Design, fabrication, simulation, and vision-guided aphid detection and spot spraying for an under-canopy field robot

by

Sudan Baral

B.E., Kathmandu University, 2023

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Carl and Melinda Helwig Department of Biological and Agricultural Engineering
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2026

Approved by:

Major Professor
Dr. Ajay Sharda

Copyright

© Sudan Baral 2026.

Abstract

This thesis presents the design, fabrication, and digital twin simulation of an under-canopy skid-steer robot and the development of a vision-guided spot spraying system for aphid detection and spray deployment on the robot. The robotic platform is purpose-built for 0.76-m row crops such as corn and sorghum. The robot has a 0.61-m end-to-end width, 1.02-m bounding length, 125-kg total mass, and 5-6 hour continuous runtime. The prefabrication design integrates a 3D CAD model with relevant design constraints, finite element analysis on key load-bearing components, and a kinematic simulation to estimate average torque requirements for motors. The mechatronic system includes 48V-24V-12V-5V power system primarily powered by a 48 V, 100 Ah $LiFePO_4$ battery, Nvidia Jetson Orin Agx as the central compute system, ODrive S1 motor controllers, and CAN for communication between controllers, the sprayer ECU, and the central computer. Actuation includes BLDC hubmotors for locomotion and 12V Teejet Solenoid valves for sprayers. Vision system includes two identical stereo cameras, OAK-D SR (short range) by Luxonis. The digital twin of the robot is created in NVIDIA Isaac Sim 5.0. The CAD model of the robot is exported as a URDF and translated as a simulation asset. The simulation setup includes mass correction, suspension joint reconstruction, and motor torque reconstruction from the manufacturer's datasheet. The robot can be actuated using rosbag replay of recorded rosbags from actual robot or over ros2bridge, which reproduces standard ROS messages and subscribes to the replay.

Building on this platform, a real-time aphid detection and spraying system is developed and deployed. A transformer-based RTDETRv2 neural network model based on the ResNet-101 backbone, accelerated with TensorRT, is used as the engine for real-time inference with cameras mounted on the either side of the robot, mounted along the front edge. The detection can be used

trigger the sprayer solenoids via two methods: a CAN-based commercial Teejet ECU and driver package, and I2C-based MOSFET-solenoid actuation driver. One of the key additions is the I²C-based driver, which uses a microcontroller interface between the Jetson AGX and the driver board with a layered safety design and force-stop functionality in case of continuous misfiring. The detections run about 24 fps per camera, with a median capture-to-detection latency of about 88-92ms. A position-based trigger mechanism localizes each detection with stereo depth, tracks it with respect to the boom using transform relationship (tf2) and EKF odometry.

Table of Contents

List of Figures viii
List of Tables ix
Chapter 1 - Introduction 1
1.1 Robotics in Agriculture 1
1.2 Perception and Edge AI for Precision Agriculture 4
1.3 Research Gap and Contributions 5
Chapter 2 - Robot Design, Fabrication and Simulation 7
2.1 Robot Design and Fabrication 7
2.1.1 Design Requirements and Constraints 7
2.1.2 Robotic Platform Chassis and Components 8
2.1.3 Motor Sizing by Simulation 10
2.1.4 Drivetrain 12
2.1.5 3D CAD and Finite Element Analysis 12
2.2 Controller Model 16
2.2.1 Symbols 16
2.2.2 Forward Kinematics 16
2.2.3 Command Pipeline 17
2.2.4 Inverse Kinematics 18
2.3 Electrical Architecture and Power System and Sensors 18
2.3.1 Electrical Architecture and Power 18
2.3.2 Compute and Software Stack 19
2.3.3 Perception and Sensor Suite 20
2.4 Digital Twin Development 21
2.4.1 CAD - URDF Generation 21
2.4.2 Closed-Loop Suspension Approximation 22
2.4.3 Simulation Environment Configuration 24
2.4.4 Physics Parameter Derivation 24
2.4.4.1 Suspension Joint Parameterization 26
2.4.4.2 Contact Material Properties 26

2.4.5 Ground Plane Modeling 26
2.4.6 Motor Torque Model 27
2.4.7 ROS 2 Bridge and Command Injection 28
2.4.7.1 Rosbag Replay Procedure 29
2.5 Motor Characterization for Digital Twin Configuration 30
2.5.1 Motor Specifications 30
2.5.2 Torque Constant Identification (K_t) 31
2.5.3 Joint Drive Parameterization in Isaac Sim 31
2.6 Conclusions 32
Chapter 3 - Real-Time Aphid Detection and Targeted Spot Spraying 33
3.1 Introduction 33
3.2 System Overview 34
3.2.1 Robot platform and compute 34
3.2.2 End-to-end data flow 35
3.3 Aphid Detection 37
3.3.1 Camera selection 37
3.3.2 Model selection 37
3.3.3 Final model 37
3.3.4 Deployment with TensorRT 40
3.3.5 Camera configuration 41
3.4 Spray Actuation 41
3.4.1 Commercial CAN sprayer: approach and limitations 42
3.4.2 Custom I^2C solenoid driver 43
3.4.3 CAN back end retained as software fallback 45
3.5 Position-Based Spray Triggering 45
3.5.1 Position-based trigger 45
3.5.2 Time-based trigger as fallback 47
3.6 System Verification 47
3.6.1 End-to-end latency and fail-safe behavior 48
3.6.2 State estimation 49
3.6.3 Position-based trigger: simulation and hardware 50

3.7 Discussion 52
3.8 Summary 54
Bibliography 56

List of Figures

Figure 2.1. Corn row spacings in North America as a percentage of total acres, 2015 (Pioneer Brand Concentration Survey) 8
Figure 2.2. Decker-03 robot platform: assembled chassis (left) and the robot fitted with the spray booms at the corn field (right). 9
Figure 2.3. Simscape Multibody model of the robot used for motor sizing. 10
Figure 2.4. Simulink block generating the smooth position, velocity, and acceleration profile for a wheel joint. 11
Figure 2.5. Combination of the straight-line and turning profiles into the four wheel commands (A–D). 11
Figure 2.6. Wheel joint torques (N m) over the simulated straight-line drive and zero-radius turn, one trace per wheel. 11
Figure 2.7. Isometric view of robot CAD assembly in Solidworks 13
Figure 2.8. Dead-weight static load case: meshed swing-arm assembly with fixtures and loads (left) and resulting von Mises stress (right). 14
Figure 2.9. Two-foot free-fall impact load case: von Mises stress (left) and factor of safety (right). 15
Figure 2.10. Schematic diagram of electrical system 19
Figure 2.11. Flowchart of the methodology for generating the URDF from the SolidWorks CAD assembly 22
Figure 2.12. Open loop kinematic chain wheel arm SolidWorks CAD assembly 23
Figure 2.13. Isaac SIM controller graph network 25
Figure 3.1. Software architecture of the solenoid spray path. 36
Figure 3.2. RTDETR Architecture 39
Figure 3.3. Actuation signal chain 43
Figure 3.4. Aphid detection and triggering lab test setup. 52

List of Tables

Table 2.1. Mechanical Properties: Al 6061-T6 vs Al 7075-T6 vs SS 304. 8
Table 2.2. Decker-03 Hub Motor Specifications (KK6 Fat Series). 12
Table 2.3. Finite element analysis simulation parameters. 14
Table 2.4. Decker-03 Perception and Navigation Sensor Suite. 21
Table 2.5. Isaac Sim Physics Parameters and Their Sources. 24
Table 2.6. KK6 Fat Series motor datasheet parameters. 30
Table 2.7. Summary of Identified Motor Parameters. 31
Table 3.1. Staged development of the spot-spraying system. 33
Table 3.2. Robot platform and compute hardware. 34
Table 3.3. Main ROS 2 topics of the spray pipeline. 36
Table 3.4. RT-DETRv2 validation metrics for the two evaluated backbones. 38
Table 3.5. Solenoid driver node interface. 44
Table 3.6. Solenoid driver node parameters. 44
Table 3.7. Sensor and boom frames relative to body. 46
Table 3.8. State-estimation accuracy across six runs. 49
Table 3.9. Simulated spray-window accuracy of the position-based trigger. 50
Table 3.10. Spray events in the two position-based trigger runs. 51

Chapter 1 - Introduction

1.1 Robotics in Agriculture

This study can be observed as a system design study, which can be divided into two sections: mechatronic design, fabrication and simulation of an under-canopy skid-steer robot, and development of vision-guided aphid detection and spraying system. This chapter review the literature pertaining to agricultural robotics, digital twins and the sim-to-real gap.

The global industrial sector is rushing towards full automation using robotics, agricultural sector is also converging on adoption of robotic platforms across different agricultural avenues. Labor shortages in developed countries is well documented, US farm worker wages have risen faster than non-farm wages over the past decade, tightening up profit margins and the labor market (Calvin et al., 2022). At the same time, the demand to feed the global population is projected to approach 9.8 billion by 2050 (United Nations, 2017).

Row-crop agriculture is a particularly demanding environment for autonomous ground vehicles. Corn and sorghum are planted in standardized 0.76-m row spacing across approximately 85% of North American corn acres (Pioneer, n.d.), creating narrow corridors that constrain platform geometry and demand precise lateral control. Once canopy closure occurs mid-season, these corridors become GNSS-denied, as satellite signals are attenuated or reflected by dense vegetation, rendering standard RTK-GNSS navigation unreliable. The sub-canopy region is nonetheless agriculturally significant: it hosts insect-pest hotspots, is the primary zone for targeted pesticide application, and is inaccessible to aerial platforms that cannot penetrate the canopy (Gai et al., 2021).

Prior work on under-canopy platforms has largely bifurcated into two categories: lightweight scouting robots designed exclusively for sensing, and large-scale machinery that

cannot fit within standard row spacings. (Kim et al., 2024) developed the Purdue AgBot (P-AgBot) and its P-AgSLAM framework, a LiDAR-based simultaneous localization and mapping system tailored to cornfield morphology for in-row and under-canopy monitoring. (Schmitz et al. 2022) proposed a reconfigurable ground vehicle for sub-canopy scouting that demonstrated shape-changing capability for inter- and intra-row traversal, though its scale limited it to sensing rather than intervention. (Gai et al., 2021) demonstrated the use of a depth camera for crop-row detection and mapping to support under-canopy navigation. Across these efforts the emphasis has been on perception and navigation; comparatively little work combines the mechanical payload capacity and suspension compliance needed for an intervention platform (such as a sprayer) with a model of the vehicle itself.

To traverse and navigate movement-constrained rows, a more viable type of drive mechanism is a skid-steer mechanism. Skid-steer kinematics are the dominant locomotion model for compact agricultural robots because of their zero-turning-radius capability, mechanical simplicity, and robustness on uneven terrain. Unlike Ackermann steering, skid steering has no explicit kinematic relation between wheel speeds and chassis motion; the instantaneous centers of rotation shift with wheel–terrain slip, so practical models are identified experimentally (Mandow et al., 2007). (Botta et al., 2023) analyzed the power balance, handling, and traction subsystem of an articulated skid-steering agricultural robot, showing that drivetrain efficiency and wheel–terrain interaction dominate field energy consumption. These slip-dependent dynamics make skid-steer platforms both attractive for field robustness and challenging to simulate faithfully, motivating the digital-twin treatment developed in this thesis.

A digital twin (DT) is an abstract mathematical representation of a physical system built from its components, joints, and their relationships, which can be exercised as a virtual

counterpart that replicates the functionalities of the physical system. The degree of synchronization between the twin and its physical counterpart varies with the use case, ranging from offline parameter-derived models used for design and evaluation to tightly sensor-synchronized models used for live monitoring (Tao et al., 2019). In robotics, DTs support pre-deployment testing of control policies, predictive maintenance, synthetic data generation for perception training, and large-scale reinforcement learning in place of costly real-world trials. (Pylianidis et al., 2021) surveyed the introduction of digital twins to agriculture and found that most agricultural DT efforts target decision support and data generation rather than models of the machine's own dynamics.

Simulation platforms have evolved from Gazebo, long the ROS standard but limited in rendering fidelity and high-load contact accuracy (Koenig & Howard, 2004), toward GPU-accelerated engines such as NVIDIA Isaac Sim, which couples the PhysX 5 solver with photorealistic rendering and a native ROS 2 bridge (NVIDIA, 2024). The value of any DT is proportional to the fidelity with which it reproduces real dynamics, and the residual sim-to-real gap remains an active challenge. (Müller et al., 2022) attribute the gap to incomplete system understanding, un-modeled dynamics, and computational constraints, and propose closing it through real-world feedback. A complementary strategy from the learning community is domain randomization, in which simulation parameters are varied so that the real world appears as just another variation to the model (Tobin et al., 2017). Off-road platforms add terra-mechanics, soil deformation, slip, and variable friction, that substantially increase modeling complexity relative to the indoor mobile robots for which most DT validation has been reported. The present work sets simulation parameters from measurements and datasheets rather than fitting them to observed behavior; sim-to-real validation is future work.

1.2 Perception and Edge AI for Precision Agriculture

Deep-learning object detection has progressed from two-stage region-proposal detectors such as Faster R-CNN (Ren et al., 2015), through single-stage real-time detectors led by the YOLO family (Redmon et al., 2016), to transformer-based detectors beginning with DETR, which recast detection as direct set prediction and removed hand-designed components such as non-maximum suppression (Carion et al., 2020). DETR's high latency was subsequently addressed by RT-DETR, which (Zhao et al., 2024) showed can outperform comparable YOLO models at real-time speeds, and by RT-DETRv2, which adds a bag of training and inference refinements for an improved accuracy/latency trade-off (Lv et al., 2024). The global attention in these transformer detectors is attractive for small objects in cluttered agricultural scenes, although it comes at a computational cost.

Two practical issues dominate field deployment. First, the domain gap: models trained on curated, close-up imagery often generalize poorly to wide-field, compressed frames captured on a moving robot, where targets occupy only a handful of pixels. Second, the compute constraint: inference must run on embedded hardware under power and thermal limits. Edge accelerators such as the NVIDIA Jetson family, together with inference optimizers such as TensorRT, have made on-board real-time detection feasible, and a growing body of work deploys such pipelines for in-field crop and pest sensing. These considerations directly motivate the detector selection and TensorRT deployment described in Chapter 3.

Among the key pests, Aphids are the ones we are interested. It poses a specific problem, that a UAV is not either able to detect or spray from above, as the aphids reside in the underside of the leaf. This poses a need for ground vehicles to access the under-canopy. Moreover, aphids are among the most economically damaging pests of row crops; the soybean aphid alone

prompted a major expansion of insecticide use across North America following its introduction, making it a focus of integrated and site-specific management (Ragsdale et al., 2011). Computer vision has been applied to aphid sensing for over a decade: (Chen et al. 2018) segmented and counted aphid nymphs on leaves with convolutional neural networks, and subsequent work has extended detection and counting to field and climate-chamber imagery. More broadly, (Rai et al. 2023) review the rapid adoption of deep learning for precision weed and pest management, noting a shift from classification toward detection and segmentation models suitable for actuation.

Translating detection into targeted chemical application is the goal of precision spot spraying, which applies product only where a target is detected and can sharply reduce chemical use relative to broadcast application. (Partel et al. 2019) developed and evaluated a low-cost, AI-based smart sprayer for site-specific weed management. Most directly related to this thesis, (Upadhyay et al. 2024) at North Dakota State University developed and field-evaluated a machine-vision and deep-learning smart-sprayer system for site-specific weed management in row crops using an edge-computing approach, demonstrating the full detection-to-nozzle pipeline on embedded hardware. These systems establish the feasibility of vision-guided spot spraying; however, they target weeds at or near ground level and are not integrated with a purpose-built under-canopy platform, nor do they address the specific detection and actuation-timing problems of spraying a mobile pest such as the aphid from a moving robot within the canopy.

1.3 Research Gap and Contributions

The reviewed literature establishes under-canopy platforms, skid-steer modeling, agricultural digital twins, real-time detection, and precision spot spraying as individually mature.

This thesis addresses that gap through two primary technical contributions, the physical platform (Chapter 2) and the on-board aphid detect-and-spray capability (Chapter 3), and one supporting methodological contribution covering the digital twin workflow (Chapter 2):

1. A versatile under-canopy chassis, Decker-03, purpose-built for 0.76-m row crops and usable for scouting, spraying, and transportation, with its complete mechanical, electrical, and software architecture documented (Chapter 2).
2. A reproducible Isaac Sim digital twin of the platform, intended as a development environment for continuing simulation-based work on navigation, detection, and spraying prior to field deployment (Chapter 2). Sim-to-real validation of the twin is left to future work.
3. A real-time aphid detect-and-spray system built on a TensorRT-accelerated RT-DETRv2 detector with a layered fail-safe design and a targeted sprayer mechanism, where each aphid was seen, using stereo depth, tf2, and EKF odometry (Chapter 3).

The thesis addresses the following research question: how can a compact under-canopy field robot be designed and equipped for real-time targeted intervention in 0.76-m row crops? The question is examined through two primary strands, the physical platform (Chapter 2) and the vision-guided detect-and-spray capability deployed on it (Chapter 3), supported by a digital twin of the platform that sets up a simulation environment for continuing development.

The two main technical chapters are complementary but independent, joined by the shared physical platform. Chapter 2 presents the design and fabrication of Decker-03 and documents a methodology for its digital twin in Isaac Sim. Chapter 3 presents the detect-and-spray system developed on the physical platform. Applying the digital twin to the Chapter 3 pipeline is identified as continuing future work.

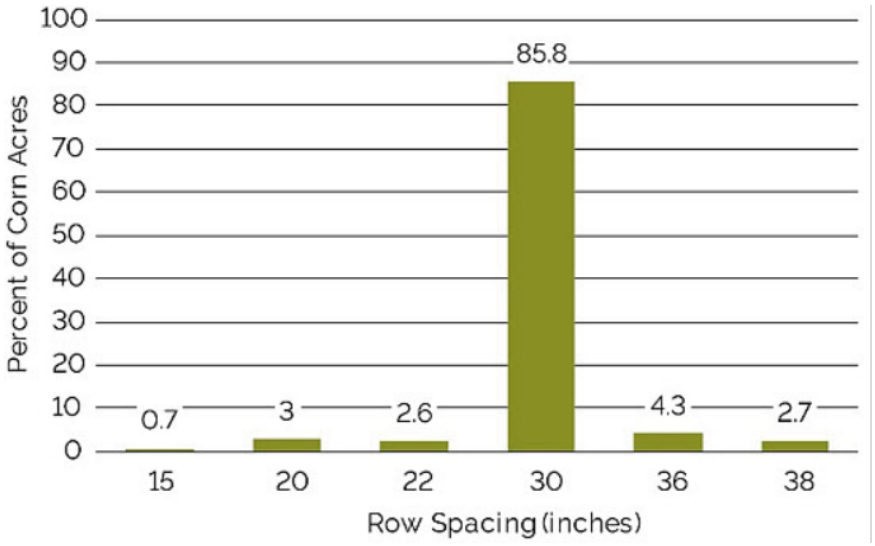
Chapter 2 - Robot Design, Fabrication and Simulation

2.1 Robot Design and Fabrication

2.1.1 Design Requirements and Constraints

The primary geometric constraint for the robot design was the 0.76-m standardized corn and sorghum row spacing across approximately 85% of North American corn acres (Fig. 2.1). Commercial robots with high payload capacity (100 kg) such as Husky A300 are too wide (0.70 m) or robots with narrow body width, such as Terrasentia, have low volumetric payload capacity. To fit within the 0.76-m row gap with some error margin space to spare, and still be capable of providing large carrying capacity, the robot was purpose-built. Robot height and length were primarily dictated by the battery, compute system, and sprayer subsystem. The carrying capacity was set at 100kg of added live weight, supporting up to a 25-gallon spray tank. The operation cycle was specified at 5-6 hours with 1m/s - 2m/s continuous traverse. The battery was required to handle the maximum current demand of all motor wheels, the compute, and the sprayer subsystem. Load-bearing structures were designed to survive the robot across unpredictable terrains and unforeseen loadings.

Figure 2.1. Corn row spacing in North America as a percentage of total acres, 2015 (Pioneer Brand Concentration Survey)



2.1.2 Robotic Platform Chassis and Components

The chassis was designed around four considerations: weight-to-strength ratio, manufacturability, modularity, and ease of access to internals. The materials preferred for chassis development were Al 6061 T6, Al 7075, and SS 304 because of their pros in terms of strength and availability. Al 6061-T6 was selected as the primary structural material for its comparable yield strength, low weight, machinability, and weld-ability.

Table 2.1. Mechanical Properties: Al 6061-T6 vs Al 7075-T6 vs SS 304.

Property	Al 6061-T6	Al 7075-T6	SS 304
Density	2.70 g cm^{-3}	2.81 g cm^{-3}	8.00 g cm^{-3}
Yield strength	276 MPa	503 MPa	215 MPa
Ultimate tensile strength	310 MPa	572 MPa	505–515 MPa
Specific strength (UTS/ρ)	115 kN m kg^{-1}	204 kN m kg^{-1}	63 kN m kg^{-1}
Young's modulus	69 GPa	72 GPa	193 GPa
Machinability	Excellent	Good	Difficult
Weldability	Excellent	Poor	Good
Corrosion resistance	Good	Moderate	Excellent

Manufacturing was done at the Kansas State University Technology Development Institute (KSU TDI), which provided aluminum sheet bending capability. The chassis is divided into upper and lower decks for simplified assembly, disassembly, and internal access.

The lower deck is divided into three chambers: the central chamber houses the 48 V $LiFePO_4$ battery pack, while the end chambers house ODrive motor controllers and the battery charging port. The upper deck houses all electrical units and the central computer platform. The compute board, IMU, and GNSS module are mounted on a sliding tray accessible from the front face of the robot, with USB-A and Ethernet ports on the tray's outer wall. The upper deck rear section provides a 12 V multi-output terminal for directly powering sprayer components. The emergency stop (E-stop) is located on the upper deck exterior. A raised platform above the upper deck provides mounting for the spray tank and LiDAR.

Figure 2.2. Decker-03 robot platform: assembled chassis (left) and the robot fitted with the spray booms at the corn field (right).



2.1.3 Motor Sizing by Simulation

To obtain a preliminary estimation of torque requirements of the motors before purchase and assembly, a simple MATLAB/Simulink simulation (R2023, MathWorks Inc., Natick, MA, USA) was performed with an initial idea of what the robot may look like. Each wheel joint was driven by a prescribed smooth position profile, generated by the block in the figures below, and the solver computed the torque each wheel needed to follow it. Two maneuvers were combined in one run, a straight-line drive followed by a zero-radius turn, obtained by summing per-wheel profiles with the appropriate sign for each maneuver with ground(concrete)-tire static friction coefficient of 0.7. During straight-line driving, the wheel torques stayed within about ±8 N m. During the zero-radius turn, which demands far more torque on a skid-steer platform because the wheels must slip sideways, two wheels held about 35 N m and the other two oscillated between 40 and 55 N m, peaking near 57 N m; the brief spikes in the first 0.3 s are the model settling onto the ground at start-up rather than part of either maneuver. The highest demand, during turning, is about 60 % of the 96.15 N m peak torque of the KK6 hub motor, which was therefore selected with a margin of about 1.7.

Figure 2.3. Simscape Multibody model of the robot used for motor sizing.

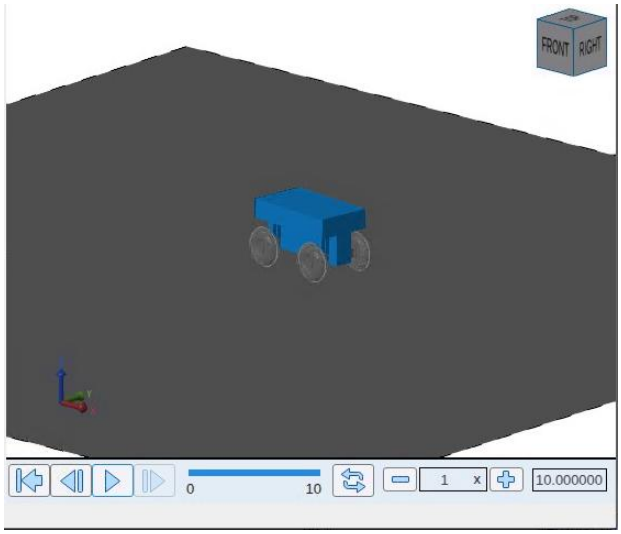


Figure 2.4. Simulink block generating the smooth position, velocity, and acceleration profile for a wheel joint.

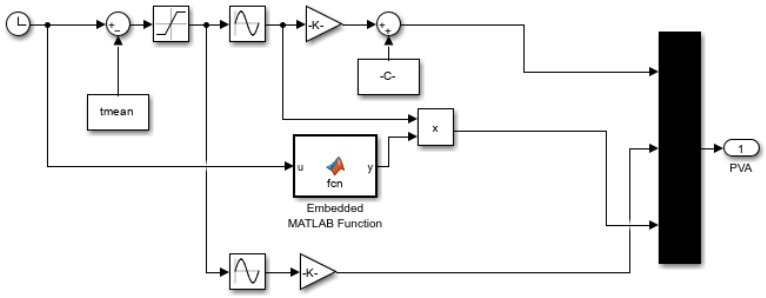


Figure 2.5. Combination of the straight-line and turning profiles into the four wheel commands (A–D).

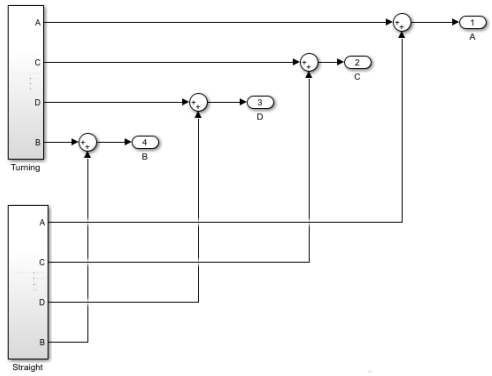
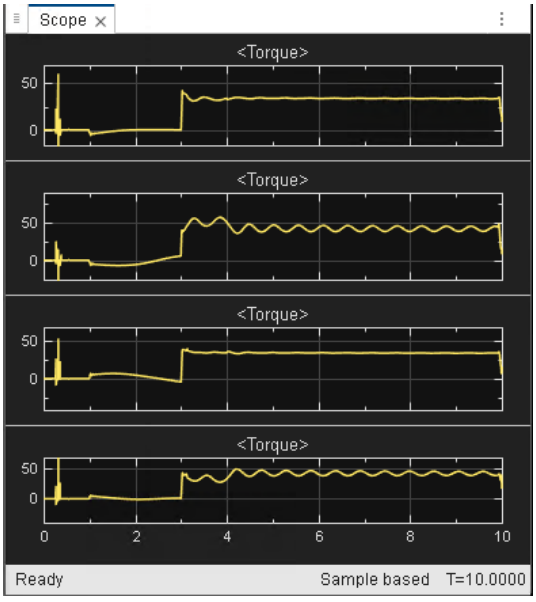


Figure 2.6. Wheel joint torques (N m) over the simulated straight-line drive and zero-radius turn, one trace per wheel.



2.1.4 Drivetrain

Robot Locomotion is provided by four 48 V brushless DC (BLDC) geared hub motors (KK6 Fat Series, UU Motor Technology Co., Ltd., Shenzhen, China), each rated at a peak torque of 96.15 N m at a stall current of 23.365 A, with a 5:1 internal gear reduction.

The ODrive S1 motor controllers are configured to allow a maximum phase current of 35 A to support high-traction field maneuvers. Key motor specifications are tabulated in Table 2.2.

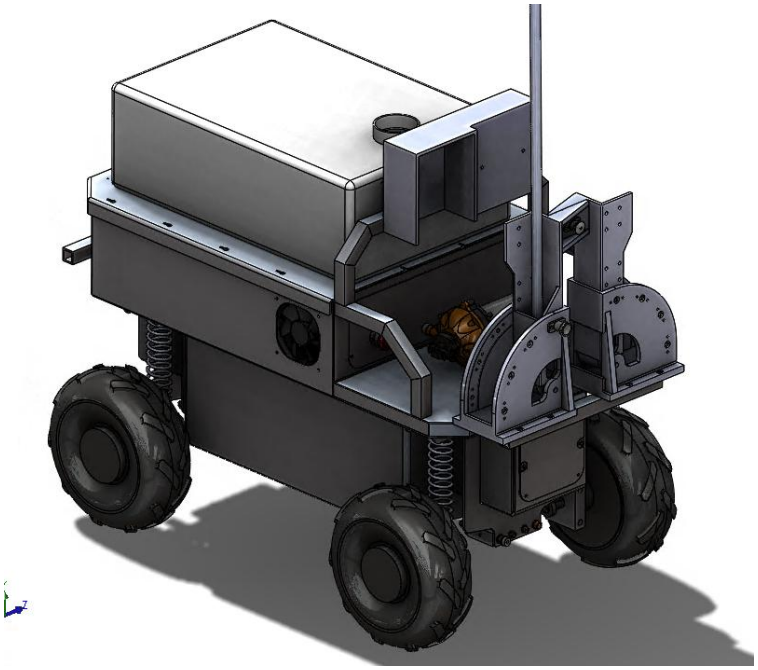
Table 2.2. Decker-03 Hub Motor Specifications (KK6 Fat Series).

Characteristic	Specification	Notes
Motor type	Geared Hub BLDC	10 pole pairs
Rated power / voltage	800 W / 48 V	24/36V compatible
Peak torque	96.15 N m	At stall current
Stall current	23.365 A	Datasheet max
ODrive current limit	35 A	Controller setting
No-load current	1.712 A	Baseline losses
No-load speed	247RPM	At 48 V
Gear ratio	5:1	Internal reduction
Wheel diameter	152.4 mm rim	300 mm outer diameter
Motor mass	7.1155 kg	Including tire

2.1.5 3D CAD and Finite Element Analysis

Following the design constraints laid upon, the robot design was created using a 3D CAD software Solidworks 2024. The robot design was an iterative process. The structural chassis started off with aluminum tube truss and suspensions started off with single swing-arm link connection. After multiple iterations and design philosophy of design for manufacturing and design for strength, a sheet metal chassis, with closed loop kinematic double wishbone suspension was finalized.

Figure 2.7. Isometric view of robot CAD assembly in Solidworks



Finite element analysis on the structurally most important part, wheel swing arm was conducted using Solidworks simulation. SolidWorks Simulation could not reliably mesh this assembly below approximately 2 mm, and a uniform 3 mm tetrahedral element size was adopted for the final analysis. At the coarse mesh limit, unnaturally high stress artefacts appeared at components such as bushings and washers, which made the assembly peak stress and factor of safety (FOS) vary and read inconsistently. To avoid simulation technical bottleneck, the bushings were removed and pin joint was used for the bolts of the joints, with same material characteristics as the actual bolt. All of the components except joints and fasteners are made of Al 6061 and the bolts are standard SS 304 steel. The swing arm is particularly important to perform FEA analysis because of its compact and load bearing nature. The link arms are the most susceptible components to experience any failure and hence are required to be made with prior estimation of their strength. To know this, the simulation was done in two scenarios. One, simple dead weight carrying Static analysis and the other being a scenario at the impact after a freefall of 0.61 m. Suspension effects are taken into consideration using the deceleration due to

suspension to simulate a more realistic impact. The results from both cases are shown in the figures below: Simulation parameters:

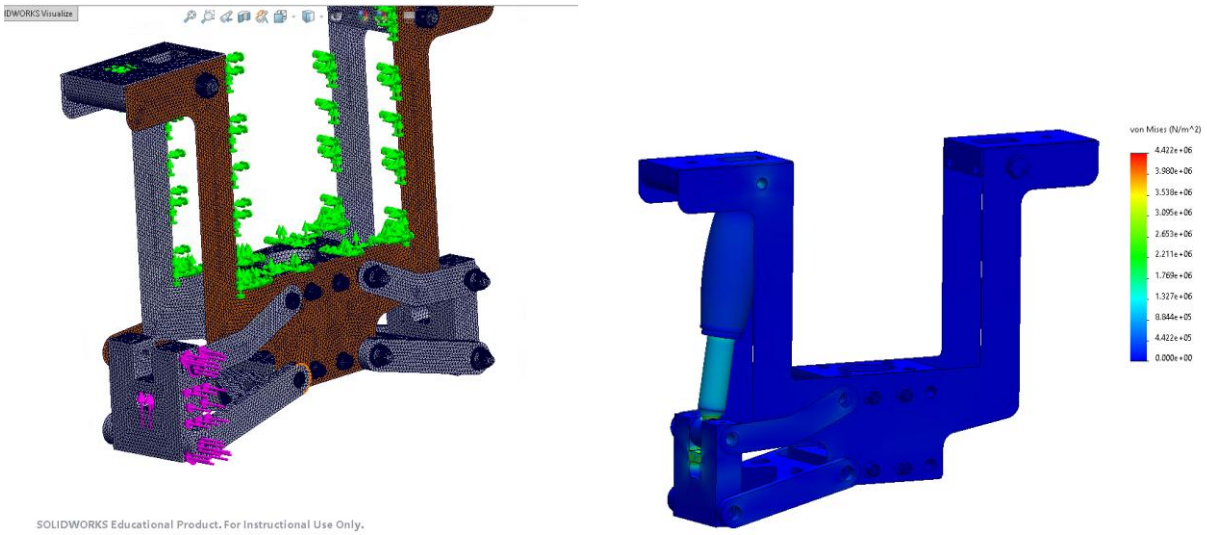
Table 2.3. Finite element analysis simulation parameters.

Parameter	Value
Analysis software	SolidWorks Simulation
Component analyzed	Wheel swing arm (link arm)
Mesh element size	3 mm (uniform)
Arm / link material	Al 6061-T6 (yield strength 276 MPa)
Bolt / pin joints	SS 304 steel (yield strength 215 MPa), modeled as pin joints
Load case 1	Static dead weight
Load case 2	0.61-m free-fall impact (suspension-decelerated)

The simulation was done for two conditions:

1. Dead weight static load bearing simulation

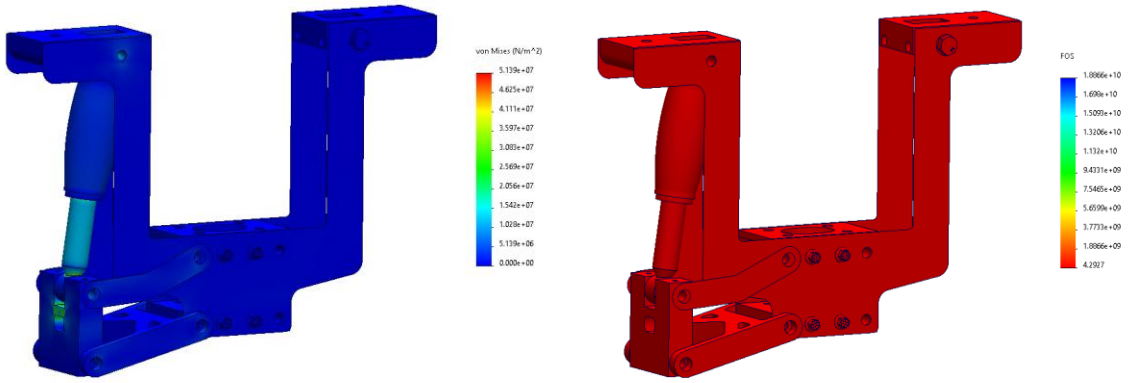
Figure 2.8. Dead-weight static load case: meshed swing-arm assembly with fixtures and loads (left) and resulting von Mises stress (right).



Static case. The dead-weight load per corner is the gravitational load $F = Mg$.

Impact case (0.61 m free fall). The impact scenario and its resulting von Mises stress field are shown below.

Figure 2.9. Two-foot free-fall impact load case: von Mises stress (left) and factor of safety (right).



The swing arm carries a total supported mass of 125 kg (robot) plus 100 kg (payload), i.e. 225 kg, which is 56.25 kg per corner across the four suspension units.

Impact velocity at contact. Falling from rest through h = 2 ft = 0.6096 m,

$$v^2 = v_0^2 + 2gh, \quad v_0 = 0 \;\Rightarrow\; v = \sqrt{2gh} = 3.458 \text{ m/s}.$$

Deceleration during the suspension stop. Bringing the corner to rest over the suspension stopping travel d,

$$v_f^2 = v^2 - 2ad, \quad v_f = 0 \;\Rightarrow\; v^2 = 2ad.$$

g-scaling factor. Equating the free-fall and stopping expressions, $2gh = 2ad$, gives

$$a = g\left(\frac{h}{d}\right),$$

where h/d is a dimensionless g-scaling factor. Evaluating this ratio yields a factor of 10.16, so a = 10.16 g.

Final applied load. Summing the inertial and gravitational contributions per corner,

$$F = ma + mg = m\,g\,(1 + 10.16) = 56.25\,g\,(11.16) \approx 5600 \text{ N}.$$

This per-corner impact load of approximately 5600 N (as recorded in the design notebook) was applied as the dynamic load case in the FEA; the static case applies the dead-

weight load $F = Mg$. The resulting von Mises stress and factor-of-safety distributions for both load cases are shown in the figures above.

Under the static dead-weight case, the swing arm is nominally unstressed: the peak von Mises stress is only about 4.4 MPa, concentrated at the lower suspension link. Under the 0.61-m free-fall impact case, the peak von Mises stress rises to approximately 51 MPa, localized at the lower control-arm pin and coilover mount, which is still far below the 276 MPa yield strength of Al 6061-T6. A factor-of-safety analysis was carried out for the 0.61-m free-fall impact case only, since it governs the design; the resulting plot is uniformly red at its lower bound, indicating a minimum factor of safety of 4.27 across the arm under this worst-case impact load. The swing arm therefore withstands the impact loading with a comfortable safety margin and no localized yielding.

2.2 Controller Model

2.2.1 Symbols

Let v, ω denote body linear and angular velocity; L the track width; r the wheel radius; $\omega_{m,i}$ the encoder velocity of motor $i \in \{0, 1, 2, 3\}$; R the minimum-radius parameter; $v_{\max}$, $\omega_{\max}$ the linear and angular velocity limits; and $a_{\max}$, $j_{\max}$ the per-axis acceleration and jerk limits. Motors {0, 1} form the left side and {2, 3} the right side; the right side is mirror-mounted, so a positive encoder reading on the right corresponds to reverse chassis motion.

2.2.2 Forward Kinematics

Per-side wheel linear velocity is selected by the faster wheel:

$$v_L = r\ \mathrm{mx}\big(\omega_{m,0},\ \omega_{m,1}\big), \qquad v_R = -r\ \mathrm{mx}\big(\omega_{m,2},\ \omega_{m,3}\big),$$

where mx(a, b) returns whichever argument has the larger magnitude, with its sign.

The body twist follows as

$$v = \frac{v_R + v_L}{2}, \qquad \omega = \frac{v_R - v_L}{L}.$$

The negative signs reconcile the mirror convention; the per-side max acts as a slip-rejection heuristic.

2.2.3 Command Pipeline

A reference (v_d, ω_d) is shaped in three stages.

Stage A, Saturation

The reference is first clipped to the platform's velocity limits:

$$v_d \leftarrow \text{clip}(v_d,\ -v_{\max},\ v_{\max}), \qquad \omega_d \leftarrow \text{clip}(\omega_d,\ -\omega_{\max},\ \omega_{\max}).$$

Stage B, Acceleration and jerk shaping

Given the latest measured velocity v_k, previous acceleration a_{k-1}, and control period Δt:

$$a^* = \frac{v_d - v_k}{\Delta t}, \qquad a_k = \text{clip}(a^*,\ a_{k-1} - j_{\max}\Delta t,\ a_{k-1} + j_{\max}\Delta t),$$

$$a_k \leftarrow \text{clip}(a_k,\ -a_{\max},\ a_{\max}), \qquad v_{k+1} = v_k + a_k\,\Delta t.$$

Applied independently to the linear and angular axes.

Stage C, Minimum-radius (bowtie) constraint

The feasible set is

$$\mathcal{F} = \{(v, \omega) :\ |v| \geq R\,|\omega|\},$$

bounded by $v = \pm R\omega$. If (v_d, ω_d) $\notin \mathcal{F}$, project orthogonally onto each boundary $v = sR\omega$, $s \in \{+1, -1\}$,

$$(v_s,\ \omega_s) = \frac{sR\,v_d + \omega_d}{R^2 + 1}\,(sR,\ 1),$$

and select the projection with the smaller squared distance

$$s^* = \arg\min_{s \in \{+1,-1\}} [(v_d - v_s)^2 + (\omega_d - \omega_s)^2], \qquad (v, \omega) = (v_{s^*},\ \omega_{s^*}).$$

2.2.4 Inverse Kinematics

The shaped twist (v, ω) maps to wheel angular velocities:

$$\omega_L = \frac{v - \omega d/2}{r}, \qquad \omega_R = \frac{v + \omega d/2}{r}.$$

A velocity-proportional torque feedforward is applied:

$$\tau_{ff,i} = k_{ff}\,\omega_i, \qquad i \in \{L, R\},$$

with feedforward gain k_{ff}.

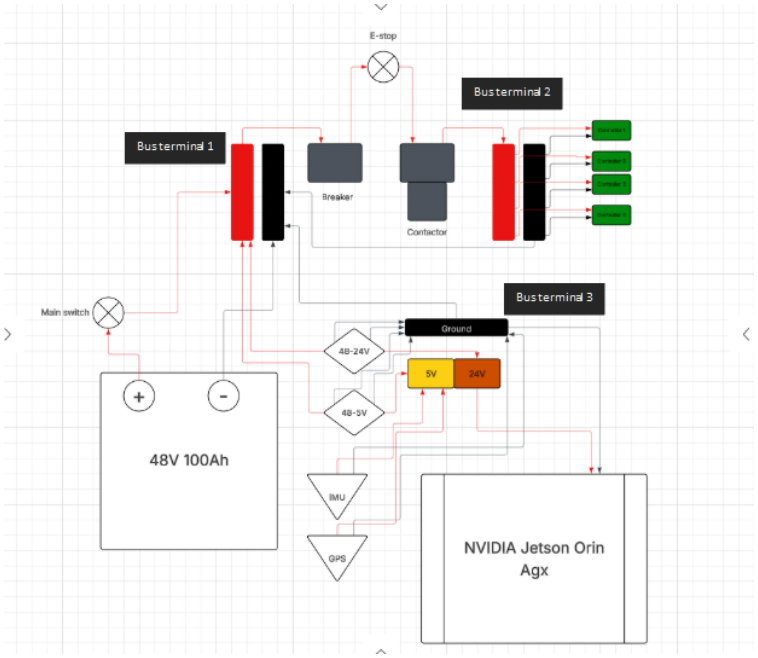
where ω_L, ω_R are the left and right wheel angular velocities (rad/s), d = 0.5334 m is the effective track width, and r = 0.1356 m is the effective wheel rolling radius derived from URDF inertia tensors. This inverse kinematic transformation is executed by the ROS2 skid-steer controller package, which publishes individual wheel velocity commands to each ODrive controller via CAN bus. Velocity feedback is provided by Hall effect sensors embedded in each motor hub.

2.3 Electrical Architecture and Power System and Sensors

2.3.1 Electrical Architecture and Power

The power system is centered on a 48 V, 100 Ah $LiFePO_4$ (DC House) battery pack providing 4.8 kW h of usable energy. $LiFePO_4$ chemistry was selected over conventional Li-ion for its thermal stability across agricultural field operating temperatures and its rated 4000-cycle service life. The battery is 520mm x 265mm x 290mm, weighing in at 37.5 Kgs. The total operational system mass, including battery, compute, sprayer subsystem (without tank and liquid), and mounting hardware not captured in the CAD model, is 125 kg.

Figure 2.10. Schematic diagram of electrical system



Power is distributed through a set of DC-DC buck converters providing regulated 24V, 12V, and 5V rails for the compute, sensing, and communication subsystems. The safety circuit includes a manually accessible main disconnect, a circuit breaker for overcurrent protection, and a hardwired E-stop that directly de-energizes all four ODrive motor controllers without passing through the compute stack, ensuring immediate halt capability regardless of software state. CAN bus communication connects the Jetson Orin AGX to all four ODrive S1 controllers. CAN was selected for its noise immunity in electrically demanding motor drive environments and its deterministic message delivery, which is important for maintaining consistent velocity command latency.

2.3.2 Compute and Software Stack

The NVIDIA Jetson Orin AGX serves as the central computer and communication hub. Its GPU architecture supports parallel processing for future perception and AI workloads, while its fanless design is appropriate for the dusty, high-humidity conditions of agricultural field

operation. The platform runs Ubuntu 22.04 LTS (Canonical Ltd., London, UK) with the ROS 2 Humble Hawksbill middleware stack (Open Robotics, Mountain View, CA, USA).

The complete command pipeline from operator input to wheel actuation is as follows: the `joy_node` receives joystick inputs from the radio controller and publishes to `/joy`; the `teleop_twist_joy` node converts these to `geometry_msgs/Twist` messages on `/cmd_vel`; the skid-steer controller applies the inverse kinematic transformation of the wheel angular-velocity equations and publishes individual wheel velocity targets via CAN to the four ODrive S1 controllers. Wheel odometry from the Hall effect sensors follows the reverse path back to the `joint_state_publisher`, which aggregates encoder positions and velocities and broadcasts them to the ROS2 TF2 tree.

2.3.3 Perception and Sensor Suite

Navigation in the GNSS-denied under-canopy environment relies on a multi-modal sensor suite. The primary perception layer consists of the Livox Mid-360 LiDAR (Livox Technology, Shenzhen, China), which operate in combination to map the 0.76-m row spacing and detect obstacles in real time. Two OAK-D SR stereo cameras (Luxonis Holding Corp., Denver, CO, USA), facing left and right into the crop rows, are used for short-range aphid detection. Inertial data from the onboard IMU and wheel velocity from the Hall effect encoders provide continuous orientation and slip estimation for terrain compensation. The RTK-GNSS module provides sub-centimeter global positioning in open headland areas; the system transitions to local relative positioning upon canopy entry. Sensor specifications are summarized in Table 2.4.

Table 2.4. Decker-03 Perception and Navigation Sensor Suite.

Sensor Type	Hardware	Function
Visual perception	Two OAK-D SR	Short-range depth-aware imaging for aphid detection (two cameras, facing left and right)
Spatial mapping	Livox Mid-360 LiDAR	High-resolution point cloud for narrow row navigation
Inertial odometry	YahBoom 10-DOF IMU	Orientation and angular velocity for terrain compensation
Locomotion feedback	Hall effect encoders	Wheel velocity feedback and odometry at 100 Hz
Global positioning	RTK-GNSS module	Centimeter-level positioning in open headlands

2.4 Digital Twin Development

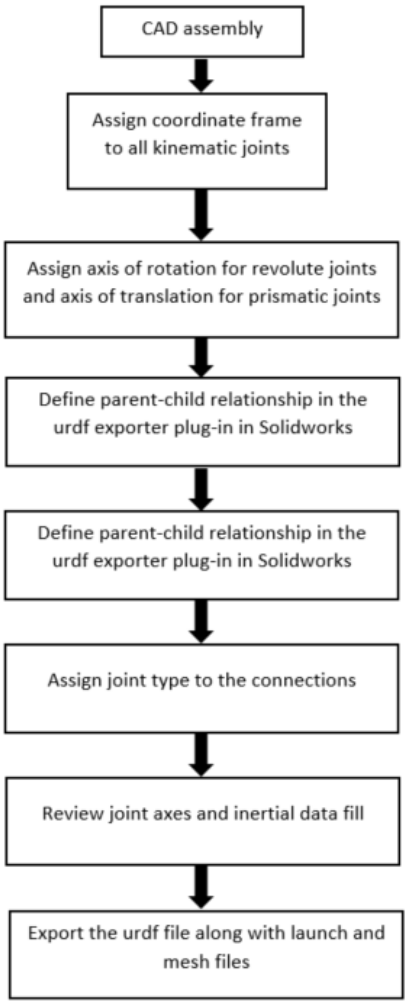
2.4.1 CAD - URDF Generation

The physical platform was modeled in SolidWorks (Dassault Systèmes SolidWorks Corp., Waltham, MA, USA) using parametric design with standard component libraries wherever available, limiting custom-fabricated parts to those with no viable COTS equivalent. This approach reduces model uncertainty, standard components have well-characterized mass and inertia properties available from manufacturer data. The SolidWorks assembly was prepared for URDF export using the `sw_urdf_exporter` plugin. Some known limitations of the URDF plugin is that it sometimes resets inertial information and mass properties of components of the assembly. Proper review and verification is recommended before using the urdf file as it is.

This process required populating the assembly with coordinate systems and joint axes at every kinematic joint prior to export. Revolute joints were defined at all four wheel hubs and at the suspension wishbone pivots; prismatic joints were defined for the coilover travel range; static reference frames were established at the LiDAR, OAK-D, and IMU mounting locations to ensure correct sensor transform offsets in the ROS2 TF2 tree. The `base_link` frame was placed at the

geometric center of the upper deck, serving as the root of the kinematic tree. The export pipeline is summarized in Fig. 2.11.

Figure 2.11. Flowchart of the methodology for generating the URDF from the SolidWorks CAD assembly

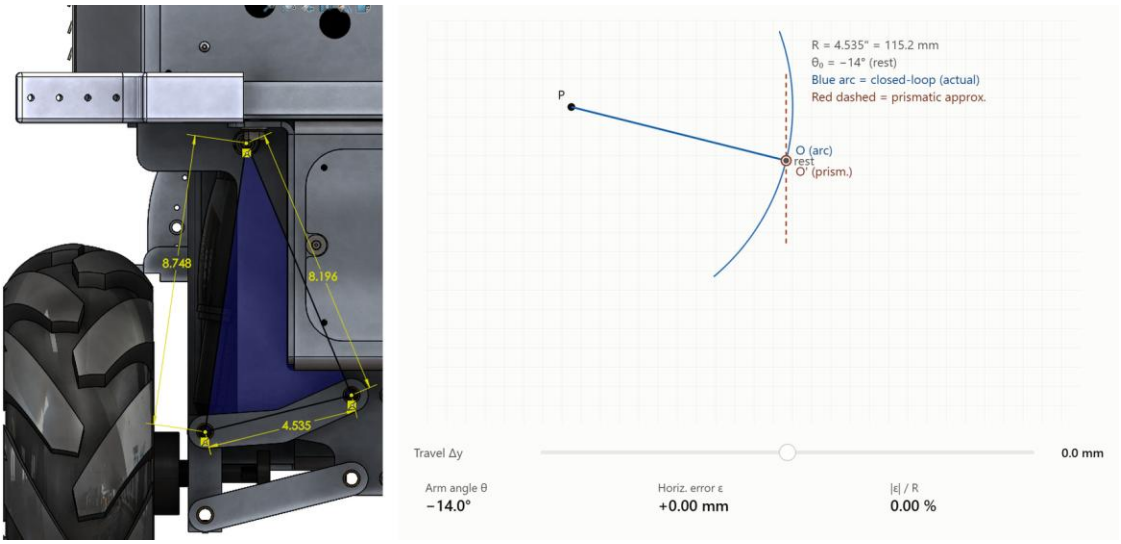


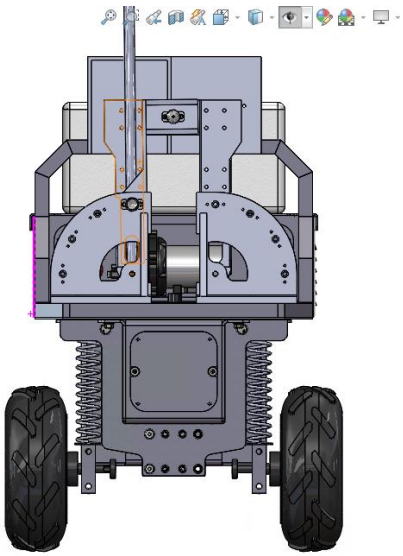
2.4.2 Closed-Loop Suspension Approximation

A known limitation of the URDF specification is its inability to represent closed-loop kinematic chains. The double-wishbone suspension is inherently a four-bar closed-loop mechanism. To resolve this, the suspension was modeled as an open-loop kinematic chain. The double wishbone has a large radius of curvature for the suspension travel, meaning it exhibits

predominantly vertical motion with minimal horizontal displacement. The lower wishbone sits at approximately 14° below horizontal at rest and swings on an effective radius of 4.535 in (115.2 mm) about its chassis pivot P. The horizontal deviation ε between the true arc path (R cos θ, R sin θ) and the vertical prismatic approximation through the rest point is $\varepsilon(\Delta y) = R\cos\theta - R\cos 14°$, with $\sin\theta = \Delta y / R - \sin 14°$ for a vertical travel Δy. Over the physical operating range, bump up to +25.4 mm and droop limited to −5 mm by the spring preload at the 125 kg ride-height load, the deviation stays within +3.4 mm at full bump and −1.4 mm at maximum droop, under 3 % of the swing-arm length. Since 3.4 mm is a small lateral distance relative to the chassis dimensions, the open-loop prismatic substitution is retained as a decent alternative to the closed-loop double-wishbone mechanism, which cannot be represented directly in URDF and therefore cannot be used for simulation.

Figure 2.12. Closed loop and Open loop kinematic chain wheel arm SolidWorks CAD assembly





2.4.3 Simulation Environment Configuration

The URDF was imported into NVIDIA Isaac Sim 5.0 using the URDF Importer extension, which converts the XML kinematic description to the Universal Scene Description (USD) format used by the PhysX 5 engine. The base link was designated as the Articulation Root in the Stage Tree, enabling the PhysX solver to compute internal joint forces and propagate kinematic state updates throughout the tree at each simulation timestep. The base was set to movable to allow the robot to respond to gravity and terrain geometry.

2.4.4 Physics Parameter Derivation

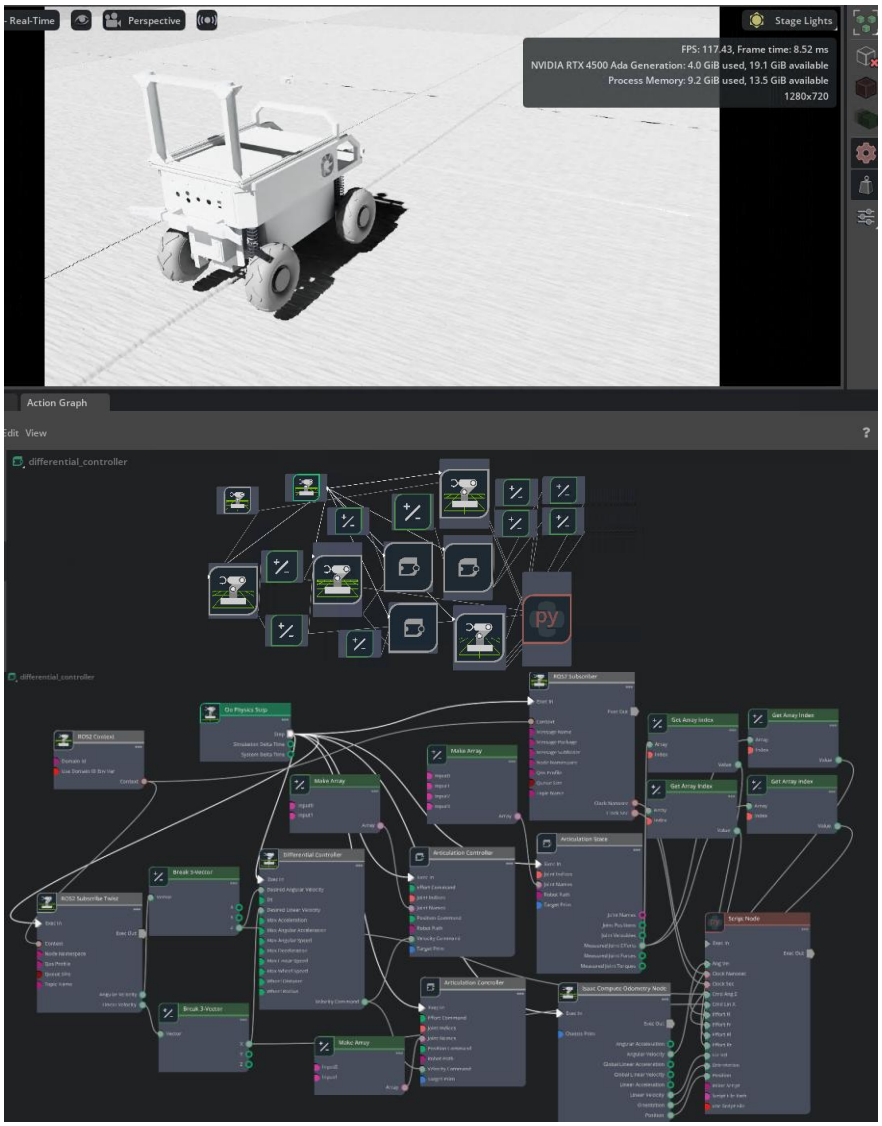
Parameters are set from measurements and datasheets rather than fitted; sim-to-real validation is future work. All parameters are summarized in Table 2.5.

Table 2.5. Isaac Sim Physics Parameters and Their Sources.

Parameter	Value	Source
Physics timestep	200 Hz	2× sensor rate
Solver position iters.	32	Contact stability
Solver velocity iters.	8	Contact stability
Joint drive stiffness	0.0	Velocity drive mode

Parameter	Value	Source
Joint drive damping	40 N m s/rad	K_t^2/R (Section 2.5)
Max joint force	96.15 N m	Motor manufacturer
Static friction	0.75	Literature (rubber/concrete)
Dynamic friction	0.60	Literature (rubber/concrete)
Restitution	0.02	Concrete (near-inelastic)
Body mass	125 kg	Physical measurement
Wheel rolling radius	0.1356 m	URDF inertia tensor

Figure 2.13. Isaac SIM controller graph network



2.4.4.1 Suspension Joint Parameterization

The coilover suspension units are rated at 15.76 kN/m, yielding a spring stiffness of:

At the nominal operational mass of 125 kg, the predicted static sag per corner is:

$$\delta = \frac{mg}{4k} = \frac{125 \times 9.81}{4 \times 15{,}761} \approx 0.0195 \text{ m} = 19.5 \text{ mm}.$$

Suspension joint travel limits were set to ±30 mm based on physical inspection of the coilover assembly.

Drive damping was set to 500 N s m^{-1} consistent with the coilover damper specification. It is noted that the SolidWorks URDF exporter does not support a spring stiffness field; the stiffness was therefore applied directly in the Isaac Sim Drive component after import, as the `dynamics` element damping alone does not replicate spring behavior in PhysX.

2.4.4.2 Contact Material Properties

Surface friction coefficients were estimated from concrete-rubber contact literature values (*Friction Coefficients for Rubber on Concrete*, n.d.) and applied as a physics material on the ground plane. A static friction coefficient of 0.75 and dynamic friction coefficient of 0.60 were used, reflecting the brushed concrete test surface conditions. Restitution was set to 0.02 to model the near-inelastic wheel-ground contact characteristic of rubber tires on rigid concrete.

2.4.5 Ground Plane Modeling

The simulation environment uses a rigid, non-deformable ground plane procedurally generated to replicate the surface roughness characteristics of the hardened agricultural concrete test track. The surface height field is defined by a fractional Brownian motion (fBm) Perlin noise function:

$$Z(x, y, A) = \sum_{i=0}^{n-1} A \cdot p^i \cdot \text{noise}\left(f \cdot \lambda^i \cdot x,\ f \cdot \lambda^i \cdot y\right) \quad (2.1)$$

where A is the amplitude, p is the persistence, f is the base frequency, λ is the lacunarity, and n is the octave count. The amplitude A was clamped to 5 mm to match the measured roughness profile of the concrete test surface; all other parameters were held constant across all trials. Parameters were set to n = 6, p = 0.5, f = 1.0, lambda = 2.0, consistent with standard fBm terrain generation for rigid surface simulation.

The use of a rigid ground plane is a deliberate scoping decision. By excluding soil deformability, the digital twin isolates drivetrain dynamics, suspension geometry, and motor behavior from the additional uncertainty of terra-mechanics parameters that are difficult to measure in the field. This establishes a controlled baseline; deformable terrain modeling using soft-body or particle-based methods is identified as future work.

2.4.6 Motor Torque Model

A motor torque model was added so that simulated joint torque and physical motor torque can be compared on a common basis. The physical platform records armature current; the Isaac Sim PhysX solver outputs an idealized joint torque.

The physical torque reconstructed from current is given by:

$$\tau_{real} = K_t(I_{real} - I_0)$$

with K_t = 4.41 N·m/A the motor torque constant and I_0 = 1.712 A the no-load current; their identification is detailed in Section 2.5. The onboard ODrive torque_est signal was not used, as it was found to be physically inconsistent.

In Isaac Sim, the PhysX velocity drive computes the required joint effort as:

$$\tau_{sim} = K_d(\dot{q}_{des} - \dot{q}) + \tau_{ff},$$

with K_d the drive damping gain acting as velocity stiffness. Because motor back-EMF reduces available torque at higher speed, a dynamic torque-speed envelope clamp is applied to prevent unrealizable solver output:

$$\tau_{\max}(\dot{q}) = \tau_{peak}\left(1 - \frac{|\dot{q}|}{\omega_0}\right), \qquad \tau_{sim} \leftarrow \text{clip}(\tau_{sim},\ -\tau_{\max},\ \tau_{\max}),$$

where τ_{peak} = 96.15 N m is the datasheet peak torque and ω_0 = 247 RPM ≈ 25.9 rad/s is the no-load speed, so available torque falls linearly to zero at no-load speed.

2.4.7 ROS 2 Bridge and Command Injection

Communication between Isaac Sim and the ROS 2 ecosystem was established via the `omni.isaac``.ros2_bridge` extension. A differential-drive articulation controller was implemented within the Isaac Sim ActionGraph. The ROS 2 twist subscriber node receives `geometry_msgs/Twist` messages on `/cmd_vel`; the differential controller decomposes the twist into per-wheel angular velocity targets using the wheel angular-velocity equations and passes them to the PhysX articulation controller. Because the default differential drive articulation controller supports only two joints, a duplicated controller was added for the rear wheel pair.

Command inputs to the simulation were sourced exclusively from rosbags recorded during physical robot trials, replayed using the `ros2 bag play` tool with the `--clock` flag to maintain temporal fidelity. To resolve the QoS mismatch between the rosbag publisher (Best Effort) and the Isaac Sim ROS 2 bridge (Reliable), a QoS override profile was applied at playback. Simulation time was synchronized to the rosbag playback clock throughout each trial.

Joint effort can be extracted from the simulation using Read Effort nodes in the ActionGraph, which query the PhysX solver for the instantaneous torque at each wheel revolute joint at each simulation timestep.

2.4.7.1 Rosbag Replay Procedure

Real-world robot behavior is transferred to the simulation by replaying the commands recorded on the physical robot, so that the digital twin is driven by exactly the inputs the real robot received. The procedure is:

1. **Record on the robot.** During a physical trial, record the velocity command (`/cmd_vel`) together with the robot's measured response wheel joint states and ODrive phase currents with `ros2 bag record`.
2. **Prepare the simulation.** Load the corrected digital twin in Isaac Sim, start the ROS 2 bridge, and configure the ActionGraph to take its time from the ROS `/clock` topic, so that the simulation and the replayed commands share one time base.
3. **Replay the commands.** Play the bag with `ros2 bag play --clock`, publishing only the command topic, and apply a QoS override that matches the bag's best-effort publisher to the bridge's reliable subscriber.
4. **Drive the twin.** The ActionGraph's twist subscriber receives the replayed commands, the differential controllers convert them into wheel velocity targets, and PhysX advances the articulation.
5. **Record the simulated response.** Publish the simulated wheel joint states and the joint efforts from the Read Effort nodes back to ROS 2 and record them with `ros2 bag record`.
6. **Compare.** Align the real and simulated recordings by timestamp and compare wheel velocities directly, and wheel torques using the torque reconstructed from current, $\tau_{real} = K_t(I_{real} - I_0)$.

This procedure provides the basis for quantitative sim-to-real validation of the twin, which is left to future work.

2.5 Motor Characterization for Digital Twin Configuration

Accurate torque data is fundamental to the dynamic validation of the digital twin. In practice, dedicated torque sensors are often unreliable due to mechanical coupling noise, calibration drift, and sensor placement constraints. Furthermore, the onboard ODrive `torque_est` signal was found to give out an incorrectly configured internal torque constant, rendering it physically inconsistent for this study. This section reconstructs the wheel torque from motor phase-current measurements using the datasheet-derived motor torque constant, so that the physical motor torque can be compared with the simulated joint torque on a common basis.

2.5.1 Motor Specifications

The drivetrain uses four UU Motor Technology KK6 Fat Series hub motors in a geared hub BLDC, 5:1 internal reduction configuration. Key datasheet parameters extracted for this study are summarized in Table 2.6.

Table 2.6. KK6 Fat Series motor datasheet parameters.

Parameter	Value	Condition
Rated voltage	48.0 V	Nominal
Rated current	17.0 A	Nominal
No-load torque	0.78 N m	247RPM, 1.712 A
Maximum torque	96.15 N m	16RPM, 23.365 A
Max. efficiency point	22.64 N m, 81.3%	217RPM, 12.831 A
Maximum output power	876.04 W	190RPM, 23.407 A
Number of pole pairs	10	Motor design

2.5.2 Torque Constant Identification (K_t)

For a BLDC motor operating in the linear regime, the electromagnetic torque is related to phase current by the torque constant K_t (N m/A). K_t is derived from the no-load and peak-torque datasheet operating points, with no-load losses subtracted to exclude iron losses and bearing friction from the electromagnetic contribution:

$$\tau_{wheel} = K_t(I - I_0),$$

with

$$K_t = \frac{\tau_{peak} - \tau_0}{I_{stall} - I_0} = \frac{96.15 - 0.78}{23.365 - 1.712} = 4.41 \text{ N m/A}.$$

2.5.3 Joint Drive Parameterization in Isaac Sim

The identified motor parameters are used to configure the PhysX `ArticulationDrive` for each wheel joint. The drive damping coefficient is set to match the motor's electrical back-EMF braking characteristic, $K_d = K_t^2/R_{ph}$, where R_{ph} is the motor phase resistance:

Joint drive stiffness is set to zero to enforce pure velocity tracking, and maximum force is capped at the operational peak torque provided by the manufacturer:

$$K_p = 0, \qquad \tau_{\max} = \tau_{peak} = 96.15 \text{ N m}.$$

A summary of all identified parameters is provided in Table 2.7.

Table 2.7. Summary of Identified Motor Parameters.

Parameter	Symbol	Value	Source
Torque constant	K_t	4.41 N m/A	Datasheet
No-load current	I_0	1.712 A	Datasheet
Joint drive damping	K_d	40 N m s/rad	
Max. joint force	τ_{max}	96.15 N m	Datasheet

2.6 Conclusions

This chapter presented the design, fabrication, and digital twin development of the Decker-03 skid-steer robotic platform, targeted at 0.76-m under-canopy row crop applications. The principal contribution of this chapter is the Decker-03 under-canopy platform itself. A supporting methodological contribution is the workflow for constructing a digital twin of the platform in NVIDIA Isaac Sim, which is positioned as a continuing area of development rather than a validated simulation of the robot's dynamics.

Together, these steps provide a reproducible pipeline from CAD and datasheets to a working Isaac Sim twin driven by real robot logs. Sim-to-real validation is left to future work. The digital twin opens the platform to rapid prototyping in simulation. Immediate avenues include: row-navigation controllers developed and tested on procedurally generated row fields prior to field deployment; aphid detection and spray-triggering software iterated in simulation before being flashed to the robot; and safety tests and control-policy development carried out in simulation, where failure modes can be explored without risk to hardware or crop. Sim-to-real validation of the twin itself remains a prerequisite for these uses.

Chapter 3 - Real-Time Aphid Detection and Targeted Spot Spraying

3.1 Introduction

Aphids are among the most economically damaging pests in corn and sorghum, and conventional control relies on broad spraying of insecticide across the whole field. Because aphid infestations are typically patchy, broadcast application wastes chemical, raises production cost, and increases environmental load. *Spot spraying* applies insecticide only where pests are detected, and it can substantially reduce chemical use when detection and actuation are both fast and spatially accurate.

This chapter describes the design, implementation, and iterative refinement of a detect-and-spray system on the Decker-03 skid-steer field robot (Chapter 2). The system detects aphids in real time from two side-mounted stereo cameras using a transformer-based object detector accelerated with TensorRT, converts those detections into spray commands, either for a fixed duration or at the position where each aphid was seen, and actuates spray nozzles through one of two actuation back ends: a commercial TeeJet sprayer controller over CAN bus, and a custom multi-nozzle solenoid driver.

The chapter first describes the system architecture, then the detector, the two actuation back ends, and the position-based trigger, and closes with the verification of the integrated system. The contribution is the design and verification of the system; field evaluation of spray efficacy is left to future work. The system was developed in four stages (Table 3.1)

Table 3.1. Staged development of the spot-spraying system.

Stage	Actuation	Triggering strategy	Spatial resolution
1	TeeJet ECU over CAN	Per-camera, fixed spray duration	2 nozzle groups
2	MCP23017 solenoid driver over I^2C (prototype)	Image split into vertical zones	6 nozzles

Stage	Actuation	Triggering strategy	Spatial resolution
3	MCP23017 solenoid driver via USB–I^2C bridge	Per-camera boom, fixed spray duration	2 booms × 3 nozzles
4	As stage 3	Position-based triggering: detections localized with stereo depth and tracked to the boom with `tf2` and EKF odometry	2 booms × 3 nozzles

3.2 System Overview

The system's perception and actuation software is isolated from the robot's base vehicle control, so that the spray pipeline can be rebuilt and redeployed without disturbing motor control, teleoperation, or navigation.

3.2.1 Robot platform and compute

Table 3.2. Robot platform and compute hardware.

Component	Specification
Robot base	Skid-steer platform driven by ODrive motor controllers over CAN; RTK-capable u-blox GNSS receivers (front and rear). This is the Decker-03 platform of Chapter 2: 0.61 m track width (for 0.76-m rows), 1.02 m bounding length, 125 kg operational mass, 1–2 m/s traverse.
Main computer	AAEON BOXER-8640AI fanless embedded PC with an NVIDIA Jetson AGX Orin module (NVIDIA Corp., Santa Clara, CA, USA; 32 GB, 8-core Arm Cortex-A78AE
Operating system	NVIDIA JetPack / L4T R36.4.3, Linux kernel 5.15.148-tegra, Ubuntu 22.04
Middleware	ROS 2 Humble, Fast DDS (`rmw_fastrtps_cpp`)
Containers	Hardware container (drive, IMU, state estimation, robot description) and perception container (cameras, detection, triggering, valve driver), built on the NVIDIA Isaac ROS image
Cameras	Two Luxonis OAK-D SR stereo cameras facing left and right into the crop rows; 640 × 400 color and aligned stereo depth at about 28 fps; mounted 1.17 m ahead of the spray boom
Inertial sensor	Yahboom 10-axis IMU over USB serial, about 100 Hz, fused with wheel odometry for state estimation

Software is split across two Docker containers that share the host network and exchange topics and transforms over Fast DDS. The hardware container carries drive control, the IMU driver, local state estimation, and the robot description (URDF); the perception container carries the camera drivers, the aphid detectors, the spray triggers, and the solenoid valve driver. This isolation lets the perception stack be rebuilt and redeployed without disturbing the vehicle's base control software.

3.2.2 End-to-end data flow

Each camera feeds its own detection node, which publishes a `Detection2DArray`. The two detection streams fan out to two independent actuation branches. The CAN branch runs `aphid_sprayer_bridge` into `sprayer_driver`, which emits CAN frames to the TeeJet ECU. The I^2C branch runs `aphid_boom_trigger` into `i2c_solenoid_driver`, which sends commands over USB serial to an Arduino Nano Every (Arduino S.r.l., Monza, Italy) that performs the I^2C writes to the MCP23017 solenoid-driver board (Microchip Technology Inc., Chandler, AZ, USA). The two branches publish on separate ROS topics (`/sprayer/nozzle_command` and `/sprayer/nozzle_command_i2c`), so they can run together or independently and a change to one never affects the other.

Within the solenoid branch, one of two interchangeable triggers converts detections into nozzle commands: the time-based boom trigger or the position-based trigger (Section 3.5), selected by the launch argument `spray_trigger:=boom|tf`. The resulting architecture is shown in the figure below. Images flow through detection to the trigger, while the hardware container supplies the robot's motion and the fixed sensor geometry that the position-based trigger needs to place each detection.

Figure 3.1. Software architecture of the solenoid spray path.

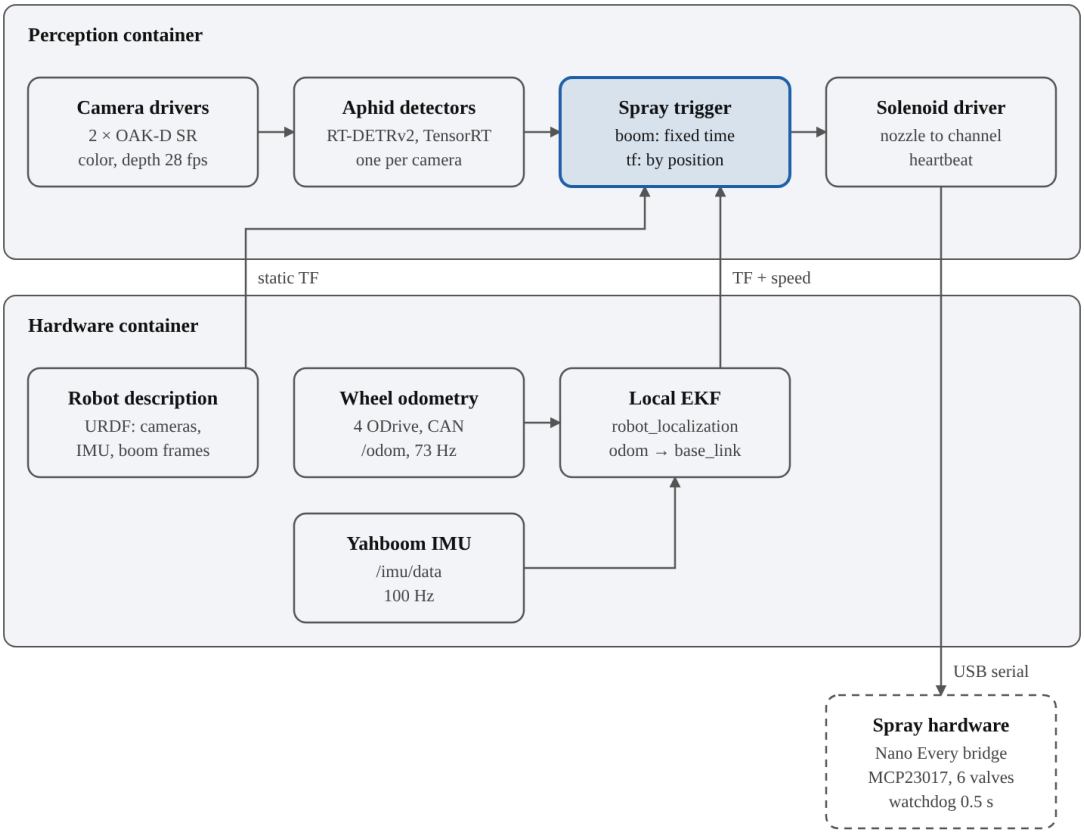


Table 3.3. Main ROS 2 topics of the spray pipeline.

Topic	Type	Publisher → subscriber	Rate
`/{left,right}/camera_node/right/image_raw`	`sensor_msgs/Image`, 640 × 400	Camera driver → detector	28–30 fps
`/{left,right}/camera_node/stereo/image_raw`	`sensor_msgs/Image`, 16-bit depth (mm)	Camera driver → position-based trigger	26–27 fps
`/{left,right}/aphid/detections`	`vision_msgs/Detection2DArray`	Detector → trigger	24 fps
`/odometry/local`	`nav_msgs/Odometry`	EKF → position-based trigger	92 Hz
`/tf`, `/tf_static`	`tf2_msgs/TFMessage`	EKF, robot description → position-based trigger	92 Hz / latched
`/sprayer/nozzle_command_i2c`	`sprayer_interfaces/NozzleCommand`	Trigger → solenoid driver	On change

The cameras sit 1.17 m ahead of the boom, which leaves about 1.17 s at 1 m/s between seeing an aphid and spraying it. The measured median detection latency of about 88-92 ms (Section 3.6.1) is well inside that budget.

3.3 Aphid Detection

The detector is a TensorRT-accelerated RT-DETRv2 model with a ResNet-101 backbone running on two side-facing stereo cameras.

3.3.1 Camera selection

A first prototype used an Arducam AR0234 sensor (Arducam Technology, Hong Kong, China) on a Power-over-Ethernet camera board, but driver failures, low throughput, and network fragility made it impractical. It was replaced with two Luxonis OAK-D SR USB stereo cameras, which run on the standard DepthAI ROS driver and provide the on-device stereo depth used for 3D localization (Section 3.5).

3.3.2 Model selection

YOLO-based detectors (YOLO26, at 640 -1024 px input) trained for the same single class produced maximum confidences of only 0.005–0.011 on live frames from the deployment camera, which was traced to a domain gap between close-up training images and the small, wide-field targets seen on the robot. A transformer-based RT-DETR model reached 0.20–0.28 with stable boxes on the same frames and was selected.

3.3.3 Final model

The deployed detector is a retrained successor of that first RT-DETR model: RT-DETRv2 with a ResNet-101 backbone (`aphid_r101`), trained on a single `aphid` class.

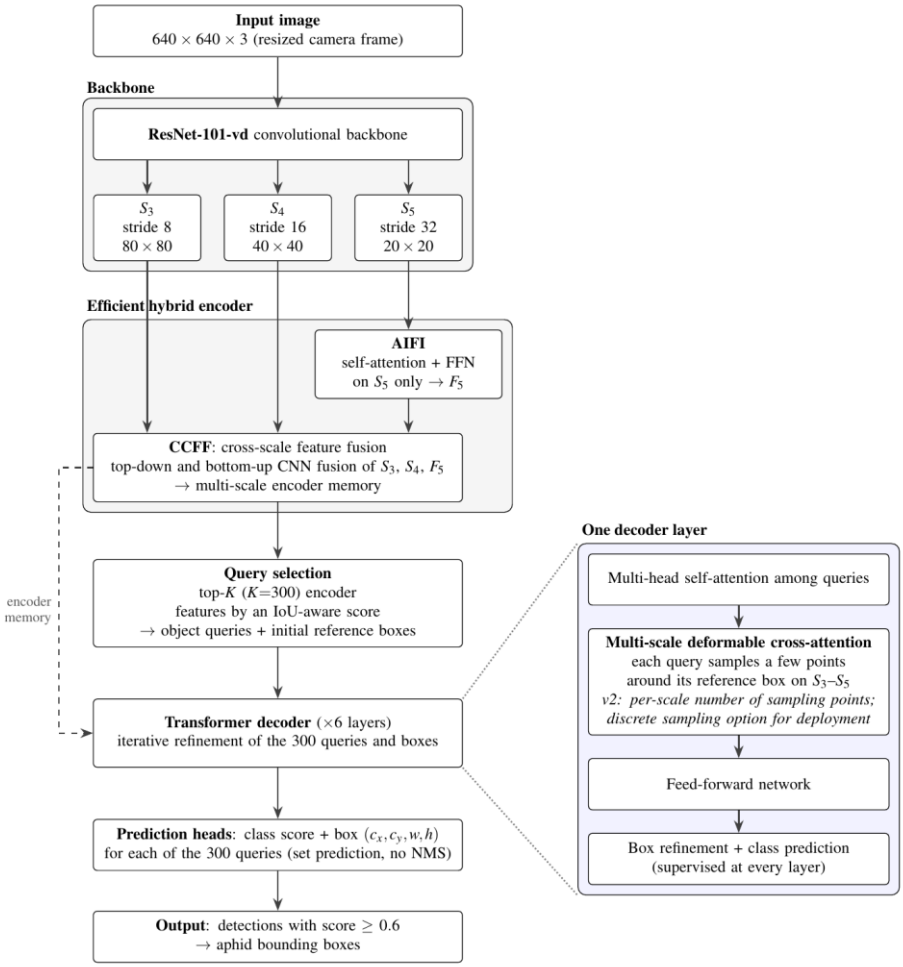
- Dataset: the detector was trained and evaluated on a single-class (`aphid`) dataset of 948 training and 237 validation images, each 640×640 px with at least one labelled box. The training split contains 27,027 aphid boxes (about 28 per image, up to 307) and the validation split 6,363 boxes (about 27 per image, up to 184). The median box is roughly 8×8 px, so the targets are very small objects. The aphid imagery was provided by Dr. Ivan Grijalva and Dr. Brian McCornack (Kansas State University).
- Training: both backbones were trained for 72 epochs with a batch size of 16 (train) / 32 (val) and identical augmentation, color jitter, zoom-out, IoU crop, and horizontal flip, on a single NVIDIA RTX 4500 Ada (24 GB) GPU under Windows and PyTorch. Training took about 4.7 h for the R101-vd model and about 4.9 h for R50-vd-m (roughly 4 min per epoch). The R50-vd-m configuration file now specifies 84 epochs, but the reported run used 72.
- Accuracy: COCO-style validation metrics are given in Table 3.4. The R101-vd backbone was selected over the lighter R50-vd-m variant, which scored lower at every AP threshold.
- Detection threshold: detections below a confidence of 0.6 are discarded in the field configuration, up from 0.25 for the first RT-DETR model.

Table 3.4. RT-DETRv2 validation metrics for the two evaluated backbones.

Metric (validation)	R101-vd (deployed)	R50-vd-m
Parameters	62.2 M	36.4 M
AP @0.50:0.95	0.232	0.205
AP @0.50	0.560	0.510
AP @0.75	0.157	0.126
AR @1 / @10 / @100	0.021 / 0.129 / 0.335	0.021 / 0.123 / 0.306
Final training loss	19.57	19.57

The heavier R101-vd backbone gains about 2.7 AP points at [0.50:0.95] and 5.0 at [0.50] over R50-vd-m for roughly 1.7× the parameters (62.2 M vs 36.4 M). All metrics are computed on the validation split at the 640×640 input resolution. The low absolute mAP@0.50:0.95 of 0.23, despite a moderate AP@0.50 of 0.56, is characteristic of very small objects: with a median box of about 8×8 px, a localization error of even one or two pixels shifts the predicted box across a large fraction of its area, so the stricter high-IoU thresholds are hard to satisfy even when the aphid is correctly detected.

Figure 3.2. RTDETR Architecture



3.3.4 Deployment with TensorRT

The model was deployed in two stages. First, **ONNX export**: the trained PyTorch model (PyTorch 2.11, CUDA 12.8) was exported to ONNX (opset 18). Post-processing is part of the exported graph. The model takes two inputs, the image tensor `images` (N ×3×640×640) and the original image sizes `orig_target_sizes` (N × 2), and produces three outputs over 300 object queries: `labels` (1 × 300), `boxes` (1 × 300 × 4), and `scores` (1 × 300). Because post-processing is included, boxes return already scaled to the original image, so the ROS node needs no decoding step.

Second, **TensorRT engine build**: the ONNX graph was compiled with the TensorRT 10 builder API (`export_trt.py`) with FP16 precision enabled, a 4 GB workspace, and an optimization profile fixed at batch size 1 and 640 × 640 input. The deployed engine (`aphid_r101_trt107.engine`) targets TensorRT 10.7, the version in the deployment container (with PyTorch 2.5 for NVIDIA Jetson).

Inference runs in a Python ROS 2 node (`aphid_detection/aphid_node`) using a TensorRT inference wrapper for RT-DETRv2, adapted from the RT-DETR reference implementation (Lyu et al.) with modifications by Hitbee-dev. Inputs and outputs stay on the GPU as PyTorch CUDA tensors. Per frame, the node converts the ROS image to OpenCV BGR (`cv_bridge`); resizes it to the 640 × 640 network input, converts it to RGB, and normalizes it to [0, 1]; runs inference and receives boxes, scores, and labels already rescaled to the original image size; drops detections below a confidence threshold (0.6 in the field launch configuration); and publishes a `vision_msgs/Detection2DArray` on `aphid/detections` together with an annotated image on `aphid/image_detections`, both relative to the camera's namespace.

One detector instance runs per camera (`/left` and `/right` namespaces). Each detector uses the right-hand sensor image of its OAK-D SR (`camera_node/right/image_raw`, 640 × 400 at about 28 fps).

- Inference latency: model inference takes about 24ms per frame on the AGX Orin with both detectors running; image conversion, resizing, and publishing the annotated image bring the image-received-to-detection time to about 48ms (median). Limiting the detector's input queue to a single frame proved essential: with a 10-frame queue, a detector slightly slower than the camera always works on frames about ten periods old, and median capture-to-detection latency was 374ms; with a one-frame queue it fell to about 88-92ms at the same throughput.

Running the engine inside the container required resolving a TensorRT version mismatch between the container and the host and an output-buffer allocation error.

3.3.5 Camera configuration

The OAK-D SR cameras run the DepthAI ROS driver with a stereo depth pipeline, with exposure capped at 3ms to limit motion blur on the moving robot, and detections are expressed in the camera's right optical frame. Inspection of the published camera information confirmed that the stereo depth image is registered to the same right sensor that the detector uses: the depth stream shares the detector's optical frame ID, resolution (640 × 400), and camera intrinsic, and is encoded as 16-bit unsigned millimeters. This alignment is a prerequisite for the 3D localization described in Section 3.5.

3.4 Spray Actuation

Two spray back ends were implemented. A commercial TeeJet CAN sprayer controller was built first and used as a reference for the spray behavior. A custom I^2C-driven solenoid board replaced

it as the primary actuator on the robot and is the actuation contribution of this chapter. The CAN back end is retained in the ROS 2 stack as a software fallback.

3.4.1 Commercial CAN sprayer: approach and limitations

The CAN sprayer driver uses the TeeJet sprayer ECU already on the robot's can1 bus. A C++ ROS 2 node (sprayer_driver) sends extended-ID CAN frames to the ECU (ID 0x18EFF8C0) using two message types: Set Target PWM (0x13) sets a duty cycle (compile-time constant, default 40 %), and Set Extended Boom Status (0x18) sets a bitmask of two nozzle groups (codes 0x15 and 0x20). Both messages are re-sent every 100ms as a keep-alive; the node publishes sprayer_interfaces/NozzleStatus for monitoring. A joystick node allows manual spraying.

A bridge node (aphid_sprayer_bridge) maps detections to CAN group commands: each camera controls one group, a group is switched on when a frame contains at least 3 detections, and it stays on for 0.762 s with a 1.0 s cooldown measured from spray start. The 0.762 s duration follows from the crop geometry at the nominal speed: three plants spaced 0.254 m apart span 0.762 m, which takes 0.762 s at 1 m/s.

Three characteristics of this back end limited its usefulness for the detect-and-spray goal:

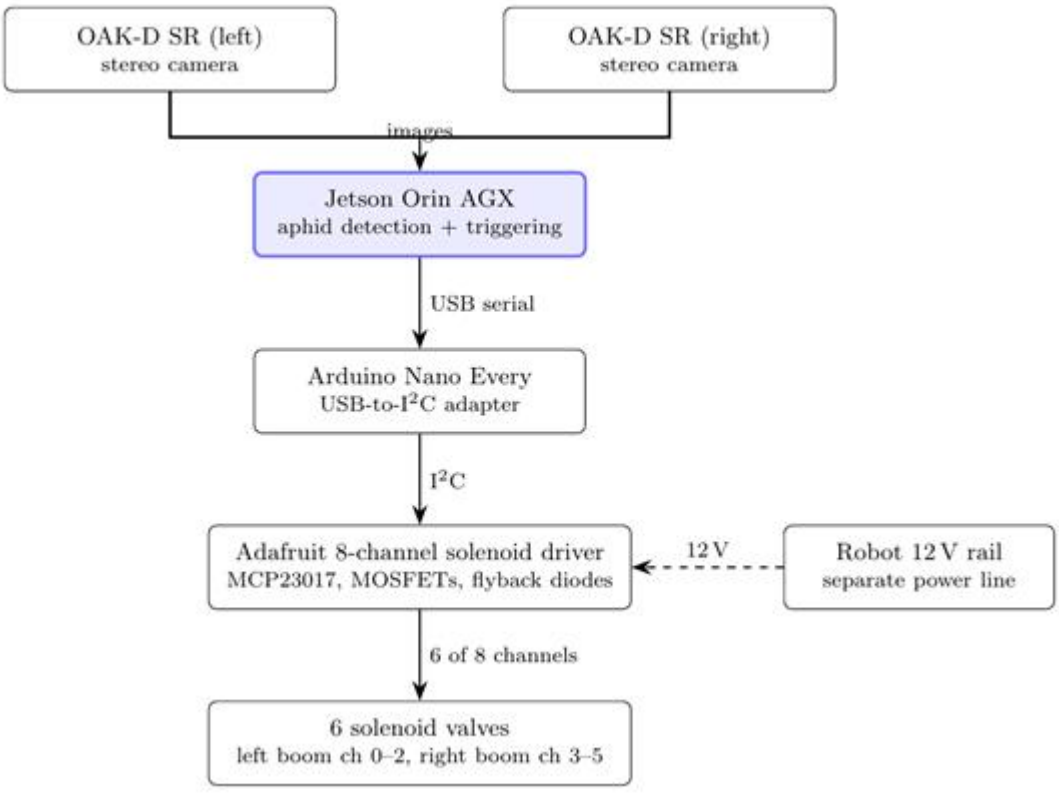
1. **Coarse spatial resolution.** Only two nozzle groups were exposed, so a detection anywhere in a camera's field of view sprayed the whole side.
2. **Dependence on proprietary hardware and protocol.** Duty cycle and group mapping were constrained by the ECU.
3. **Fragility without the ECU.** If the ECU was unpowered or disconnected, the CAN controller's transmit queue filled up ("No buffer space available") and the bus logged acknowledgement errors continuously.

These three motivated a custom actuator with per-nozzle control, no proprietary dependencies, and well-defined behavior when the host is disconnected.

3.4.2 Custom I²C solenoid driver

The replacement is built around an Adafruit I²C to 8-Channel Solenoid Driver. The board carries an MCP23017 16-bit I²C I/O expander (default address 0x20). Each output drives an onboard N-channel MOSFET (AO3406) that switches a separate solenoid supply (V+, 3–24 V DC); the valves' positive side ties to the shared V+ rail, and the MOSFET completes the path to ground when its channel is active. Each channel includes a fly-back diode across the coil to clamp back-EMF and an LED to show channel state. Six channels are used: nozzles 0–2 of the left boom on channels A2–A4, nozzles 3–5 of the right boom on channels A5–A7. The actuation signal chain, from the two cameras through to the valves, is shown in the figure below.

Figure 3.3. Actuation signal chain



wo OAK-D SR cameras feed the Jetson Orin AGX, which drives the solenoid valves through a USB-to-I2C bridge and the Adafruit driver board. Signal paths are solid; the 12 V supply (dashed) reaches the board on a separate line from the robot's power rail.

The field computer's I^2C bus did not reach the board through the Jetson's 40-pin header; the carrier board's I^2C path was faulty. An Arduino Nano Every was placed between the Jetson and the board as a USB-serial-to-I^2C bridge. The Jetson sends line-based ASCII commands over USB, the Nano performs the I^2C writes, and a udev rule gives the bridge a stable device name (/dev/solenoid). The firmware adds its own safety layer: a 500ms communication watchdog drops all outputs if no command arrives, pulses are capped at 5 s, and the output latch is cleared before the pins are configured at start-up.

The ROS 2 driver node (i2c_solenoid_driver) takes NozzleCommand messages, maintains the commanded state as an 8-bit mask, and sends it to the Nano over the serial bridge when the state changes (Tables 3.5 and 3.6).

Table 3.5. Solenoid driver node interface.

Interface	**Name**	**Type**
Subscribed	/sprayer/nozzle_command_i2c	sprayer_interfaces/NozzleCommand
Published	/sprayer/nozzle_status_i2c	sprayer_interfaces/NozzleStatus
Service	solenoid_emergency_stop	std_srvs/SetBool

Table 3.6. Solenoid driver node parameters.

Parameter	**Default**	**Purpose**
port	/dev/solenoid	Bridge serial device
heartbeat_period	0.2 s	Interval for re-sending the latched mask (under the 0.5 s firmware watchdog)
max_on_duration	5.0 s	Forces off any channel left on longer than this

Layered safety design. Five independent mechanisms switch valves off, so no single tested software or communication failure leaves a valve open:

1. **Trigger node.** Every spray has a fixed end time.

2. **Driver node.** Any channel on longer than max_on_duration is forced off, covering a trigger node that crashes after sending ON.
3. **E-stop service.** Switches all channels off and ignores further ON commands until released.
4. **Firmware watchdog.** If the driver process dies or USB is disconnected, the Nano drops all outputs within 500ms.
5. **Shutdown and start-up.** The driver sends all-off on exit, and the firmware starts with outputs cleared.

3.4.3 CAN back end retained as software fallback

The CAN driver and bridge nodes remain in the ROS 2 stack and publish on separate topics from the solenoid back end, so a reconnected TeeJet ECU can be driven again without code changes. A launch argument disables the CAN back end when the ECU is not connected, avoiding the queue-filling behavior described above.

3.5 Position-Based Spray Triggering

The goal of the triggering layer is to spray where each aphid was seen, independent of the robot's speed. The position-based trigger does this by localizing every detection in 3D, fixing it in an odometry frame at the image timestamp, and opening the boom while the robot carries the boom over that point. A simpler time-based trigger, which sprays for a fixed duration from the moment a detection arrives, is kept as a fallback for constant-speed operation. Both are selected at launch by spray_trigger:= tf|boom.

3.5.1 Position-based trigger

Every sensor and the boom are attached to one robot frame, and a local EKF moves that frame through a fixed odom frame, so a detection can be carried from the camera to the boom with tf2. The transform chain is odom base_link body {cameras, IMU, sprayer}. The frames base_link and body coincide; body is the CAD origin of the robot description, 0.207 m behind the front

axle and 0.466 m above the wheel centers. Axes follow the ROS convention (x forward, y left, z up). Fixed sensor and boom offsets are listed in Table 3.7.

Table 3.7. Sensor and boom frames relative to body.

Frame	**x (m)**	**y (m)**	**z (m)**	**Rotation**
left_camera_mount	0.343	0.222	0.203	yaw +90° (looks along +y)
right_camera_mount	0.343	−0.222	0.203	yaw −90° (looks along −y)
sprayer	−0.826	0	0	none
imu	0.152	−0.127	−0.127	none

Each camera mount carries an optical frame (z out of the lens, x right, y down). Because the cameras face sideways, the image's horizontal axis runs along the direction of travel, so a target's forward position comes mostly from its pixel column scaled by its depth.

A robot_localization EKF in planar mode publishes the odom base_link transform and /odometry/local. It takes forward and lateral velocity from wheel odometry and heading rate from the IMU gyro only; wheel-odometry yaw is excluded because skid-steer slip makes it unreliable (the measured bias is reported in Section 3.6.2). Only relative motion matters for spraying: a target is fixed in odom when it is seen and tracked over the next one to two meters, so slow drift of odom does not move the spray point.

Each detection is turned into a stored target in four steps:

1. **Depth.** The median depth of a pixel patch at the bounding-box center is read from the aligned stereo depth image; values outside 0.05–2.0 m are rejected.
2. **Back-projection.** The pixel is back-projected with the camera intrinsics,
3. **Anchoring.** The point is transformed to odom with the transforms at the image timestamp ,

o and merged with any stored target within 1 cm.

4. **Tracking.** At every 50 Hz control tick, each stored target is re-expressed in the sprayer frame using the current pose, and only its forward coordinate is used, since the whole boom fires together.

Stereo depth replaced an earlier monocular estimate based on an assumed aphid size, . For approximately 13 px targets, a 2 px bounding-box error already gives approximately 15 % depth error, before any variation in true aphid size.

A boom opens while at least one stored target lies in a spray window around it, where cm is the padding, is the forward speed, and covers the valve's opening delay (set to zero in the current configuration). Targets 5 cm past the boom are discarded, and if the robot stays below 0.1 m/s with a target under the boom, the spray is cut after 2 s. The left camera drives nozzles 0–2 and the right camera nozzles 3–5.

3.5.2 Time-based trigger as fallback

The time-based trigger (aphid_boom_trigger) opens a boom when a camera reports at least 3 detections in one frame, keeps it on for a fixed 0.762 s, and holds a 1.0 s cooldown measured from spray start. A 10 Hz timer closes the valves when the spray time has elapsed. This behavior is correct at the design speed of 1 m/s with plants spaced on 0.254-m centers; at other speeds the spray point shifts by (from the detection latency) and by s (from the fixed duration). It is retained as the simpler, lower-dependency fallback: it needs no stereo depth, no tf2, and no EKF, and it was used during the fail-safe and bench verification described in Section 3.6.1.

3.6 System Verification

The system was verified as an integrated design rather than evaluated for agronomic effect. Three properties were checked: end-to-end timing and fail-safe behavior; state-estimation accuracy; and the spray-window of the position-based trigger, with ground truth from water-

sensitive papers in two hardware runs. Field performance on live aphid infestations is left to future work (Section 3.7).

3.6.1 End-to-end latency and fail-safe behavior

The actuation chain was first tested in software against a program that emulated the Nano's protocol and 500ms watchdog on a pseudo-terminal. Six scenarios were exercised and all behaved as expected: a left-camera spray fired on 3 detections; a 2-detection frame produced no output; a right-camera spray fired and released after the spray duration; the max-on-duration forced off a stuck channel at 5 s; the e-stop blocked ON commands; and killing the driver dropped all outputs within the firmware watchdog.

The integrated system was then run on the bench with the left camera live and the solenoid driver connected through the bridge. With aphids continuously visible, the boom cycled about 0.82 s on and 0.20 s off, which follows from the fixed 0.762 s spray duration (checked by a 10 Hz timer) and the 1 s cooldown. The three nozzles in a boom switched on up to about 13ms apart because each nozzle is a separate NozzleCommand, and therefore a separate serial transaction.

Throughput and latency were measured on the robot. The budget set by the 1.17 m camera-to-boom spacing is 4.7 s at 0.25 m/s, 2.3 s at 0.5 m/s, and 1.17 s at the 1 m/s operating ceiling. On-robot capture-to-detection latency, measured by matching image and detection header stamps over the two hardware trigger runs, was 88–92ms median and 364–476ms across the two cameras. For the position-based trigger, latency within the budget does not shift the spray point because each detection is anchored at its image-header timestamp.

Actuation latency, from each ROS nozzle command to the Nano's first acknowledgement over USB, was 120–153 ms median and 164–192 ms p95 on turn-on, and 75–166 ms median on

turn-off. A single 2.57 s maximum on turn-off was observed in Run 2. The latency is bounded by the Nano's 5 Hz status rate (expected bias approximately 100ms); raising the status rate to 50 Hz is listed as future work.

3.6.2 State estimation

The EKF state estimate was measured in six indoor, joystick-driven runs with tape-measured stopping distances (Table 3.8). The first four runs were straight-line characterization runs on each side; the last two (TF trigger 3 and 4) are the dedicated hardware trigger runs described in Section 3.6.3. All runs used the final configuration: gyro-only EKF heading, wheel-odometry only (yaw excluded), 3ms camera exposure, and detection threshold 0.6.

Table 3.8. State-estimation accuracy across six runs.

Run	True dist. (m)	EKF dist. error	Mean / max speed (m/s)	EKF heading err. vs. gyro	Side drift (cm)
Right 1	2.540	−3.9 %	0.24 / 0.51	0.6°	7.5
Right 2	2.134	−6.0 %	0.32 / 0.60	0.3°	4.6
Left 1	2.159	−3.6 %	0.24 / 0.50	0.04°	4.9
Left 2	2.108	−6.6 %	0.28 / 0.57	0.05°	3.3
TF trigger 3	3.277	−10.8 %	0.27 / 0.53	0.02°	8.5
TF trigger 4	3.429	−1.8 %	0.36 / 0.71	0.03°	13.4

IMU, wheel odometry, and EKF rates were 100.6, 73, and 92 Hz. The stationary drift with the robot at rest for 15 s was ±1 mm.

The EKF under-read distance by 1.8–10.8 % across the six runs and held heading within 0.6° of the gyro. Wheel odometry alone reported 52–63° of turning on runs that were straight;

this is why the wheel-odometry yaw rate is excluded from the EKF. With gyro-only heading, the IMU constrains orientation but not distance, so the distance error is inherited from the wheels. The absolute distance error does not shift the spray point in the position-based trigger because stored targets and the sprayer frame are both expressed in the same EKF frame; their difference is unchanged under a shared frame error (Section 3.6.3).

3.6.3 Position-based trigger: simulation and hardware

The position-based trigger was first exercised in simulation with targets driven past the boom (Table 3.9). Both the onset and end of the simulated spray fell within 2 cm of the target window, which is within one control tick of travel at 0.5 m/s (1 cm) plus the padding resolution.

Table 3.9. Simulated spray-window accuracy of the position-based trigger.

Metric	Value
Onset error relative to target entry	1.7 cm early
End error relative to target exit	0.8 cm early
Target speed	0.5 m/s

The trigger was then verified on the robot in two dedicated hardware runs (tf_trigger_run1 and tf_trigger_run2), both using the same configuration: spray-window margin ±0.10 m, valve delay 0.0 s (not yet measured), detection threshold 0.6, and the final EKF (gyro-only heading; wheel-odometry only, yaw excluded). In both runs the plants were placed on the robot's left side only, and both cameras ran the full pipeline (detector, trigger, driver), so the right-facing camera served as a running false-positive meter.

Run 1 covered 3.277 m of tape distance at 0.27 m/s mean (0.53 m/s max). The EKF under-read distance by 10.8 % and matched the gyro heading within 0.02°. The left boom fired one continuous ON cycle of 8 s whose sprayer-x span covered all three taped plant positions (−1.032 m to +0.683 m, 1.72 m span over three plants at approximately 0.9 m spacing). Ground

truth was taken with water-sensitive papers placed at each taped plant mark on the left side: every paper was wet after the run, confirming the boom opened over the correct odom coordinate of each plant. On the empty right side, the detector produced 155 “aphid” frames out of 588 (26 %) at scores 0.60–0.74, averaging 1.1 boxes per hit frame; these stored targets drove three short right-boom cycles. Plant-side frames averaged 2.8 boxes per hit frame.

Run 2 covered 3.429 m at 0.36 m/s mean (0.71 m/s max) with plants again on the left side. The EKF distance error was 1.8 % and heading matched the gyro within 0.03°. The left boom fired three distinct cycles whose sprayer-x spans were 0.21 m, 0.79 m, and 1.73 m, separated by 3 cm and 26 cm gaps. The empty right side produced 46 “aphid” frames out of 737 (6 %) at scores 0.60–0.72, driving two right-boom cycles. Plant-side frames averaged 3.7 boxes per hit frame; empty-side frames averaged 1.1. Ground-truth papers were not placed for Run 2; the sprayer-x coordinates at each left-boom ON event align with the taped plant positions.

Table 3.10. Spray events in the two position-based trigger runs.

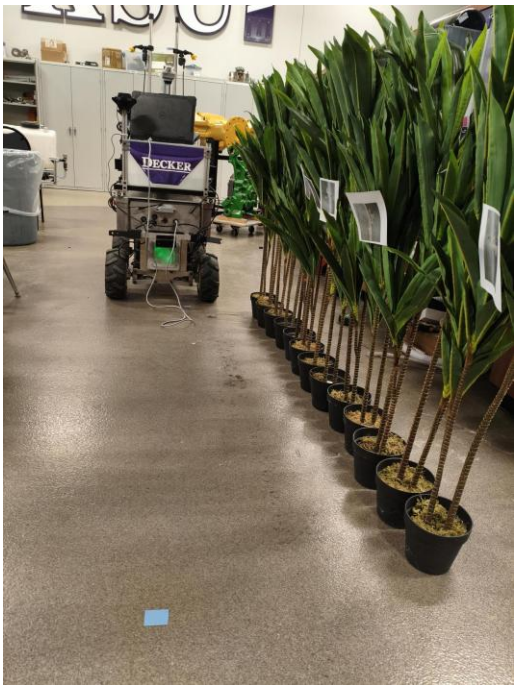
RUN	PLANT-SIDE CYCLES	PLANT-SIDE SPANS (M)	EMPTY-SIDE CYCLES	GROUND TRUTH
1	1	1.72	3	papers wet at taped positions
2	3	0.21, 0.79, 1.73	2	—

The stored target coordinate and the sprayer-x are both carried in the same EKF frame, so an absolute error in that frame does not translate to a relative error between them. The ground-truth papers in Run 1 were wet at the correct taped positions despite the 10.8 % EKF distance under-read.

These results verify the system as an integrated design: the actuation chain fails safely under the tested scenarios, end-to-end timing fits inside the camera-to-boom budget, the EKF

holds heading, and the position-based trigger places spray on the correct plant positions in two hardware runs ground-truth-verified with water-sensitive papers. Characterization of placement error across the full operating envelope, multiple speeds, turning maneuvers, live-aphid field conditions, measured valve delay is left to future work.

Figure 3.4. Aphid detection and triggering lab test setup.



3.7 Discussion

The main limitation of the system is that the hardware trigger runs cover a narrow slice of the operating envelope. Only two mean speeds were tested, both along straight-line trajectories, with printed targets rather than live aphids. The position-based trigger is designed to be independent of the EKF's absolute accuracy, but speed sweeps, turning maneuvers, and live-aphid tests are needed before any field-performance claim can be made.

Several design choices led to the current form. On the perception side, two USB OAK-D SR cameras replaced the earlier PoE Arducam/OAK-FFC setup because the PoE prototype was unreliable, the OAK-D SR runs on the stock DepthAI driver and provides stereo depth without

additional work. For detection, RT-DETR, and later RT-DETRv2-R101, was deployed through TensorRT after three YOLO26 variants (n/640, s/960, s/1024) trained on the same aphid dataset held live-frame confidences at or below 0.011 while RT-DETR produced spatially stable detections at 0.20–0.28 under the same training setup.

On the actuation side, an Adafruit MCP23017 I^2C expander driving per-nozzle MOSFET solenoids replaced the TeeJet CAN ECU as the primary actuator. The TeeJet exposed only two nozzle groups and no hardware-level visibility, whereas the Adafruit board is inexpensive, has a smaller physical footprint, and is debuggable at the pin level. The I^2C link was routed through an Arduino Nano Every over USB rather than through the Jetson's 40-pin header because the carrier's I^2C header was faulty, the Nano bridge was available immediately and places the safety-critical timeout logic in dedicated firmware. The CAN and solenoid back ends publish on separate ROS 2 topics so that both can run concurrently, and a change to one cannot propagate to the other. Safety is layered across the trigger node, the driver node, the e-stop service, the firmware watchdog, and startup/shutdown clearing, so different failure modes are covered by different mechanisms rather than a single timeout.

For spray triggering, the position-based trigger was adopted as the primary path and the time-based trigger kept as a selectable fallback. The position-based trigger lifts the fixed-speed assumption of fixed-duration spraying and is now ground-truth-verified; the time-based trigger is retained for simpler constant-speed cases. For 3D target localization, stereo depth was used rather than monocular depth from an assumed aphid size: for approximately 13-pixel targets a 2-pixel bounding-box error already produces approximately 15 % depth error, before any variation in actual aphid size.

Several measured values point to specific future work. The EKF under-read distance by 1.8–10.8 % across six runs; this does not shift the position-based spray point, because stored targets and the sprayer frame are both carried in the same EKF frame, but the ±0.10 m spray-window margin currently covers the error. Per-wheel encoder logging and a wheel-radius calibration would let the margin tighten to ±0.03 m, reducing chemical use per plant. The valve-opening delay is set to 0.0 s in the current configuration; at higher operating speeds, measuring the time from the ROS nozzle command to the first visible drop with high-speed video, or by raising the Nano's 5 Hz status rate to 50 Hz for a sharper actuation measurement and setting valve_delay would improve placement at speed. The empty-side false-positive rate was 26 % and 6 % in the two hardware runs at the current 0.60 detection threshold; a detection-threshold or minimum-boxes-per-frame sweep is left to future work. More broadly, the operating envelope needs widening with higher-speed runs, turning maneuvers, and live-aphid tests before any field-performance claim can be made, and the TensorRT engine still needs to be built on the deployment device.

3.8 Summary

This chapter presented the design and verification of a real-time aphid detect-and-spray system on the Decker-03 skid-steer field robot (Chapter 2). The system runs a TensorRT-accelerated RT-DETRv2 detector on images from two side-mounted OAK-D SR stereo cameras, and drives a six-nozzle solenoid driver through a USB-to-I^2C Arduino Nano Every bridge with five independent fail-safe layers; a commercial TeeJet CAN back end is retained as a software fallback. A position-based trigger localizes each detection with stereo depth, fixes it in an EKF odom frame via tf2, and opens the boom as the sprayer passes over it; a simpler time-based trigger is selectable as a constant-speed fallback. In two dedicated hardware runs the position-

based trigger placed spray on the correct taped plant positions, verified with water-sensitive papers on the plant side. End-to-end capture-to-detection latency was 88–92 ms median against a 1.17 s budget at 1 m/s, and the EKF held heading within 0.6° of the gyro while under-reading distance by 1.8–10.8 % across six runs. Characterization of spray-placement error across the full operating envelope, and field evaluation on live aphid infestations, are the next steps.

Taken together with Chapter 2, this thesis contributes a physical platform, a vision-guided intervention stack deployed on it, and a digital twin methodology for continuing development. The detect-and-spray system described in this chapter was developed and verified on the physical robot. Closing the sim-to-real loop between the digital twin and the intervention stack is identified as the next development milestone, with simulation-side work on row navigation, detector dataset augmentation, and trigger evaluation as next steps.

Bibliography

Botta, A., Cavallone, P., Baglieri, L., Colucci, G., Tagliavini, L., & Quaglia, G. (2023). "In-depth analysis of power balance, handling, and the traction subsystem of an articulated skid-steering robot for sustainable agricultural monitoring." *SN Appl. Sci.*, 5:103. URL https://doi.org/10.1007/s42452-023-05321-y.

Calvin, L., Martin, P., & Simnitt, S. (2022). "Adjusting to higher labor costs in selected U.S. fresh fruit and vegetable industries." *Economic Information Bulletin EIB-235*. U.S. Department of Agriculture, Economic Research Service, Washington, DC.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). "End-to-end object detection with transformers." *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pp. 213–229.

Chen, J., Fan, Y., Wang, T., Zhang, C., Qiu, Z., & He, Y. (2018). "Automatic segmentation and counting of aphid nymphs on leaves using convolutional neural networks." *Agronomy*, 8(8):129. URL https://doi.org/10.3390/agronomy8080129.

Gai, J., Xiang, L., & Tang, L. (2021). "Using a depth camera for crop row detection and mapping for under-canopy navigation of agricultural robotic vehicle." *Comput. Electron. Agric.*, 188:106301.

Kim, K., Deb, A., & Cappelleri, D. J. (2024). "P-AgSLAM: In-row and under-canopy SLAM for agricultural monitoring in cornfields." *IEEE Robot. Autom. Lett.*, 9(6):4982–4989. URL https://doi.org/10.1109/LRA.2024.3386466.

Koenig, N., & Howard, A. (2004). "Design and use paradigms for Gazebo, an open-source multi-robot simulator." *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 2149–2154.

Lv, W., Zhao, Y., Chang, Q., Huang, K., Wang, G., & Liu, Y. (2024). "RT-DETRv2: Improved baseline with bag-of-freebies for real-time detection transformer." *arXiv*, arXiv:2407.17140.

Mandow, A., Martínez, J. L., Morales, J., Blanco, J. L., García-Cerezo, A., & González, J. (2007). "Experimental kinematics for wheeled skid-steer mobile robots." *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 1222–1227.

Müller, M. S., Jazdi, N., & Weyrich, M. (2022). "Self-improving models for the intelligent digital twin: Towards closing the reality-to-simulation gap." *IFAC-PapersOnLine*, 55(2):126–131.

Partel, V., Kakarla, S. C., & Ampatzidis, Y. (2019). "Development and evaluation of a low-cost and smart technology for precision weed management utilizing artificial intelligence." *Comput. Electron. Agric.*, 157:339–350.

Pioneer (n.d.). "Row width in corn grain production." *Pioneer Agronomy*. Corteva Agriscience. URL https://www.pioneer.com/us/agronomy/row-width-grain-corn.html. Accessed October 15, 2026.

Pylianidis, C., Osinga, S., & Athanasiadis, I. N. (2021). "Introducing digital twins to agriculture." *Comput. Electron. Agric.*, 184:105942.

Ragsdale, D. W., Landis, D. A., Brodeur, J., Heimpel, G. E., & Desneux, N. (2011). "Ecology and management of the soybean aphid in North America." *Annu. Rev. Entomol.*, 56:375–399.

Rai, N., Zhang, Y., Ram, B. G., Schumacher, L., Yellavajjala, R. K., Bajwa, S., & Sun, X. (2023). "Applications of deep learning in precision weed management: A review." *Comput. Electron. Agric.*, 206:107698.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). "You only look once: Unified, real-time object detection." *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 779–788.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). "Faster R-CNN: Towards real-time object detection with region proposal networks." *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 28.

Schmitz, A., Badgujar, C., Mansur, H., Flippo, D., McCornack, B., & Sharda, A. (2022). "Design of a reconfigurable crop scouting vehicle for row crop navigation: A proof-of-concept study." *Sensors*, 22(16):6203. URL https://doi.org/10.3390/s22166203.

Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). "Digital twin in industry: State-of-the-art." *IEEE Trans. Ind. Informat.*, 15(4):2405–2415.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). "Domain randomization for transferring deep neural networks from simulation to the real world." *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 23–30.

United Nations, Department of Economic and Social Affairs, Population Division (2017). "World population prospects: The 2017 revision." *Technical report*. United Nations, New York.

Upadhyay, A., Sunil G C, Zhang, Y., Koparan, C., & Sun, X. (2024). "Development and evaluation of a machine vision and deep learning-based smart sprayer system for site-specific weed management in row crops: An edge computing approach." *J. Agric. Food Res.*, 18:101331. URL https://doi.org/10.1016/j.jafr.2024.101331.