WAVE
Hands-On Activity Report

Students: Nick Hollenback, Alejandro Lopez, Sydney Garrison, Terry Nguyen

Instructors: Dr. Abadi, Dr. Moghadam

Date: 3/29/26

Table of Contents

Team Information

Course/Section	Lab Date	Time Slot	Lab Location
Workforce for Autonomous Vehicle Engineering (WAVE)	03/11/26	4:30PM – 6:00 PM	RVR 4005
EEE Students	**CE Students**	**Team Number**	**Email Contact**
Sydney Garrison, Terry Nguyen	Nick Hollenback, Alejandro Lopez	Team 2	garr4726@gmail.com terrytndn@gmail.com nicholashollenback@gmail.com

Team Photo from Activity 5:

(Garrison, Lopez, Hollenback, Nguyen)

Purpose and Learning Outcomes

In this Hands-On Activity, Civil Engineering and Electrical Engineering students team up to complete a series of activities using a Quanser QCar 2 autonomous vehicle. The goal of WAVE is to prepare both civil engineers and electrical engineers for work with Autonomous Vehicles (AVs) in the future.

Pre-Lab Preparation (User Manuals & Concept Reviews)

Reference: QCar2_Hardware_Tests_MATLAB.pdf

Before starting the lab, students will review a series of documents that will prepare them for the lab. These documents include hardware and software overviews, information on sensors and sensor fusion, and information on color coding.

Document	Key takeaways (2–3 bullets)
user_manual_system_hardware.pdf	• The QCar 2 integrates a variety of sensors (LiDAR, depth camera, CSI cameras, IMU, encoders, Jetson Orin) • Detailed extrinsic matrix transforms define between sensors and the vehicle body. This is essential for perception and fusion tasks. • There are some Electrical and environmental constraints which include strict current limits, ESD sensitivity, and indoor only operation, and pins limited between 5V and 1.8V
user_manual_software_simulink.pdf	• Simulink models must use fixed step solves and correct QUARC target settings • To get data without Simulink models, you must use Rate transition blocks at high and low signals. • Deployment uses Monitor and Tune, with QUARC monitor providing visibility into model loading, execution, and debugging
user_manual_power.pdf	• LiPo battery safety is critical and has strict warnings • Charging requires careful handling and should never be left unattended. Don't charge below 10V. If the battery is hot, cool down before charging. • The QCar 2 shuts down automatically under 10V, warnings will start at 10.5V
Concept_review_sensor_noise.pdf	• Sensor noise is modeled as additive Gaussian white noise (ym = y + w) • Mean and Standard Deviation characterize the noise distribution • Real sensor data often fits a Gaussian distribution
Concept_review_sensor_fusion.pdf	• Sensor fusion improves accuracy by combining complementary strengths of different sensors • Complementary filters blend low-pass filtered slow signals with high-pass filtered fast signals. • Complementary filters will add up to 1
Concept_review_colour_spaces.pdf	• Images are stored as matrices with channels (grayscale use 1 channel, RDG uses 3). Each pixel holds intensity values from 0-255 • RGB mixes lightness and color, making it sensitive to lighting changes. HSV distinguishes light intensity from color information. • Grayscale and bitonal images are from weighted RBG combinations or thresholding, with grayscale approximating human brightness perception.

Hardware Tests

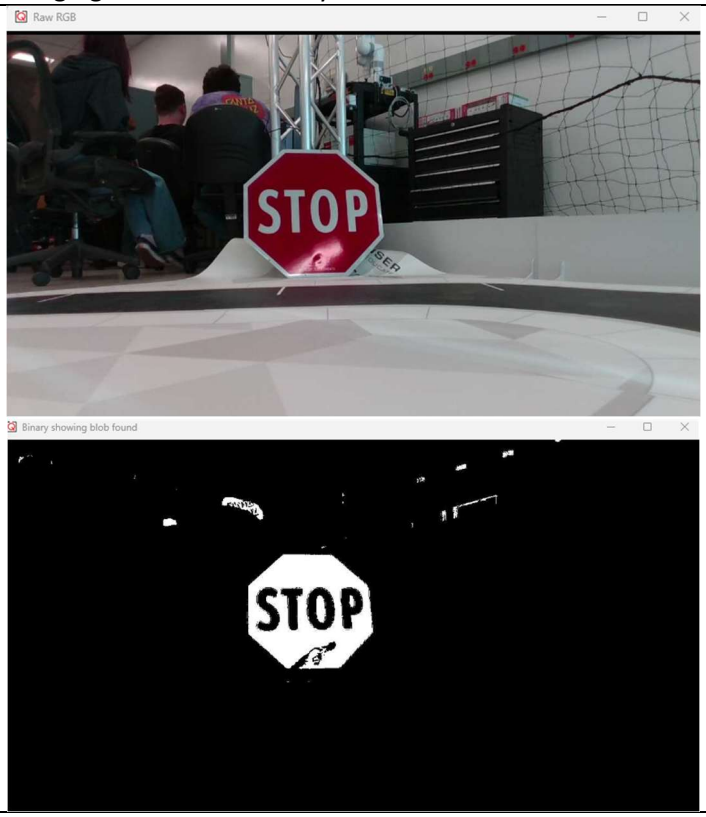
Complete the required hardware checks and record results.

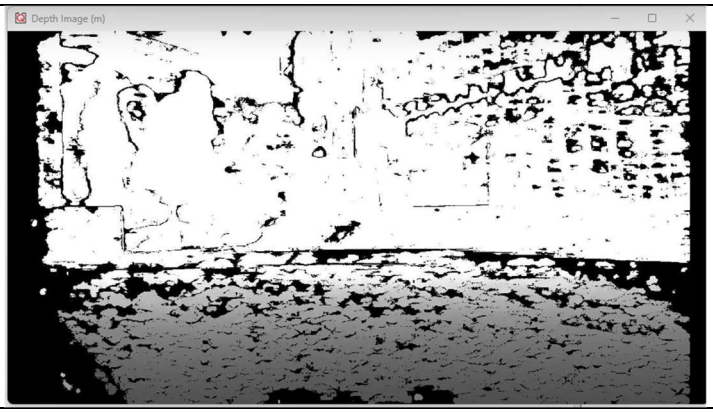
Test Item	Procedure (short)	Expected Result	Observed Result
Power-up / connections	Power the QCar 2 using a fully charged battery and perform a quick ping test to confirm that your system is connected	The car will power on and respond to the ping	The expected result was observed, the car powered on without issue and responded to a ping from the computer
Sensor connectivity check	Run the corresponding Simulink modules in MATLAB for the real sense camera, CSI cameras, and the LIDAR scanner a verify all the sensors are working properly	We should be able to view the different angles from the CSI cameras, See the depth of the image from the real sense camera, and obtain a polar plot from the LIDAR Scanner	All expected results were observed successfully
Data logging test	Verify that the LIDAR Scanner was able to take data a generate a plot saving the data successfully	We will see a polar plot from the lidar scanner meaning the data was transferred and saved	All expected results were observed successfully
Safety / E-stop verification	Safety / E-stop verification tests were not run for our lab section	Safety / E-stop verification tests were not run for our lab section	Safety / E-stop verification tests were not run for our lab section

Activity 1: RGBD Imaging

Reference: QCar2_RGBD_Imaging_MATLAB.pdf

Objective	The objective of this activity is to tune parameters so that certain street signs stand out. We can utilize color codes to tune the hue, saturation and value parameters so that only the stop sign or school crossing sign is visible. Then, using the depth camera, we can also extrapolate how far away this stop sign is. Color filtering is used to increase processing speed and lower processing power for object detection software. This is valuable in AVs, because reaction and processing time are critical.
Setup & Parameters	To set up this activity, we had to point the QCar 2's front cameras at the stop sign and open the appropriate Simulink

	model. We also had to ensure the IP address in Simulink hardware settings matched the car's.
Procedure Summary	First, we opened the Simulink file for activity 1. We want to capture only the stop sign and then find its distance from the camera. To do this, we set the hue to 0 for red, and adjust the value and saturation to match the stop sign on-camera. The module findStopSignLocation will identify the largest red section, and mark the center of it as the stop sign location. Then using the depth camera, the Simulink model accounts for the different field of view, and extracts the depth coordinates of the stop sign. We were also able to replicate this with a bright yellow school crossing sign by changing the hue to 35 for yellow.
Results (plots/screenshots)	

	
Issues & Fixes	We had to find an online reference to accurately choose our color information. Other than this, we had no issues.

Reflection:

It was interesting to see how color filtering greatly simplifies object detection and classification. Instead of training an AI model with hundreds of pictures of an item, this algorithm can classify objects based on color. Because existing road infrastructure already uses standardized colors and shapes for its street signs, we can use this consistency to our advantage when designing autonomous vehicles.

It was cool to see that the bright orange wall outlets and someone's shirt logo also stood out in this color-filtered image. This helped reinforce students' understanding of how the Simulink model was processing the RGB data.

1) What worked as expected? What did not?

Everything worked as expected. We were even able to identify other red objects including a t-shirt logo, a red bag, and the orange-red wall outlets.

2) What did you learn about the sensors/algorithms in this activity?

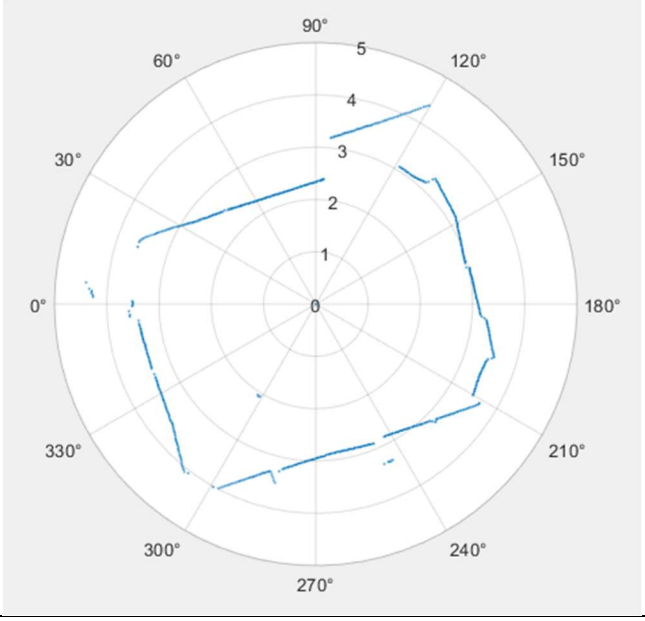
When an analog video signal is converted into the digital domain, it is stored in matrices where individual pixels have a color code associated with them. We can use this information to filter an image by the values each pixel has. This helps make object detection algorithms more efficient.

3) If you repeated the activity, what would you change?

We were a little bit short on time, so we didn't take as much time exploring as we might have otherwise. If we repeated this activity, I would like to try finding the correct color codes of other standard sign colors like blue, orange, and green.

Activity 2: Point Cloud Generation

Reference: QCar2_LIDAR_Point_Cloud_MATLAB.pdf

Objective	Capture LIDAR data from the RP LIDAR A2 on the QCar, send this data to a polar plot, and generate a point cloud map.
Setup & Parameters	Position the car inside the barriers on the map and using the provided Simulink file, run the module and verify its output of a point cloud map.
Procedure Summary	Launch the Simulink implementation of Lidar Point Cloud and set the Maximum Distance parameter within the Point Cloud subsystem accordingly. Next hit the run button and observe the produced plot, verifying the barrier locations
Results (plots/screenshots)	
Issues & Fixes	The produced polar plot was able to grab the barrier locations, but some parts of the plot seemed to be missing or misplaced. We ran the Simulink module multiple times and picked the best plot of the bunch.

Reflection:

1) What worked as expected? What did not?

The Simulink module and corresponding QCar platform ran as expected, but the polar plot seems to have some gaps and misplacements in the barriers.

2) What did you learn about the sensors/algorithms in this activity?

We figured out that the gaps in the Point Cloud polar plot were partially due to objects in the way closer to the QCar that prevented us from scanning portions of the barrier. The closer an object was, even if tiny, mean the larger the gap in the barrier on the polar plot. Other gaps and misplacements were likely due to the imperfect lighting in the lab as well as real gaps in the barrier itself.

3) If you repeated the activity, what would you change?

If the activity was repeated, we would seal the windows preventing stray sunlight from interfering with the sensor, as well as verifying that there are no objects within the barriers obstructing the LIDAR Scanner, and checking that the barriers were placed in a way that there are no gaps in between barriers

Activity 3: Manual Drive

Reference: Manual_Drive_MATLAB.pdf

Objective	Using a controller, we will manually drive the QCar around the map.
Setup & Parameters	Before running the example, we will connect the Logitech F710 Gamepad to the QCar using a USB dongle inserted into one of the USB 3.0 ports on the QCar
Procedure Summary	We will run the manual drive Simulink module allowing us to capture commands from the controller and pilot the QCar while simultaneously monitoring the battery remaining power consumption in watts as well as the car's speed in m/s all from the computer
Results (plots/screenshots)	As expected, as soon as we clicked the run button the QCar was able to be controlled using the Logitech Gamepad, and we took turns driving the car
Issues & Fixes	We were able to successfully pilot the QCar and the demo presented no issues.

Reflection:

1) What worked as expected? What did not?

Everything in this Demo worked as expected. The QCar responded to input from the controller and presented us with no issues.

2) What did you learn about the sensors/algorithms in this activity?

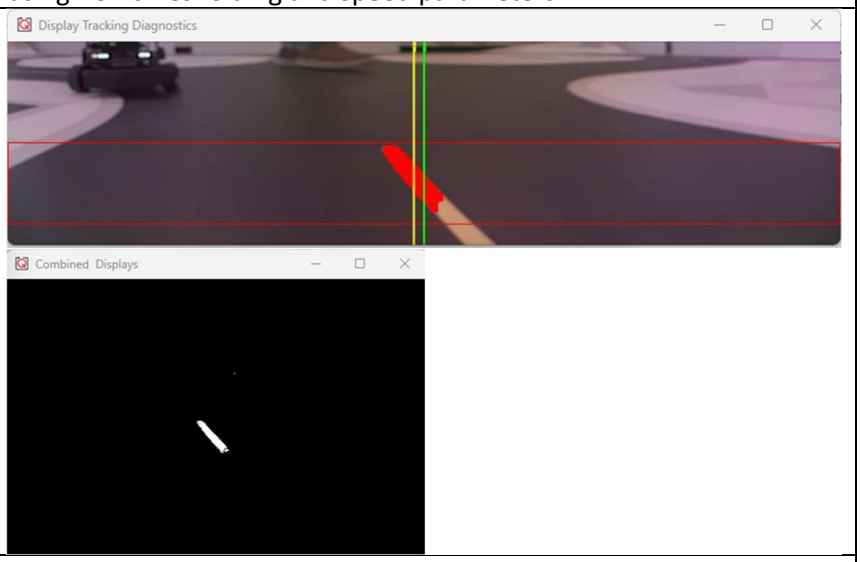
I believe the most important concept from this activity was the idea of an arming function when using remote control of a potentially dangerous device. To pilot the car and send commands you always have to hold down a specific button on the controller arming the car and making it receptive to other inputs. In this way you cannot accidentally pass inputs to the car unintentionally.

3) If you repeated the activity, what would you change?

Nothing would need to be changed if the activity was repeated, as there were no issues during the demonstration.

Activity 4: Lane Following

Reference: QCar2_Lane_Following_MATLAB.pdf

Objective	The Objective of this activity is to have our QCar implement lane following with obstacle detection in a MATLAB environment. We have the QCar use CSI data to extract lane tracking information, when then will generate steering/heading commands to the QCar.
Setup & Parameters	For the setup, we first had our QCar placed in an environment where there is a lane with lines on the floor. We then needed to activate the MATLAB code file to turn on the cameras. For parameters, we used HSV color thresholding, so the cameras would know what colors to be looking for to track. We also had speed parameters, where we could say how fast we wanted the QCar to move.
Procedure Summary	Made the QCar autonomously follow a lane and avoid any obstacles using HSV thresholding and speed parameters.
Results (plots/screenshots)	
Issues & Fixes	No issues and fixes were needed

Reflection:

1) What worked as expected? What did not?

   Our detection wasn't too good as it was hard to detect the proper color that we wanted without getting the curb scanned. Everything else worked as expected.

2) What did you learn about the sensors/algorithms in this activity?

   I learned that HSV thresholding is a very important and useful tool engineers can use to detect certain objects for AVs like this.

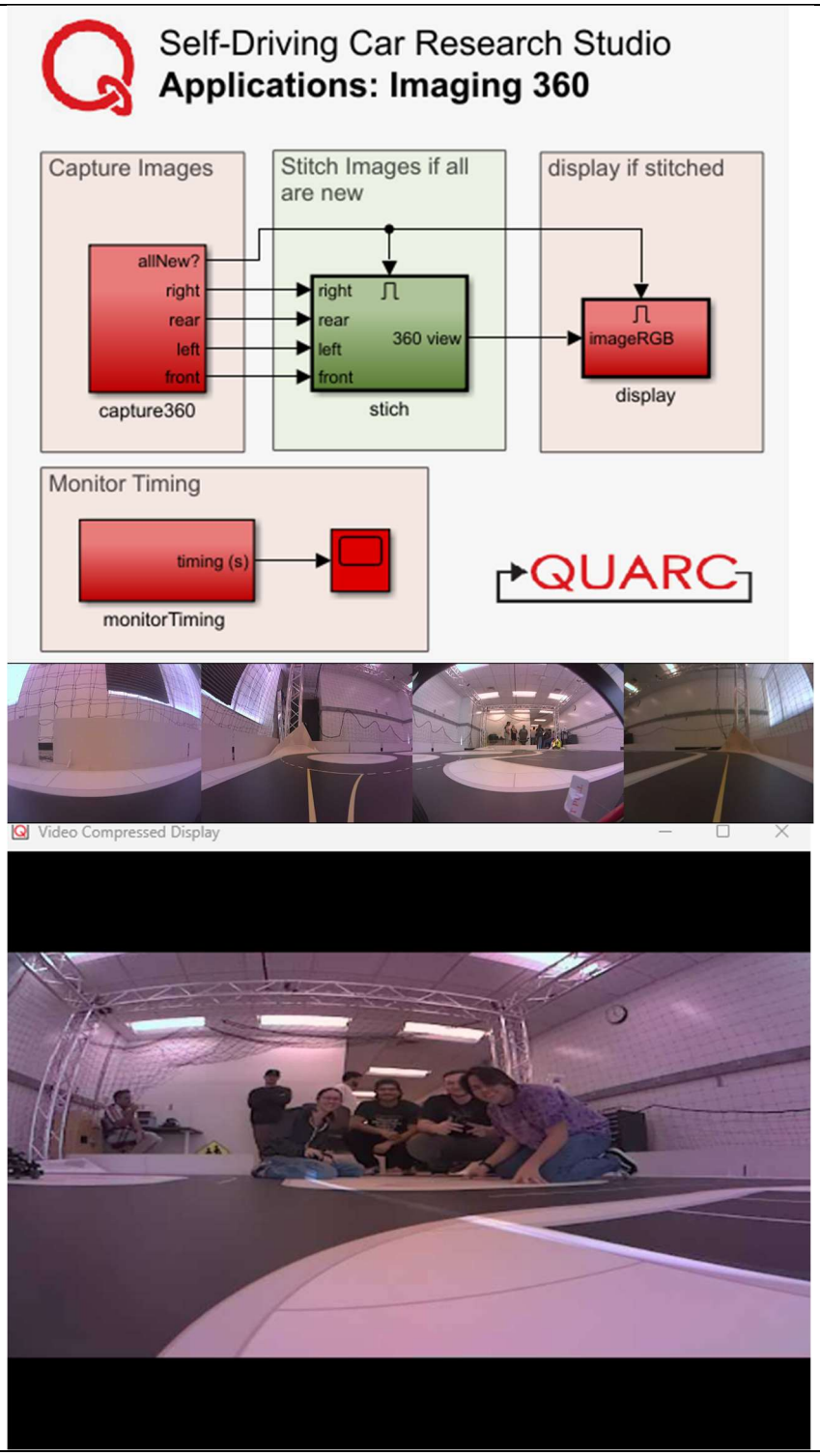
3) If you repeated the activity, what would you change?

   I would only change my initial HSV threshold to the one I knew worked, other than that I wouldn't change anything.

Activity 5: 360 Vision

Reference: QCar2_360_Vision_MATLAB.pdf

Objective	The objective of this activity is to create a 360-degree visual system for our QCar by capturing, stitching, and displaying images from four cameras. It will combine the front, left, right, and rear CSI cameras all into one continuous view, allowing for a full 360 view.
Setup & Parameters	For our setup, we had to activate the MATLAB code file for our QCar so the cameras would turn on. We also placed our QCar in the middle of an open area to scan whatever it sees. For our parameters, we had a resolution of 640 x 480 pixels with RGB format for coloring at 30 frames per second. Our stitching order was also set as a parameter. In this procedure, it was right, rear, left, and front.
Procedure Summary	We activated 4 cameras on top of the QCar to create a 360-degree camera for the car to detect and see all around it.

Results (plots/screenshots)	
Issues & Fixes	There were no issues or fixes

Reflection:

1) What worked as expected? What did not?

The 4 CSI cameras were stitched properly, and we got clear feed from the cameras. We even were able to take pictures of pretty good quality.

2) What did you learn about the sensors/algorithms in this activity?

I learned about how powerful stitching was as a tool and how you can turn regular cameras into a 360-degree camera if done properly

3) If you repeated the activity, what would you change?

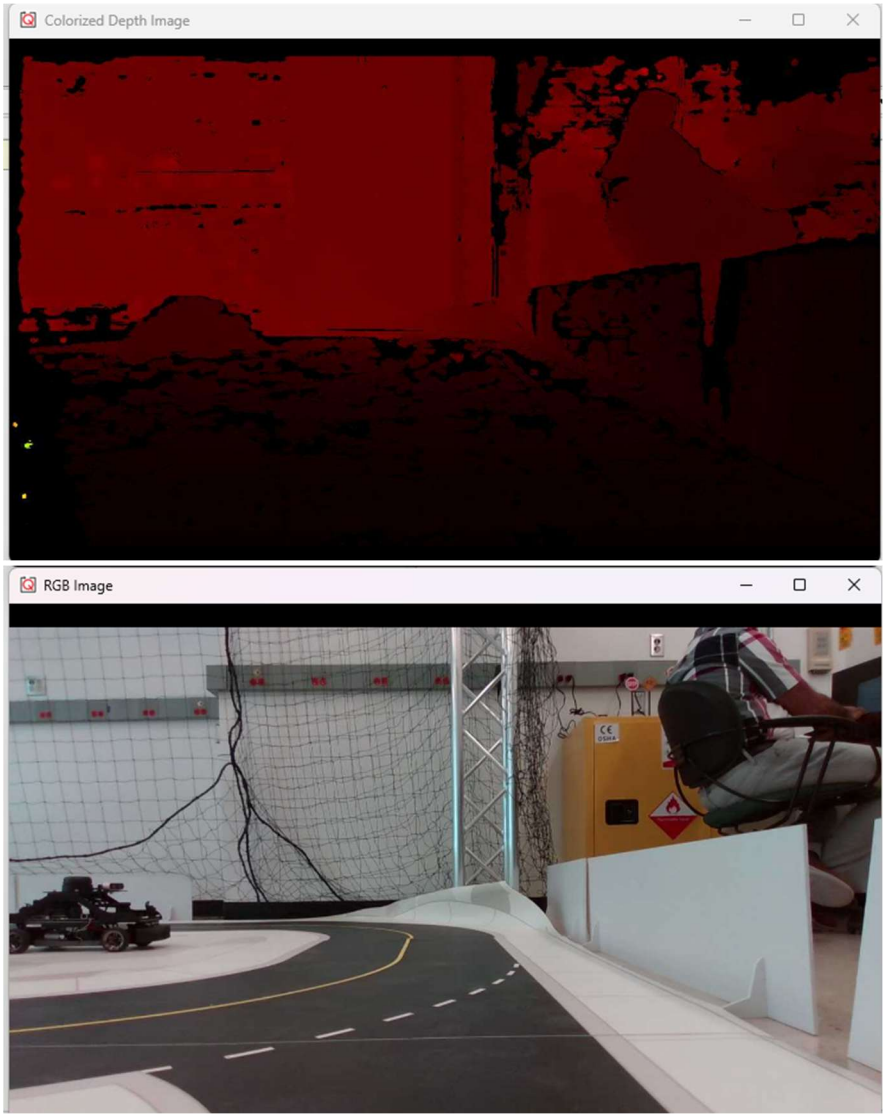
Everything worked as expected so there wasn't anything we would need to change if repeated.

Discussion and Conclusions

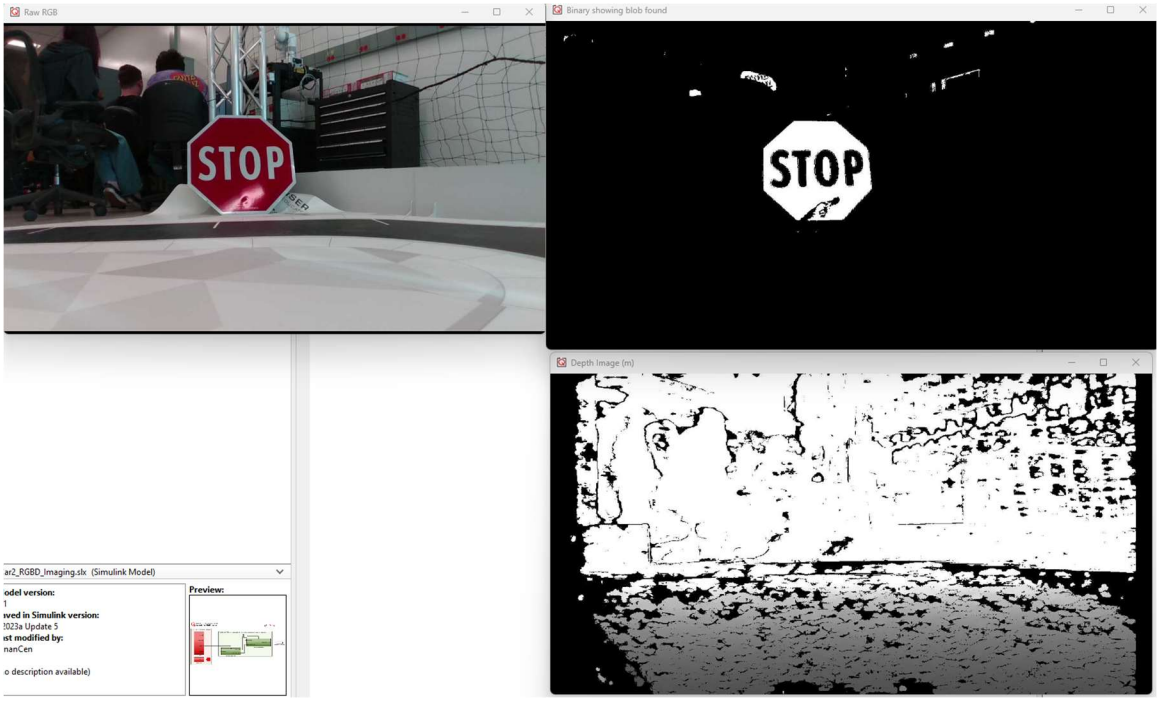
Overall, this lab successfully demonstrated the core concepts behind autonomous vehicle sensing and perception using the QCar platform. The hardware tests and multiple activities confirmed that the system was able to reliably capture and process data from cameras and LiDAR, while also highlighting real-world limitations such as sensor noise, environmental interference, and data inconsistencies. Activities like RGBD imaging and point cloud generation showed how different sensors can be used to extract meaningful information about the environment, even when the data is imperfect.

Additionally, this lab reinforced the importance of combining multiple sensing techniques and carefully tuning system parameters to achieve accurate results. While some issues such as gaps in LiDAR data and unintended color detections were observed, these challenges provided insight into how real autonomous systems must handle uncertainty and variability. Overall, the lab provided a strong foundation in sensor integration, data interpretation, and system-level thinking, all of which are essential for developing reliable autonomous vehicle technologies.

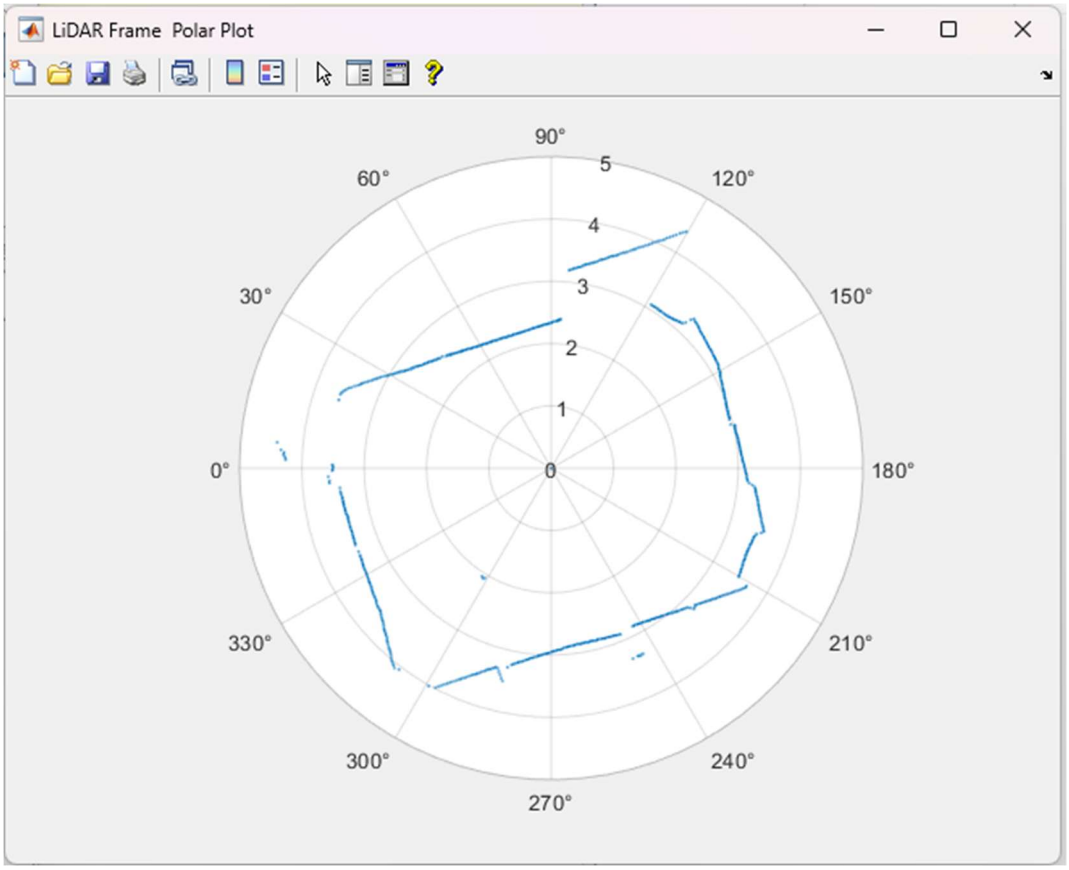
Appendix: Figures, Screenshots, and Data



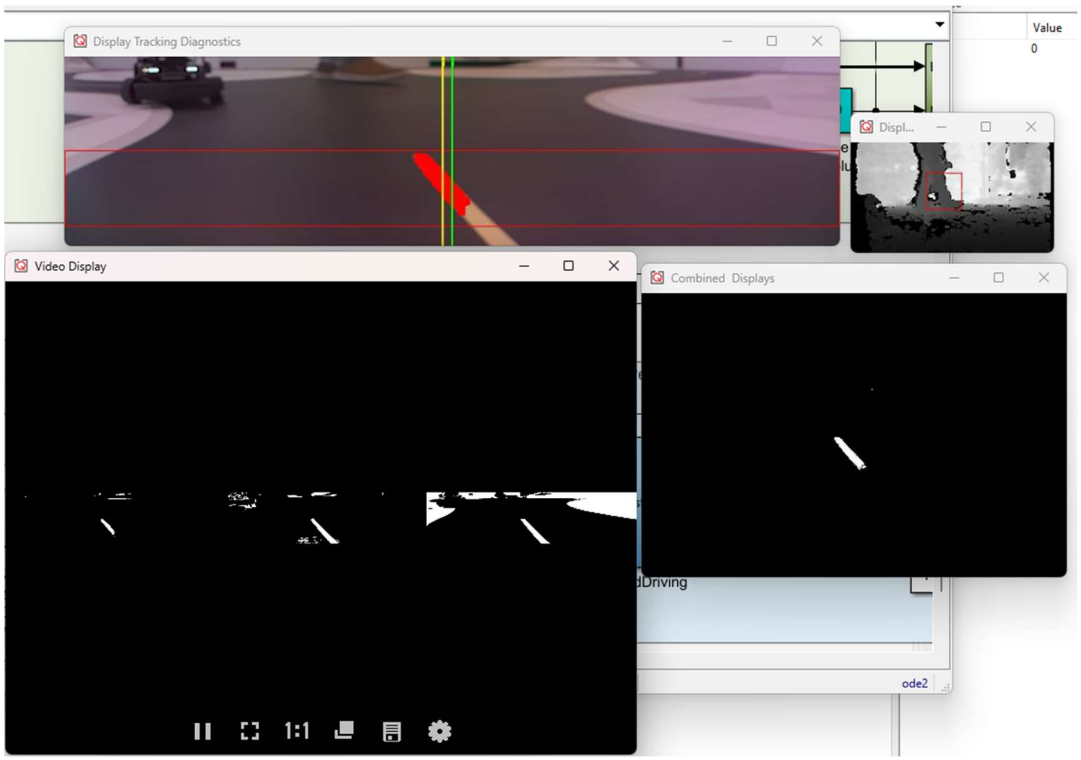
Depth Camera and RGB Camera



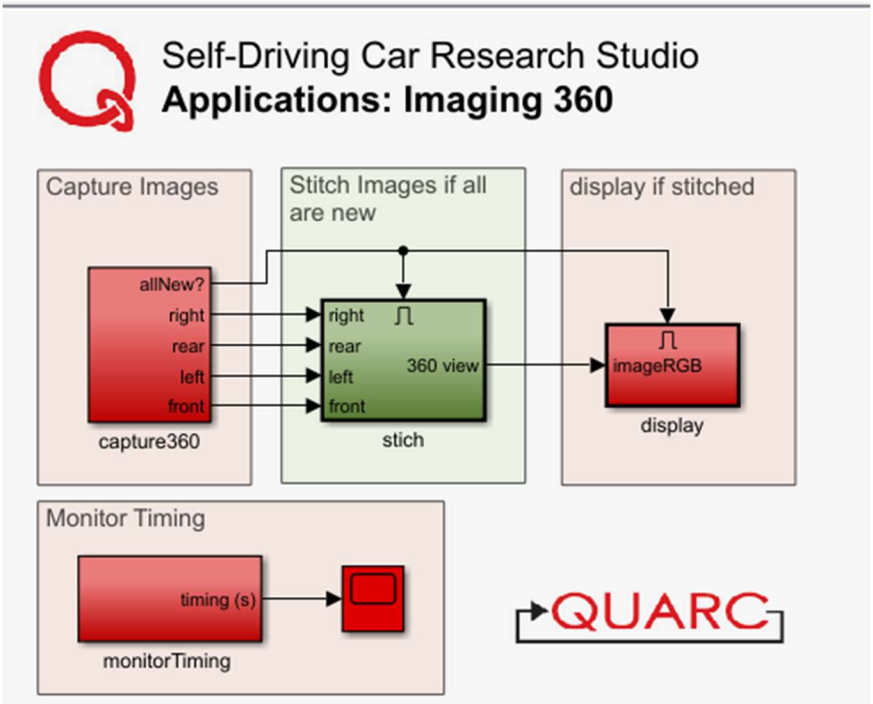
Activity 1: RGBD Imaging (stop sign detection)



Activity 2: Point Cloud Generation (Lidar map)

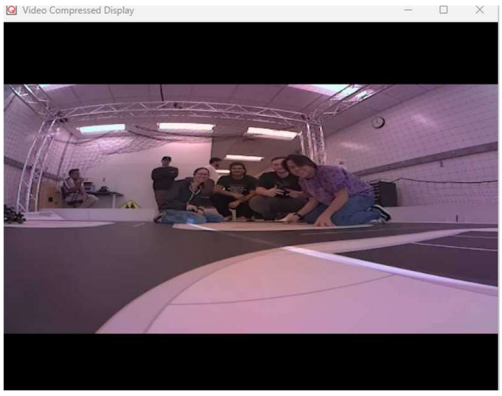


Activity 4: Lane Following

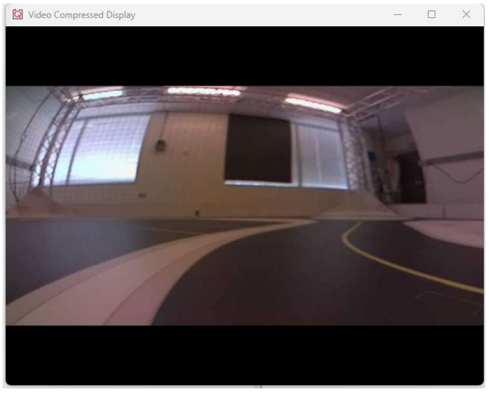


Activity 5: Imaging 360

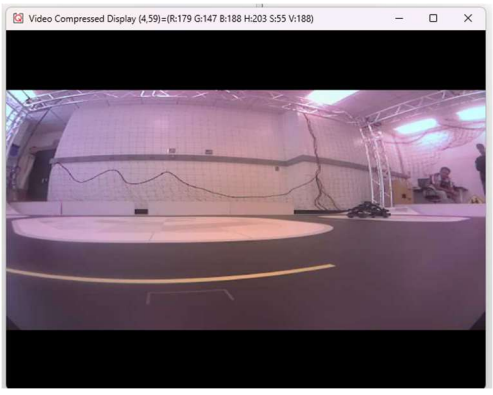
Activity 5: 360 Vision



Activity 5: CSI Camera 1



Activity 5: CSI Camera 2



Activity 5: CSI Camera 0

Activity 5: CSI Camera 3