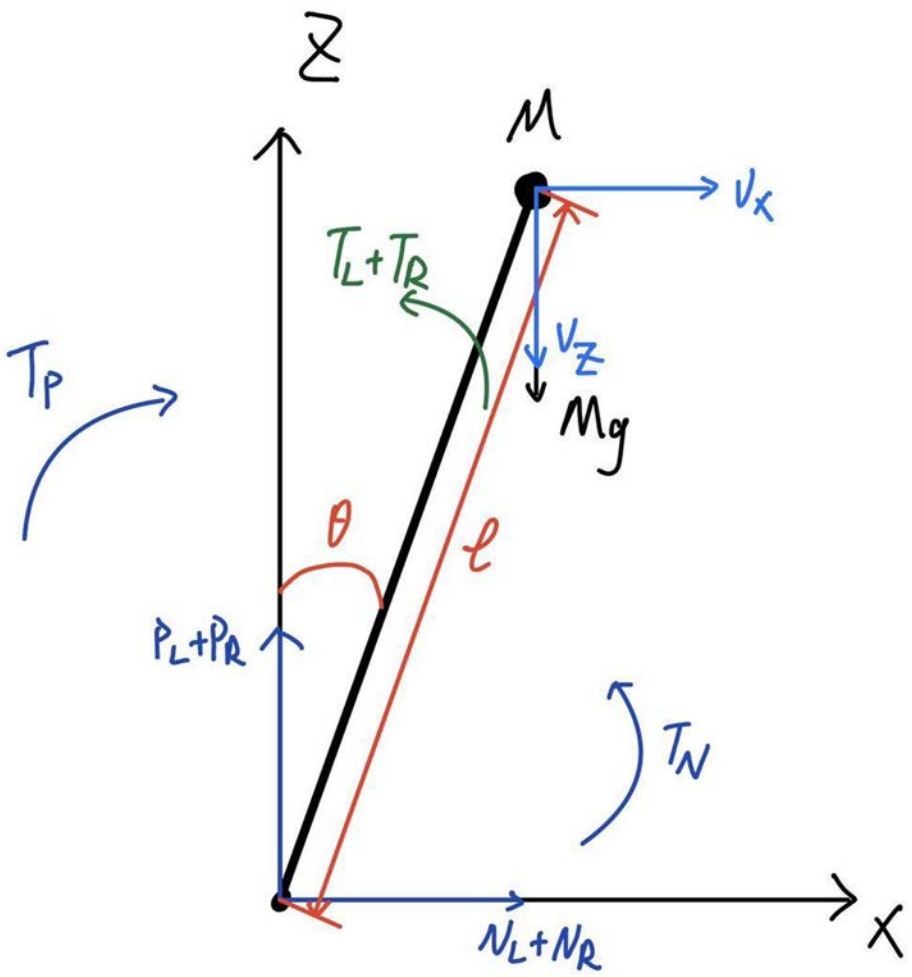
$$T_N = (N_L + N_R) * l * cos(\theta), \quad T_P = (P_L + P_R) * l * sin(\theta) \quad (9)$$

$$I_y\ddot{\theta} = T_P - T_N - (T_L + T_R) \quad (10)$$

Combine (7), (8),(9), (10):

$$(I_y + Ml^2)\ddot{\theta} = Mg * lsin(\theta) - M\ddot{x} * lcos(\theta) - (T_L + T_R) \quad (11)$$

Wheel Motion Model - Self Balance (Pt.2)



$$\ddot{x} = \frac{\ddot{x_L} + \ddot{x_R}}{2} = \frac{T_L + T_R - (N_L + N_R)R}{2(\frac{I_w}{R} + mR)} \quad (4)$$

$$N_L + N_R = M(\ddot{x} + l * cos(\theta)\ddot{\theta} - l * sin(\theta)\dot{\theta}^2) \quad (7)$$

Combine (4) and (7):

$$(\frac{2I_w}{R^2} + 2m + M)\ddot{x} = \frac{T_L + T_R}{R} - M * lcos(\theta)\ddot{\theta} + M * lsin(\theta)\dot{\theta}^2 \quad (12)$$

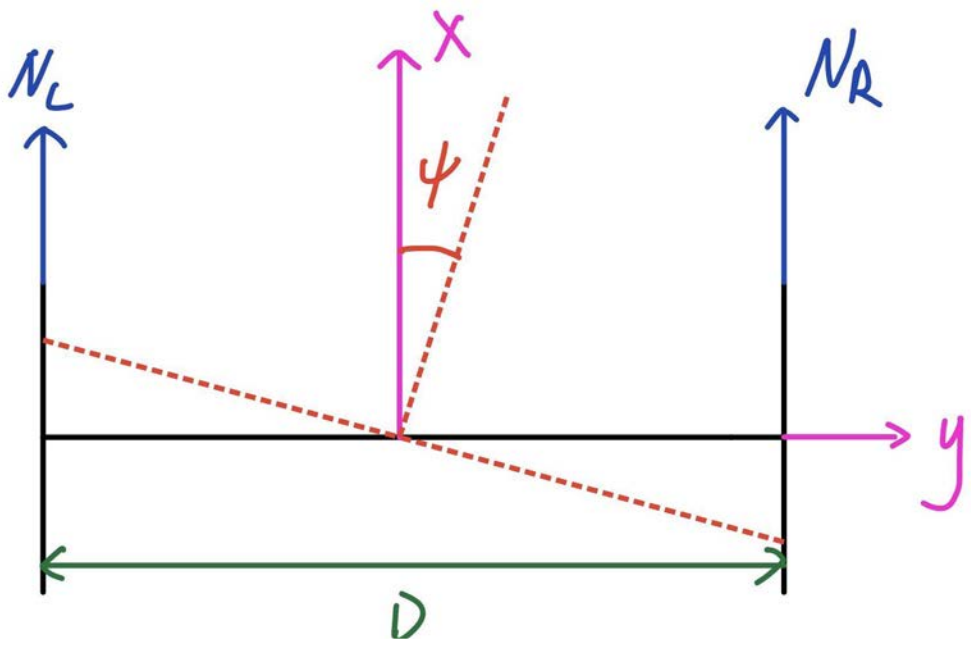
Because the robot can keep the balance at steady state by changing a tiny pitch angle(θ), we can linearize the parameters:

$$cos(\theta) = 1, \quad sin(\theta) = \theta, \quad \dot{\theta}^2 = 0$$

After simplification, we can get:

$$\begin{cases} (\frac{2I_w}{R^2} + 2m + M)\ddot{x} = \frac{T_L + T_R}{R} - Ml\ddot{\theta} \\ (I_y + Ml^2)\ddot{\theta} = Mgl\theta - M\ddot{x}l - (T_L + T_R) \end{cases} \quad (13)$$

Wheel Motion Model - Rotation



Net torque analysis:

$$I_z\ddot{\psi} = \frac{D}{2}(N_L - N_R) \tag{14}$$

The relationship between the yaw angular acceleration and left and right wheel acceleration is:

$$\ddot{\psi} = \frac{\ddot{x}_L - \ddot{x}_R}{2}$$

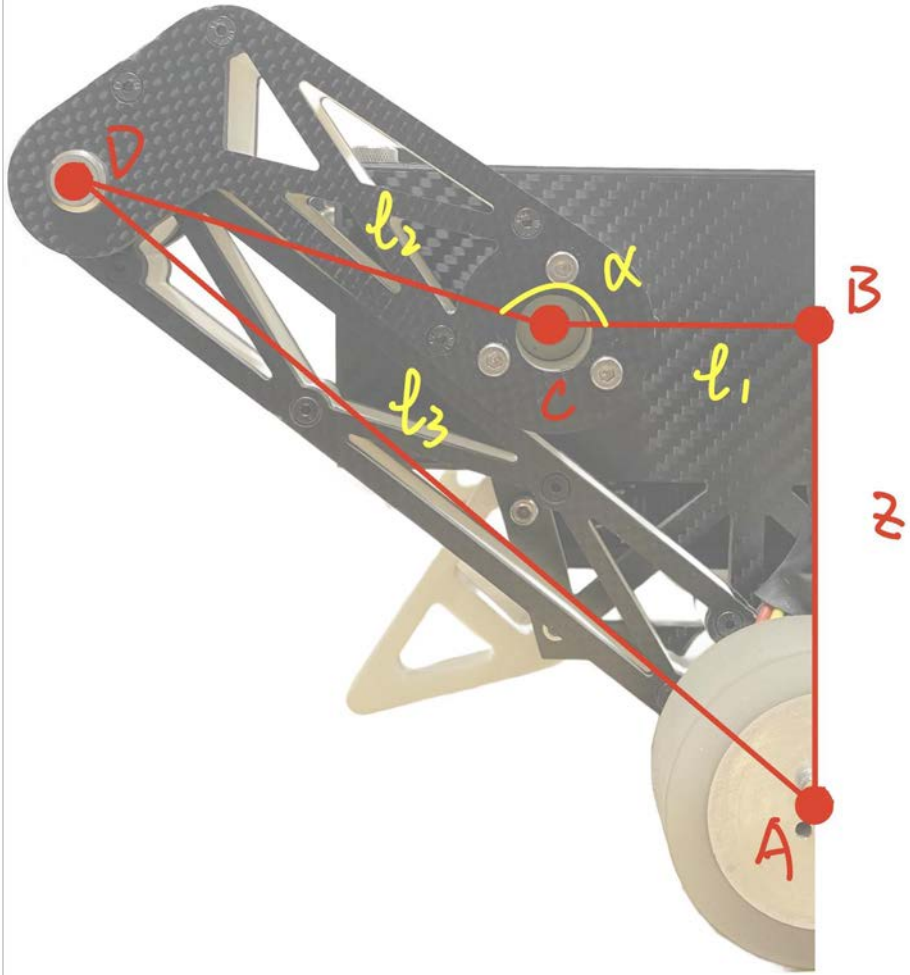
After simplification:

$$\ddot{\psi} = \frac{T_L - T_R}{R(\frac{2I_z}{D} + \frac{I_w D}{R^2} + mD)} \tag{15}$$

Control System

- Physical Model
  - Wheel Motion Model
  - Leg Motion Model
- Control Theory
  - Linear Quadratic Regulator(LQR) Algorithm
  - Virtual Model Control(VMC) Algorithm

Leg Motion Model

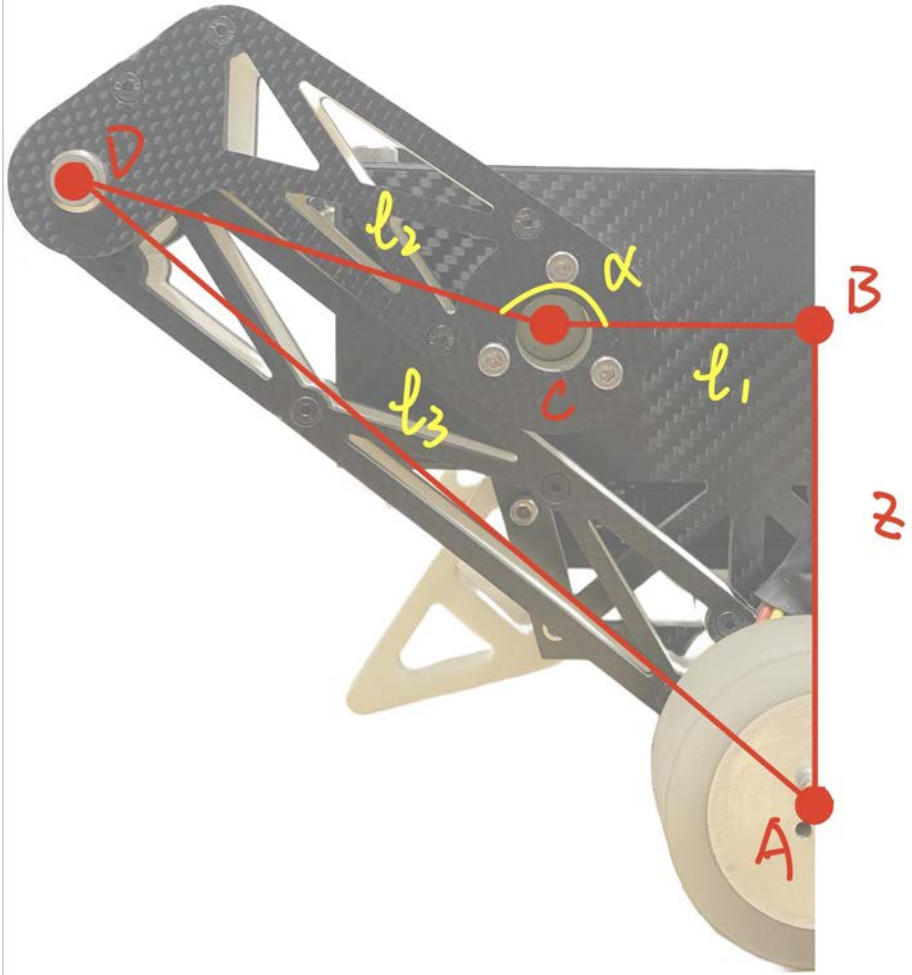


Parameter Declaration

Label	Meaning	Unit
l_1	Half of the length of the body's side.	m
l_2	The length of the active rod.	m
l_3	The length of the connecting rod.	m
z	The height of the body.	m
z_{min}	The minimum height of the body.	m
z_{max}	The maximum height of the body.	m
Δz	The height scope of the body.	m
α	The angle of the leg motors.	$degree$
α_{min}	The minimum angle of the leg motors.	$degree$
α_{max}	The maximum angle of the leg motors.	$degree$

A: the center of the wheel motor
B: the center of the robot body
C: the center of the leg motor
D: the joint of the active and connecting legs

Leg Motion Model

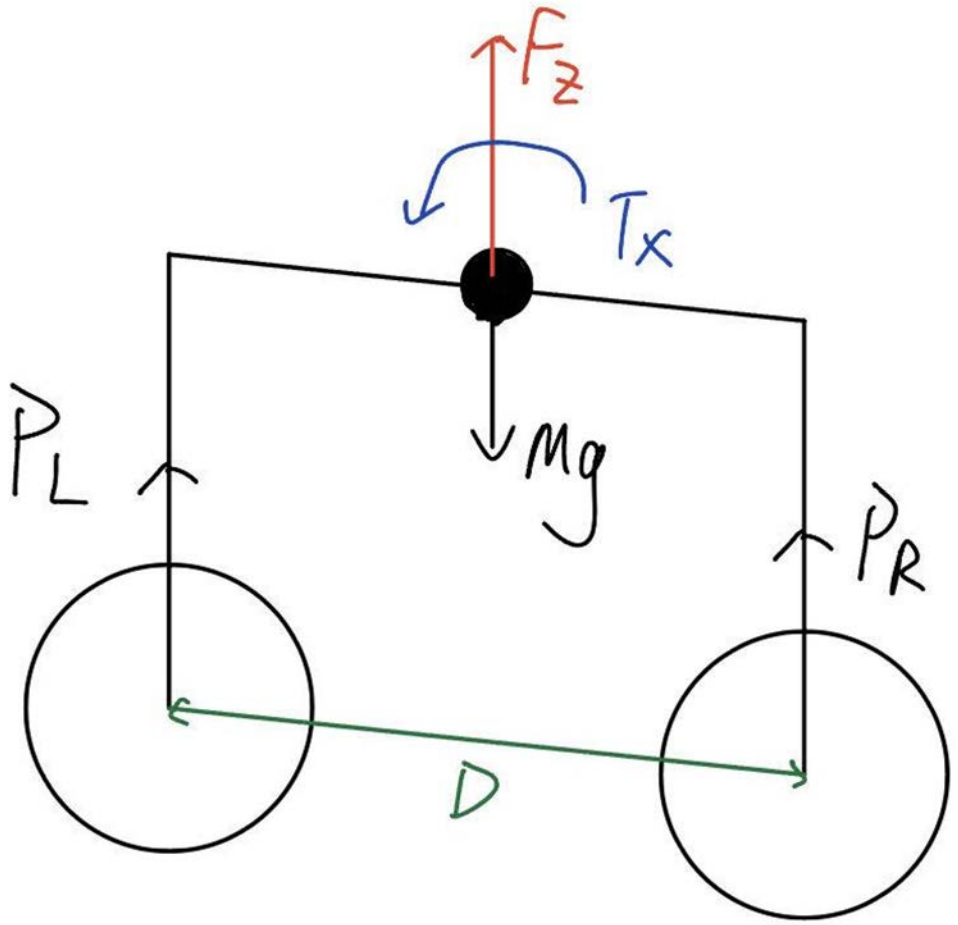


By using Pythagorean theorem:

$$l_3^2 = (z + l_2 sin(\alpha))^2 + (l_1 - l_2 cos(\alpha))^2$$

$$z = \sqrt{l_3^2 - (l_1 - l_2 cos(\alpha))^2} - l_2 sin(\alpha) \quad (16)$$

Leg Motion Model - Support Phase



Net force and torque analysis:

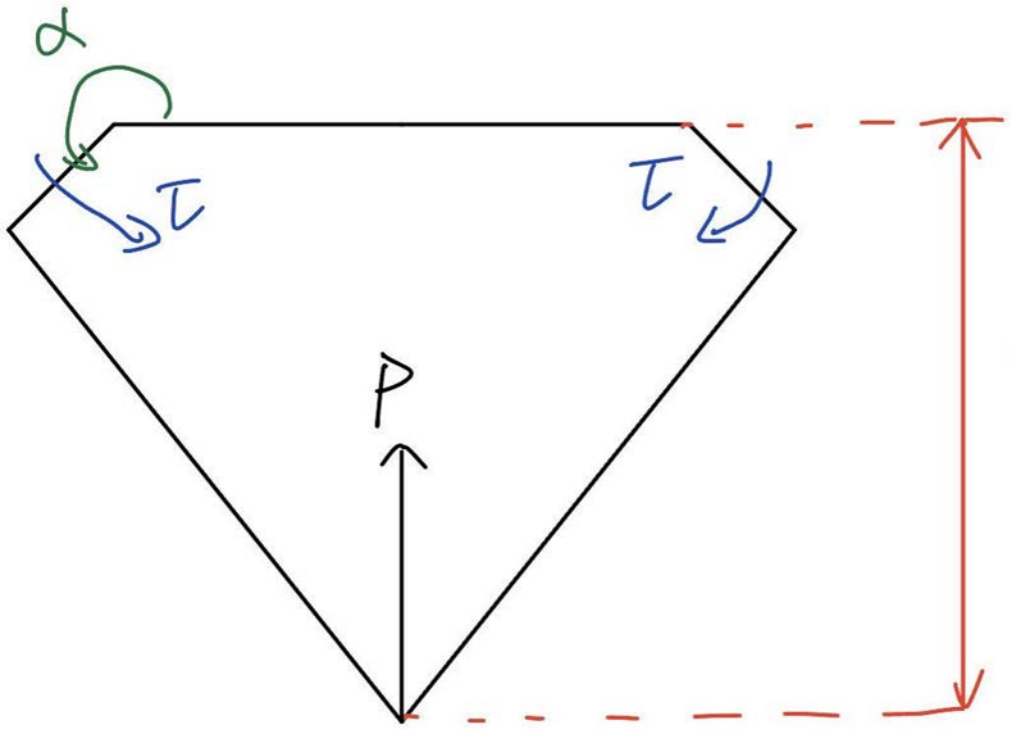
$$F_z = P_L + P_R - Mg \tag{17}$$

$$T_x = (P_R - P_L) * \frac{D}{2} \tag{18}$$

After simplification, the end-effector forces are:

$$\begin{cases} P_L = \dfrac{F_z + Mg}{2} - \dfrac{T_x}{D} \\ P_R = \dfrac{F_z + Mg}{2} + \dfrac{T_x}{D} \end{cases} \tag{19}$$

Leg Motion Model - Five Link model



By using the principle of virtual work:

$$2\tau * \delta\alpha = P * \delta z \quad \text{P represents } P_L \text{ or } P_R. \tag{20}$$

Take the derivative of z w.r.t α in (16):

$$\delta z = -[\frac{(l_1 - l_2 cos(\alpha))l_2 sin(\alpha)}{\sqrt{l_3^2 - (l_1 - l_2 cos(\alpha))^2}} + l_2 cos(\alpha)]\delta\alpha \tag{21}$$

The relationship between torque and end-effector force:

$$\tau = -[\frac{(l_1 - l_2 cos(\alpha))l_2 sin(\alpha)}{\sqrt{l_3^2 - (l_1 - l_2 cos(\alpha))^2}} + l_2 cos(\alpha)] * \frac{P}{2} \tag{22}$$

Control System

- Physical Model
  - Wheel Motion Model
  - Leg Motion Model
- Control Theory
  - Linear Quadratic Regulator(LQR) Algorithm
  - Virtual Model Control(VMC) Algorithm

LQR Algorithm - State Space Model (Pt.1)

Summarize the wheel motion equations:

$$\begin{cases} a\ddot{x} = T_L + T_R - b\ddot{\theta} \\ c\ddot{\theta} = d\theta - e\ddot{x} - (T_L + T_R) \\ \ddot{\psi} = f(T_L - T_R) \end{cases}$$

$$\begin{cases} a = R(\frac{2I_w}{R^2} + 2m + M) \\ b = MlR \\ c = I_y + Ml^2 \\ d = Mgl \\ e = Ml \\ f = \frac{1}{R(\frac{2I_z}{D} + \frac{I_w D}{R^2} + mD)} \end{cases}$$

$$\begin{cases} \ddot{x} = \frac{-bd}{ac - be}\theta + \frac{c + b}{ac - be}(T_L + T_R) \\ \ddot{\theta} = \frac{ad}{ac - be}\theta - \frac{a + e}{ac - be}(T_L + T_R) \\ \ddot{\psi} = f(T_L - T_R) \end{cases}$$

Determine the state vector and input vector

$$\mathrm{X} = [x, \dot{x}, \theta, \dot{\theta}, \psi, \dot{\psi}]^T$$

$$\mathrm{u} = [T_L, T_R]^T$$

LQR Algorithm - State Space Model (Pt.2)

The state space model is:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-bd}{ac-be} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{ad}{ac-be} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{c+b}{ac-be} & \frac{c+b}{ac-be} \\ 0 & 0 \\ -\frac{a+e}{ac-be} & -\frac{a+e}{ac-be} \\ 0 & 0 \\ f & -f \end{bmatrix} \cdot \begin{bmatrix} T_L \\ T_R \end{bmatrix}$$

A

B

Wheel Model Parameters

Label	Value	Unit
m	0.174	kg
M	3.848	kg
I_w	1.4741×10^{-4}	$kg \cdot m^2$
I_y	3.7049207×10^{-2}	$kg \cdot m^2$
I_z	6.563965×10^{-2}	$kg \cdot m^2$
R	0.03	m
l	0.152	m
D	0.3504	m
g	9.81	m/s^2

LQR Algorithm - State Space Model (Pt.3)

Plug in the parameter values:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-bd}{ac-be} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{ad}{ac-be} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -14.7416 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 114.0117 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ \frac{c+b}{ac-be} & \frac{c+b}{ac-be} \\ 0 & 0 \\ -\frac{a+e}{ac-be} & -\frac{a+e}{ac-be} \\ 0 & 0 \\ f & -f \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 21.0112 & 21.0112 \\ 0 & 0 \\ -105.5102 & -105.5102 \\ 0 & 0 \\ 67.6110 & -67.6110 \end{bmatrix}$$

Check controllability :

$$C(A,B) = [B|AB|A^2B|...|A^5B], C(A,B) \in R^{6\times 12} \longrightarrow rank(C(A,B)) = 6 \longrightarrow \text{Controllable}$$

LQR Algorithm - Stability Analysis

The eigenvalues of A matrix:

$$\lambda = [0, 0, 10.6776, -10.6776, 0, 0]^T \longrightarrow \text{Unstable}$$

Feedback control:

$$u = -KX \qquad \text{where } K = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} \end{bmatrix}$$

The cost function is:

$$J = \int_0^\infty (X^T Q X + u^T R u) dt$$

LQR Algorithm - Feedback Gain Selection

Matrix Q and Matrix R:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 90000 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, R = \begin{bmatrix} 1500 & 0 \\ 0 & 1500 \end{bmatrix}$$

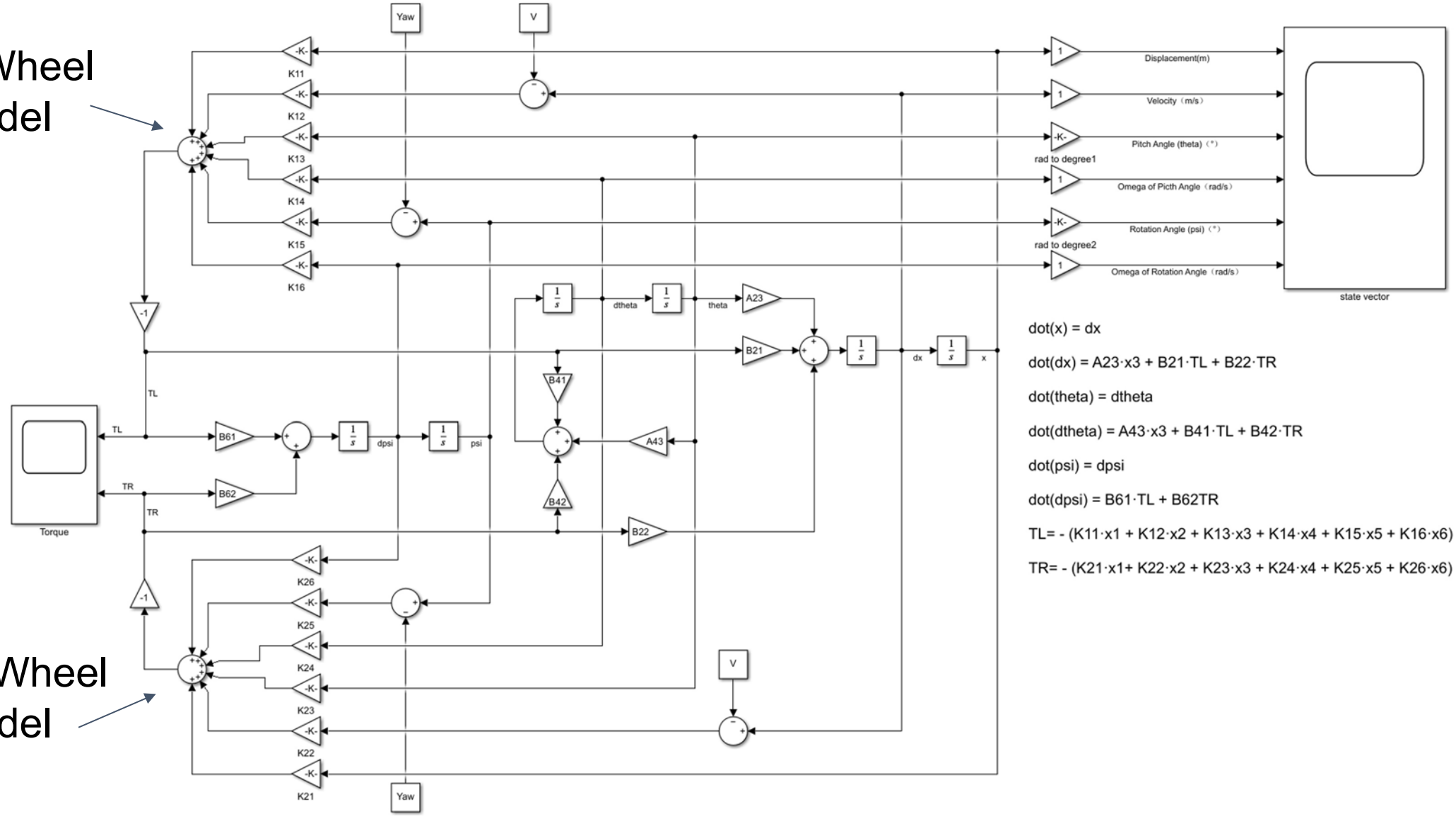
By using Matlab, the feedback gain K is:

$$K = \begin{bmatrix} -0.0183 & -2.5897 & -9.2569 & -1.1094 & 2.5820 & 0.1954 \\ -0.0183 & -2.5897 & -9.2569 & -1.1094 & -2.5820 & -0.1954 \end{bmatrix}$$

LQR Algorithm - Simulink Model

Left Wheel Model

Right Wheel Model



dot(x) = dx

dot(dx) = A23·x3 + B21·TL + B22·TR

dot(theta) = dtheta

dot(dtheta) = A43·x3 + B41·TL + B42·TR

dot(psi) = dpsi

dot(dpsi) = B61·TL + B62TR

TL= - (K11·x1 + K12·x2 + K13·x3 + K14·x4 + K15·x5 + K16·x6)

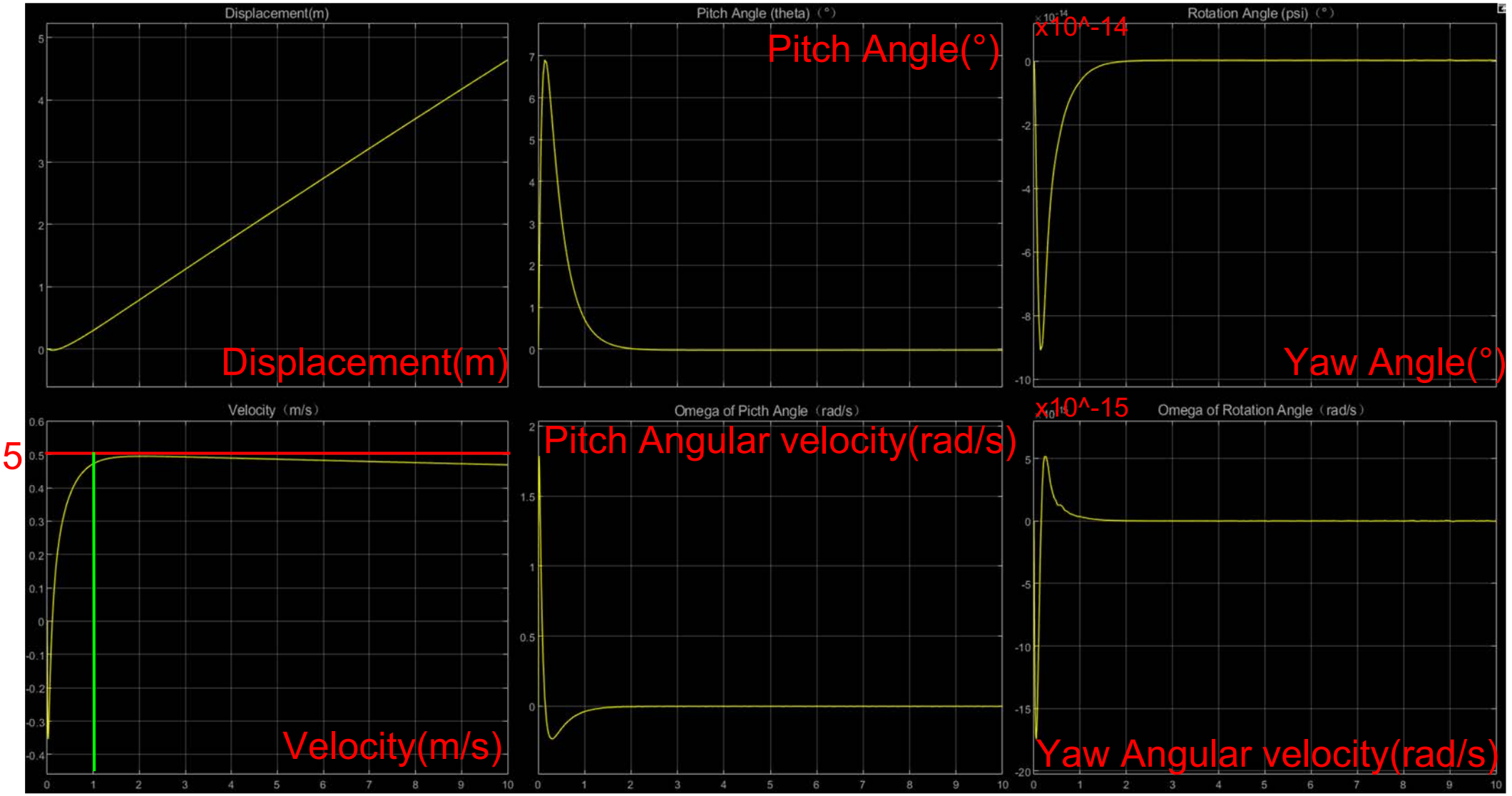
TR= - (K21·x1+ K22·x2 + K23·x3 + K24·x4 + K25·x5 + K26·x6)

LQR Algorithm - Velocity Simulation Test Result (State Vector)

Set Velocity to 0.5 m/s

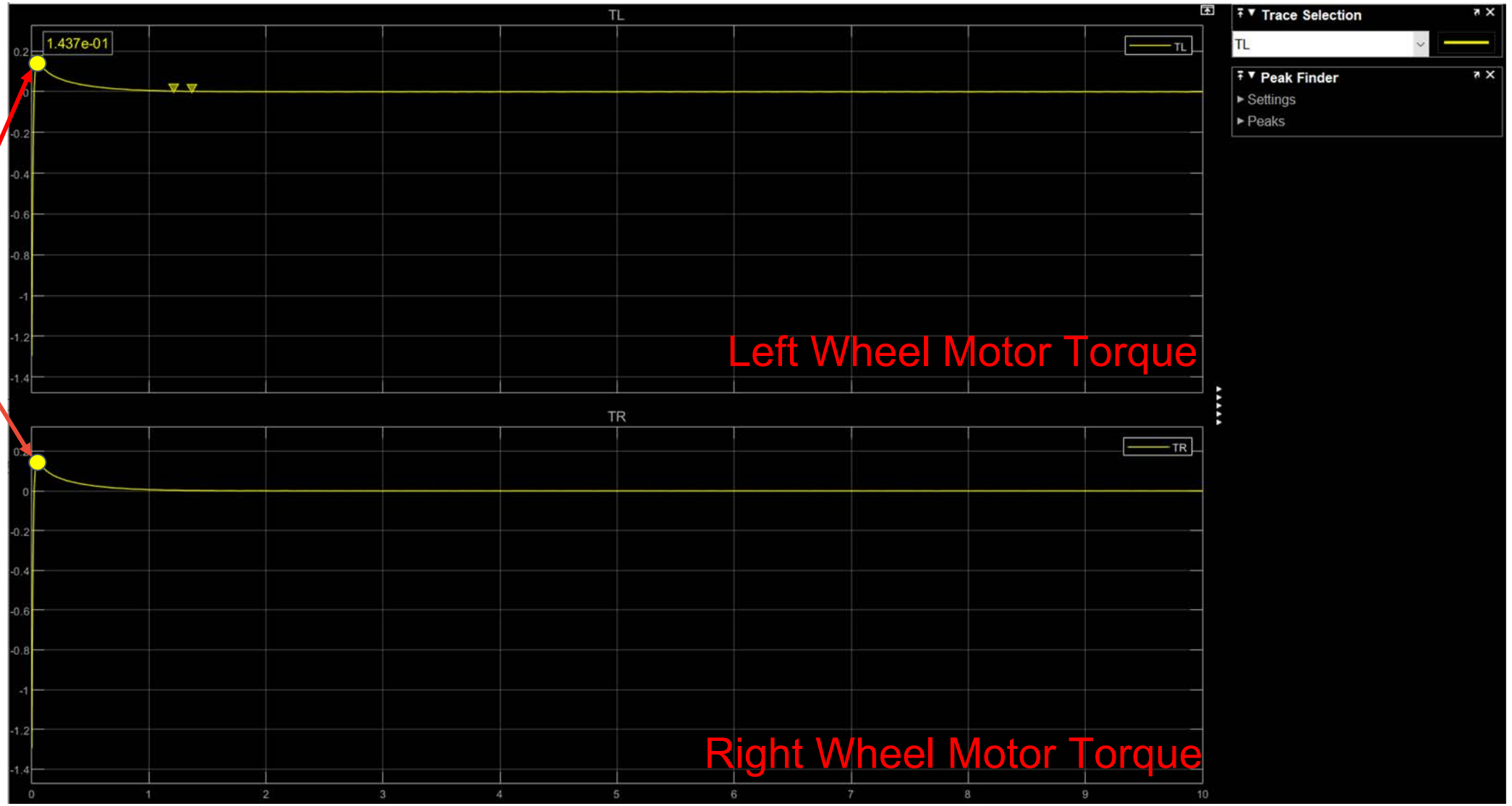
1s converge

Almost no rotation



LQR Algorithm - Velocity Simulation Test Result (Input Vector)

0.1437 Nm
Meet our
requirement
(< 0.3 Nm)

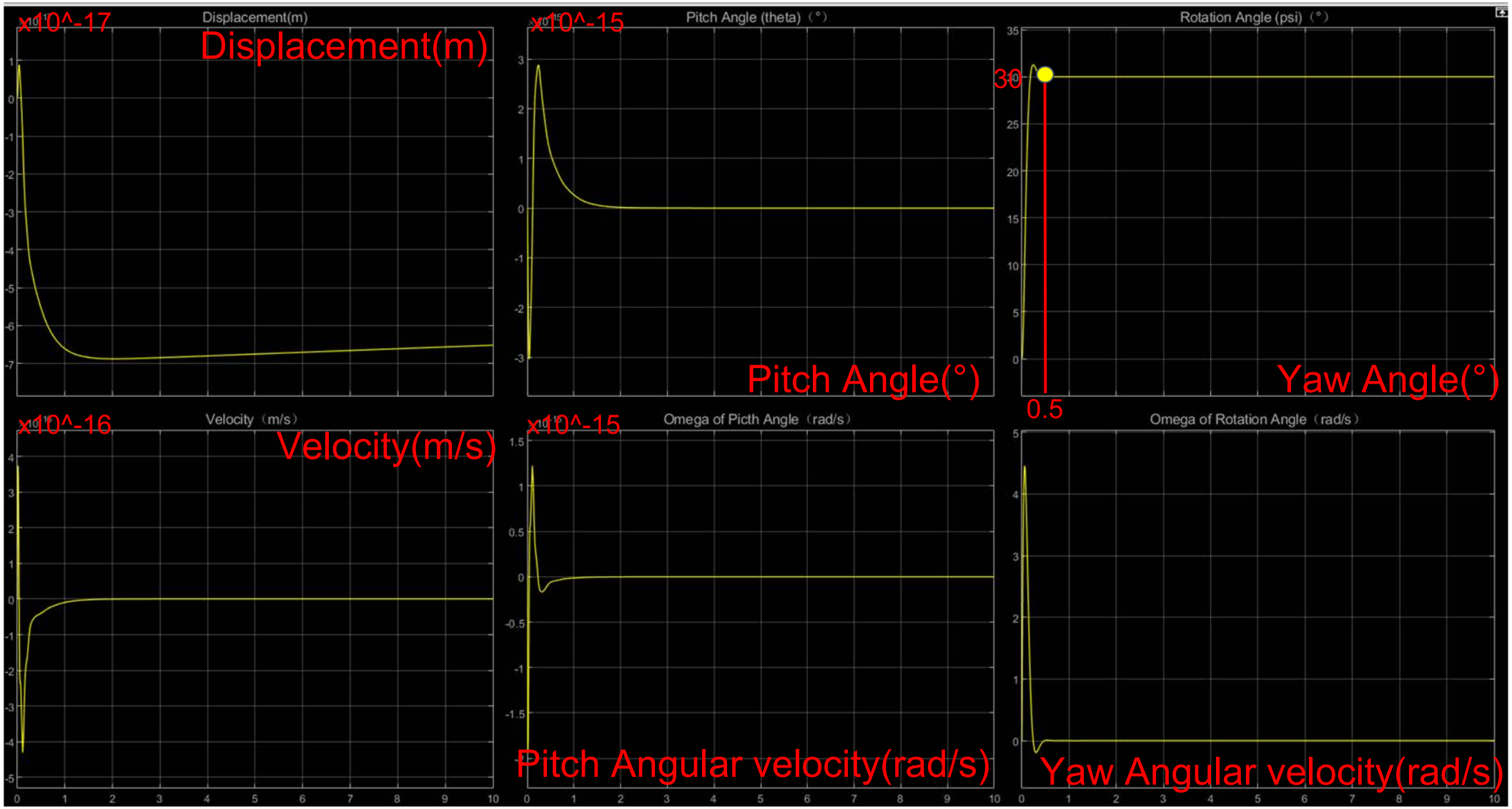


LQR Algorithm - Rotation Simulation Test Result (State Vector)

Set rotation angle to 30°

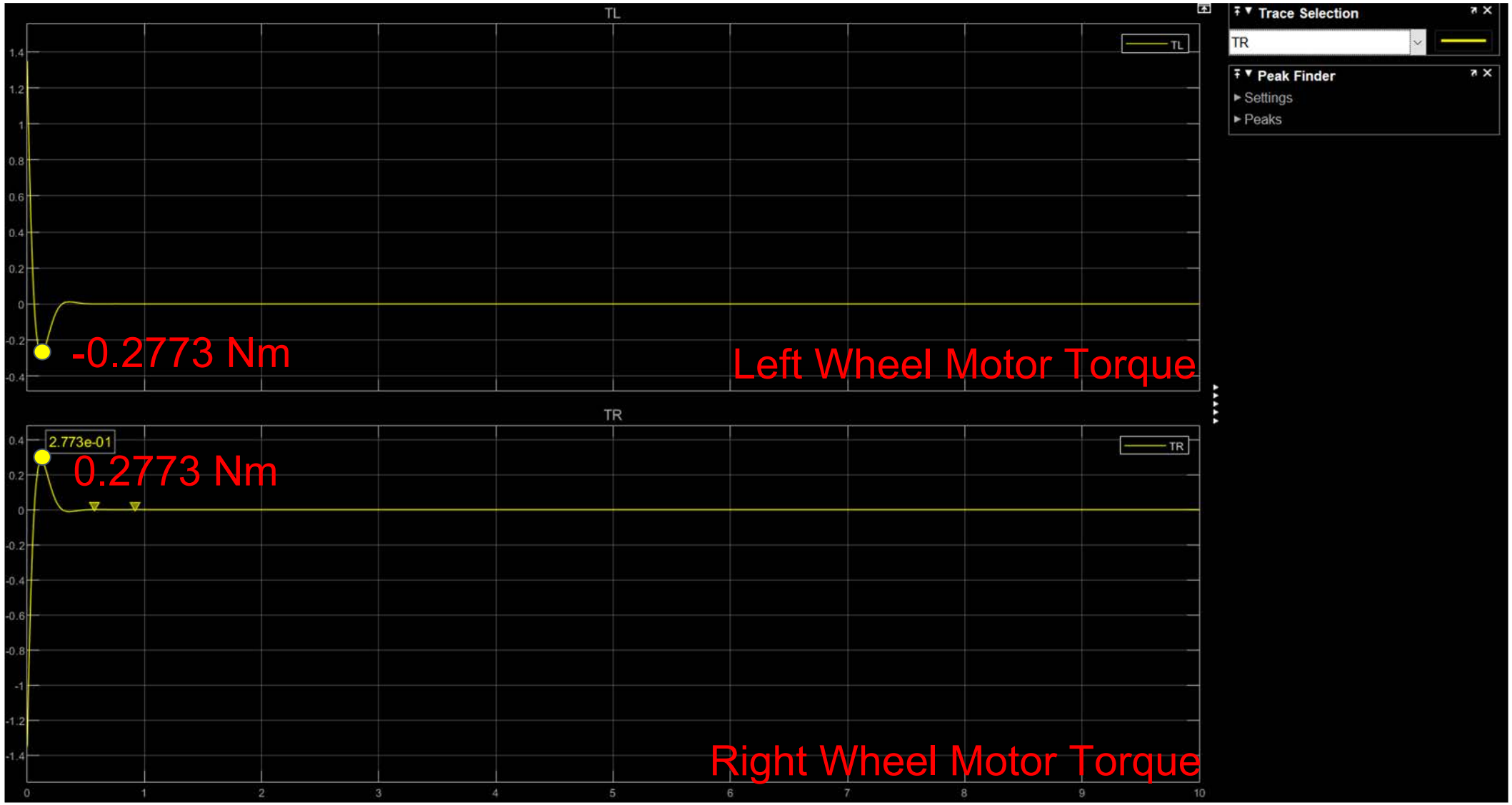
0.5s converge

Almost no translation



LQR Algorithm - Rotation Simulation Test Result (Input Vector)

Left: -0.2773 Nm
Right: 0.2773 Nm
Meet our requirement (< 0.3 Nm)



Control System

- Physical Model
  - Wheel Motion Model
  - Leg Motion Model
- Control Theory
  - Linear Quadratic Regulator(LQR) Algorithm
  - Virtual Model Control(VMC) Algorithm

VMC Algorithm - Spring-Damping System

The force spring-damping system model:

$$F_z = k_1(z_{desired} - z) + c_1(0 - \dot{z}) = M\ddot{z}$$

The torque spring-damping system model:

$$T_x = k_2\phi + c_2(0 - \dot{\phi}) = I_x\ddot{\phi}$$

Leg Model Parameters

Label	Value	Unit
M	3.848	kg
I_x	7.0768703×10^{-2}	$kg \cdot m^2$
l_1	0.05	m
l_2	0.1	m
l_3	0.18	m
D	0.3504	m
g	9.81	m/s^2

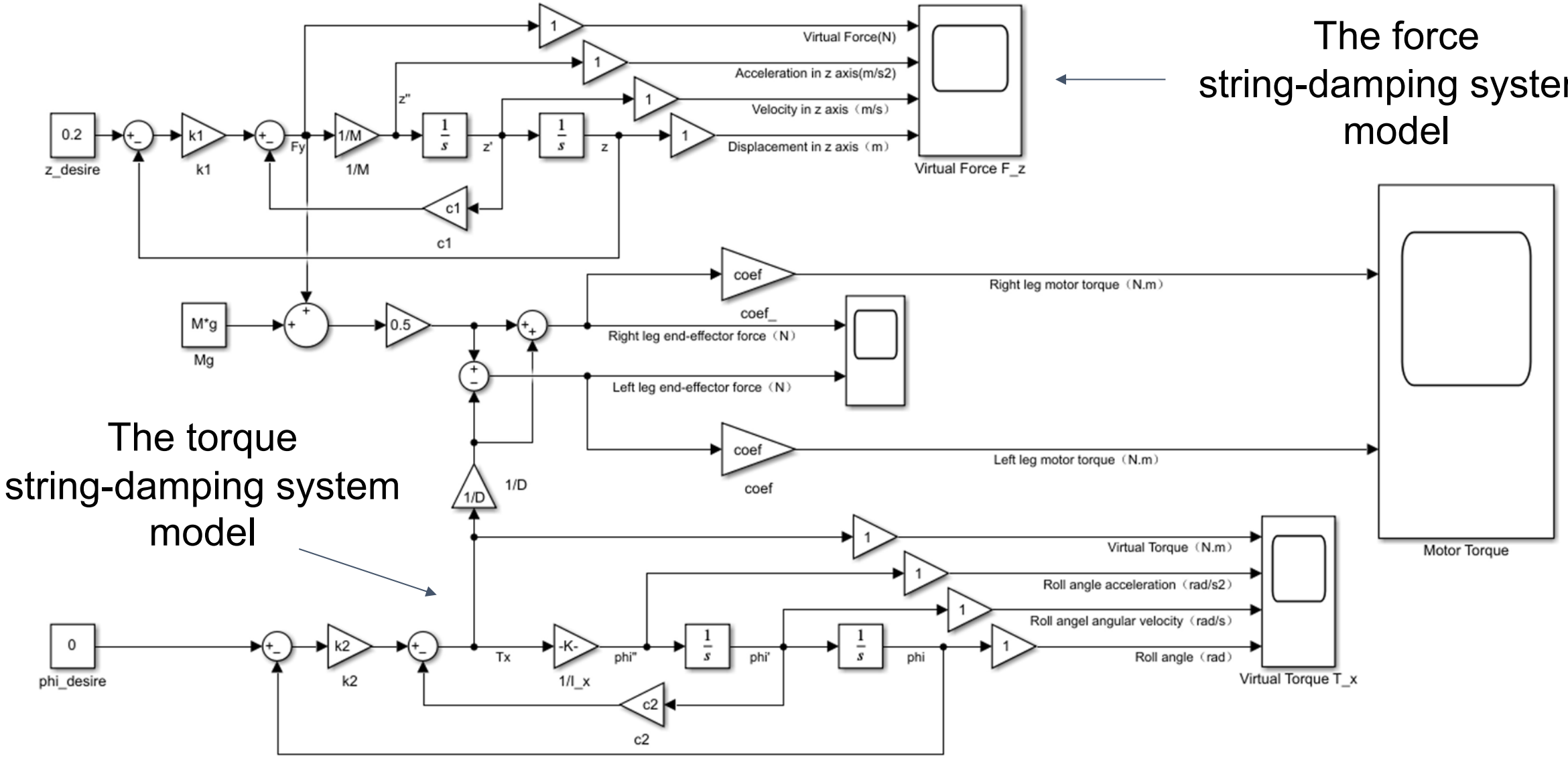
The motor output torque and end-effector forces:

$$\tau = -[\frac{(l_1 - l_2cos(\alpha))l_2sin(\alpha)}{\sqrt{l_3^2 - (l_1 - l_2cos(\alpha))^2}} + l_2cos(\alpha)] * \frac{P}{2}$$

$$\begin{cases} P_L = \dfrac{k_1(z_{desired} - z) - c_1\dot{z} + Mg}{2} - \dfrac{k_2\phi - c_2\dot{\phi}}{D} \\ P_R = \dfrac{k_1(z_{desired} - z) - c_1\dot{z} + Mg}{2} + \dfrac{k_2\phi - c_2\dot{\phi}}{D} \end{cases}$$

P represents P_L or P_R

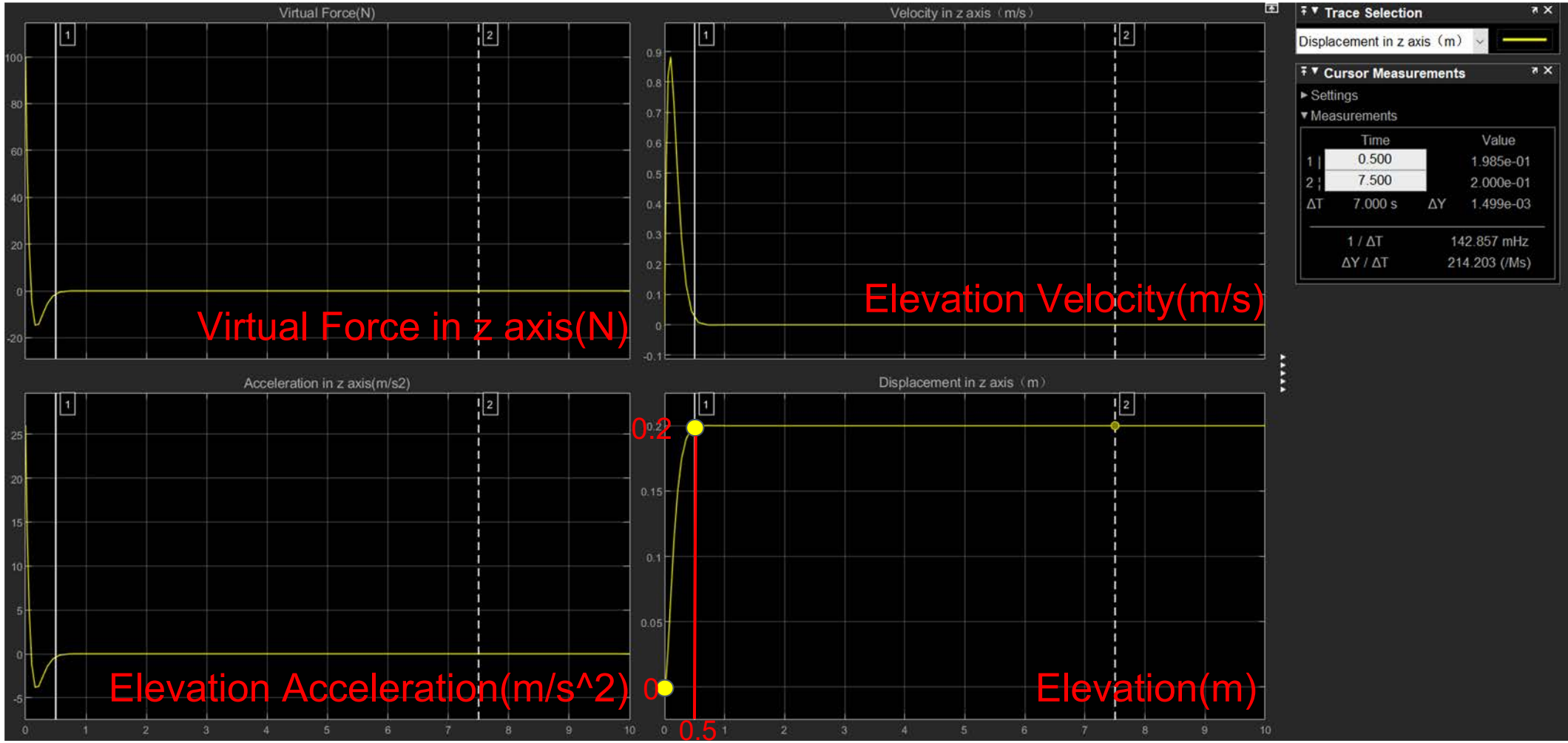
VMC Algorithm - Simulink Model



VMC Algorithm - Elevation Simulation Test Result (Internal Variables)

Set Elevation Height to 0.2 m

0.5s converge



VMC Algorithm - Elevation Simulation Test Result (Torque)

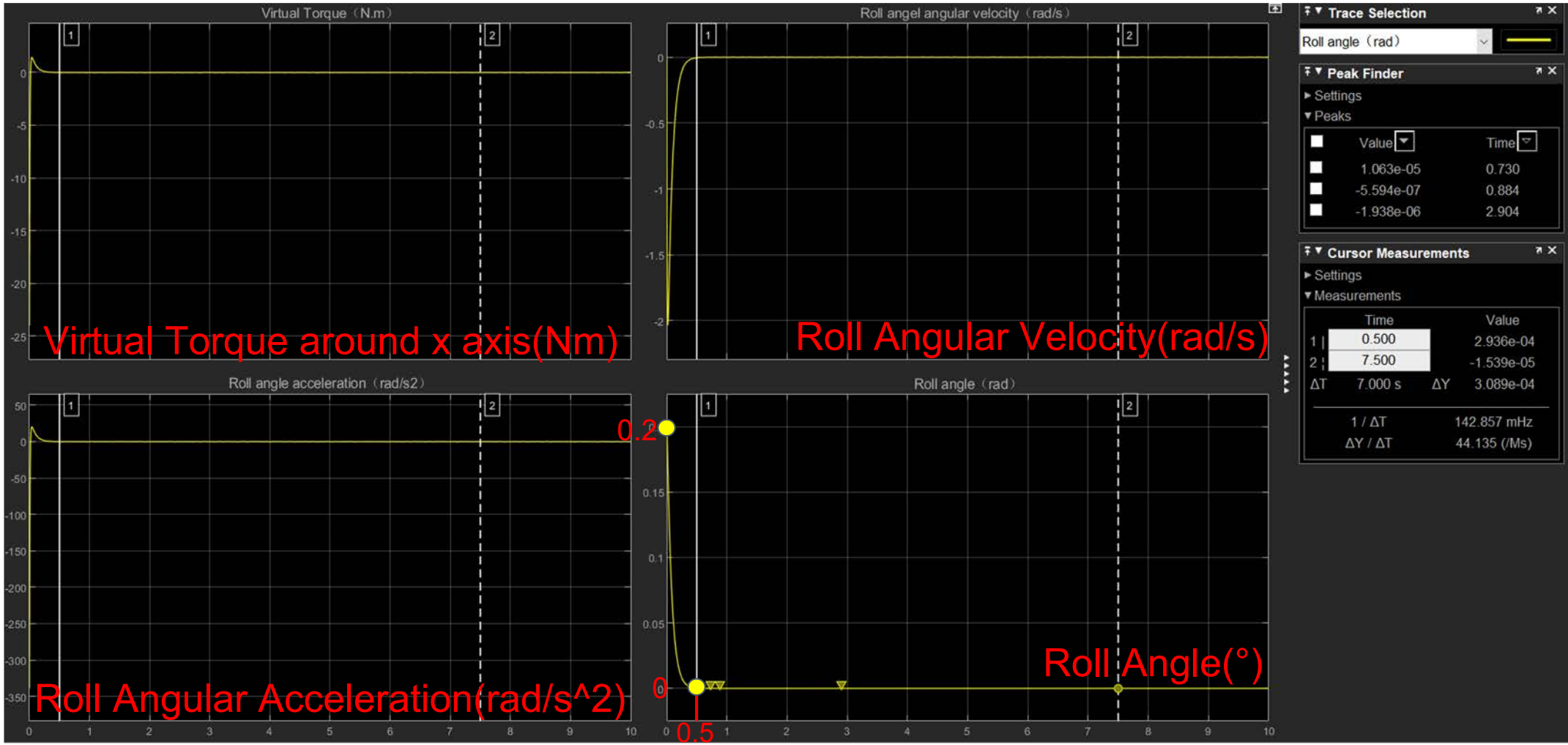
2.944 Nm
Meet our
requirement
(< 7 Nm)

VMC Algorithm - Adaptive Suspension Test (Internal Variables)

Set initial roll angle to 0.2 rad

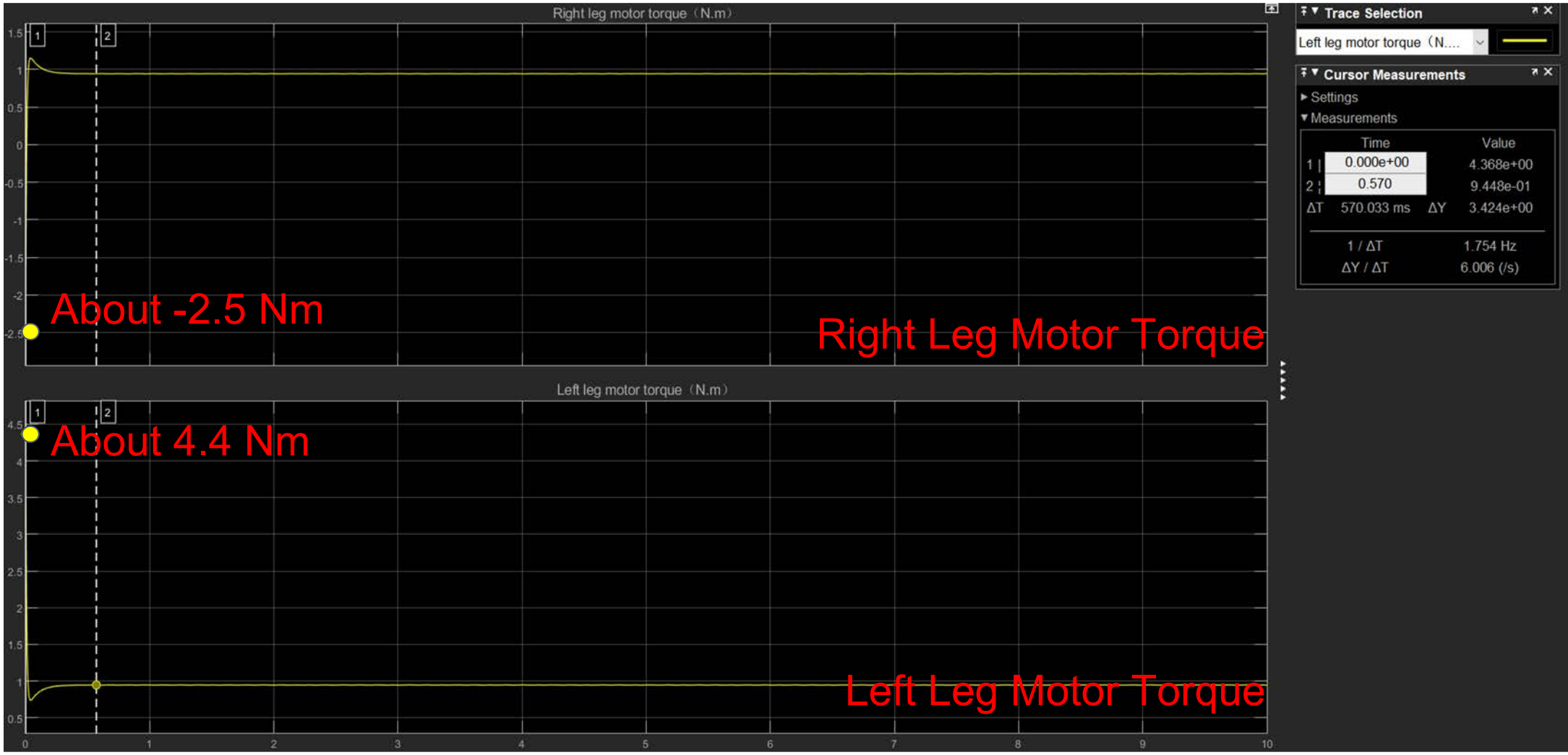
Robot is in a tilted position

0.5s converge



VMC Algorithm - Adaptive Suspension Test (Torque)

Left: 4.4Nm
Right: -2.5 Nm
Meet our requirement (< 7 Nm)



Conclusion & Highlights

- **Mechanical Structure**
  - Carbon fiber plates with 3D printed parts
- **Electrical System**
  - Highly integrated microcontroller board & power board
- **Embedded Software**
  - Multi-protocol communication and IMU sensor fusion
- **Control System**
  - Hybrid control of wheel and leg motors.

Future Work

- **Mechanical Structure**
  - Bigger wheels with more powerful motors
- **Electrical System**
  - More powerful microcontroller
  - IMU heater circuit
- **Embedded Software**
  - More sophisticated IMU filtering algorithms, e.g. Extended Kalman Filter (EKF)
- **Control System**
  - Remodeling the leg linkage motion
  - Jump state analysis with ground detection

References

- Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. Nonlinear complementary filters on the special orthogonal group. IEEE Transactions on automatic control 53.5 (2008): 1203-1217.
- https://zhuanlan.zhihu.com/p/563048952?utm_id=0
- H. GuoXuang, Hyun - GitHub Repository, https://github.com/HuGuoXuang/Hyun
- RoboMaster, Forum Post Title, https://bbs.robomaster.com/forum.php?mod=viewthread&tid=12268
- RoboMaster, Forum Post Title, https://bbs.robomaster.com/forum.php?mod=viewthread&tid=22803

Thank You For Watching

The Grainger College of Engineering

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN