ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Automated Guided Vehicle Wireless Charging System with DSP-Based Position-Adaptive Frequency Control

---

Team #14

Jingzhou Ding (jd47@illinois.edu)
Jinru Cai (jinruc2@illinois.edu)
Jiaxin Cao (jiaxin18@illinois.edu)
Yaxin Li (yaxinl2@illinois.edu)

TA: Yuhang Wang

May 2026

Abstract

This report presents an automated guided vehicle (AGV) wireless charging system that combines vision-assisted docking with position-adaptive wireless power transfer (WPT). The vehicle uses a Raspberry Pi 4B for navigation, USB camera processing, and ArUco marker recognition, while an STM32 microcontroller performs low-level motor control. The charging station uses an ESP32 to communicate with the Raspberry Pi over Wi-Fi and Transmission Control Protocol (TCP), then forwards alignment data over a wired link to a TMS320F28335 digital signal processor (DSP). The DSP parses the corrected two-dimensional position data, applies a high-resolution calibrated lookup table (LUT) with bilinear interpolation, and generates full-bridge pulse-width modulation (PWM) signals for the resonant power stage. The lithium battery charging target is 12.6 V at 2 A, or 25.2 W. A high-input trial near 15 V verified the available charging-power headroom but drew more than 3.5 A at the input, so routine testing was reduced to a 12 V input to protect battery lifetime, long-term reliability, and safety margin. Calibration data from 133 measured coil-offset points at 12 V input show optimal frequencies from 61.75 kHz to 66.42 kHz and measured output power from 15.70 W to 18.80 W.

Contents

1 Introduction

Wireless power transfer (WPT) is a useful charging method for electric vehicles and automated guided vehicles (AGVs) because it removes exposed plug contacts and allows charging to begin automatically when a vehicle docks. The main practical limitation is coil misalignment: when the transmitter (Tx) and receiver (Rx) coils are laterally offset, the magnetic coupling coefficient decreases and the converter efficiency drops [1], [2]. A previous generation of this project used a fixed-frequency charging system, so it could not compensate for the coupling changes that occur during realistic parking.

This project addresses that problem with two coordinated mechanisms. First, the AGV uses camera-based ArUco marker detection to estimate its pose relative to the charging station and reduce coil offset before charging. Second, the charging station uses DSP-based pulse frequency modulation (PFM) to adjust the CLLC resonant converter frequency after the remaining offset is known [3], [4]. The current implementation differs from the original design document by separating vehicle and station responsibilities more cleanly: the Raspberry Pi performs navigation and marker recognition on the vehicle, the STM32 performs vehicle motion control, and an ESP32 at the charging station bridges Wi-Fi/TCP data from the Raspberry Pi to the wired DSP control hardware.

1.1 Objective

The objective is to demonstrate an AGV that can drive to a charging station, align its onboard receiver coil with the station transmitter coil, and receive wireless charging power with frequency compensation for residual misalignment. The intended result is a safer and more autonomous charging process than plug-in charging, with better efficiency than a fixed-frequency WPT system under imperfect parking.

1.2 High-Level Requirements

1. **Wireless charging performance:** The battery-rated charging target is 12.6 V at 2 A, corresponding to 25.2 W. A high-input test near 15 V confirmed that the power stage has the headroom needed for the rated charging point, but the input current exceeded 3.5 A. For battery lifetime, long-term reliability, and safety, routine testing is therefore performed at 12 V input. DC-to-DC transfer efficiency shall remain at least 75% for lateral coil misalignment up to ± 2 cm when adaptive frequency control is active.
2. **Vision and parking accuracy:** The Raspberry Pi and camera shall estimate ArUco marker position with ± 5 mm lateral accuracy and $\pm 3^\circ$ angular accuracy at the operating distance. After precision parking, the remaining coil misalignment shall be no more than 1 cm.
3. **Adaptive frequency control:** The DSP shall adjust the CLLC switching frequency over the calibrated 62–67 kHz region based on received offset data, with firmware limits around the operating range for protection. After the Raspberry Pi finishes its

averaged final alignment measurement, the packet-transfer and DSP-update latency shall be no more than 50 ms.

4. **Output voltage stability:** The receiver output voltage ripple shall be no more than 0.5 V peak-to-peak during steady-state charging.

1.3 Block Diagram

Figure 1 shows the updated system architecture. Unlike the original design document, the Raspberry Pi no longer sends station-side frequency-control data directly to the DSP. Instead, the station ESP32 receives alignment data over Wi-Fi/TCP and forwards it to the DSP through a wired station-side link.

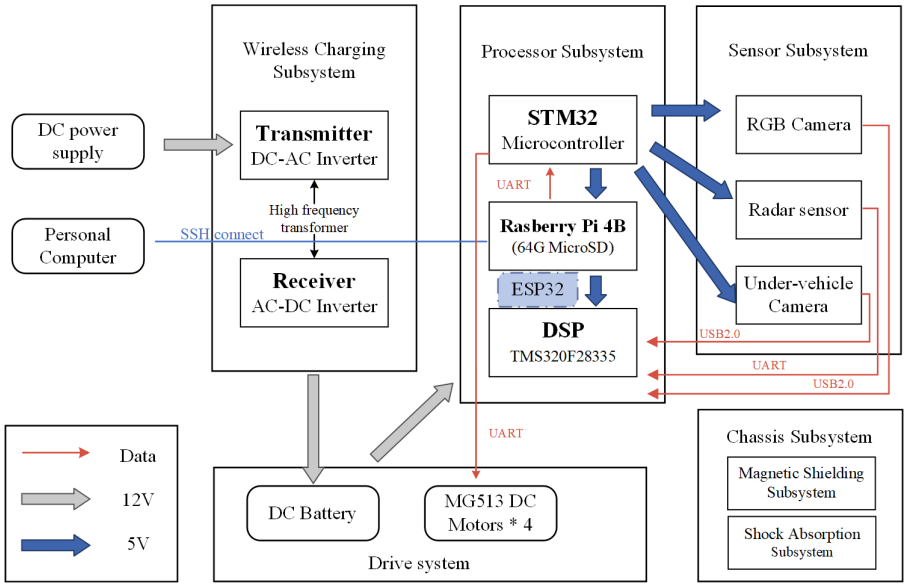


Figure 1: Overall AGV wireless charging system architecture, including the wireless charging, processor, sensor, drive, and chassis subsystems.

1.4 Project Changes

The main architectural change from the design document is the addition of an ESP32 at the charging station. The Raspberry Pi remains on the AGV and handles navigation, camera input, and ArUco marker recognition. The STM32 remains on the AGV and focuses on low-level motion control. The station-side ESP32 now handles the wireless TCP connection to the Raspberry Pi and relays alignment data to the DSP over a wired link, which reduces the need for the DSP to manage networking and keeps the high-frequency power stage control localized at the charging station.

2 Design

2.1 System Architecture

The final design is divided into two physical platforms. The AGV carries the Raspberry Pi, camera, STM32, motor driver, encoders, receiver coil, rectifier, and battery/load interface. The charging station contains the ESP32, DSP, gate drivers, full-bridge inverter, transmitter coil, and adjustable DC input supply. This split keeps navigation and motion control on the moving vehicle while keeping power conversion and frequency control at the fixed charging station.

Appendix C provides photographs of the implemented charging subsystem, STM32 controller, Raspberry Pi controller, and full AGV prototype.

The data path begins when the Raspberry Pi detects the ArUco marker mounted at the charging station. The Raspberry Pi estimates the lateral offset Δx, longitudinal offset Δy, and yaw error $\Delta\theta$ between the AGV receiver coil and the station transmitter coil [5], [6]. It then sends motion commands to the STM32 for precision parking and sends the final alignment data to the station ESP32 over a TCP socket. The ESP32 relays the received packet over a wired station-side link to the DSP. The DSP uses this offset to select a switching frequency for the CLLC converter.

2.2 Design Alternatives and Corrective Actions

Several design decisions changed during the semester as the team moved from the proposal architecture to the integrated prototype. Tables 1 and 2 summarize the main alternatives, the selected approach, and the corrective actions taken when an early approach was not robust enough for the final system. The main design pattern was to keep high-bandwidth sensing and navigation on the Raspberry Pi, time-critical actuation on the STM32, network handling on the ESP32, and switching control on the DSP.

2.3 Wireless Charging Subsystem

The wireless charging subsystem transfers power across an air gap to the AGV battery and load interface. The power stage uses a CLLC resonant converter because its gain changes with switching frequency [3]. This allows the controller to compensate for coupling changes caused by coil misalignment. The implemented controller uses 64 kHz as the baseline frequency, uses a calibrated LUT for position-adaptive operation, and applies firmware frequency limits for protection.

Figure 2 shows the charging-specific power and data paths. The Raspberry Pi on the intelligent car provides the final position information to the station ESP32 over Wi-Fi/TCP, and the ESP32 forwards the position information to the DSP. The DSP then controls the transmitter full bridge. Energy flows from the transmitter full bridge through the transmitter coil, across the wireless coupling gap, through the receiver coil and receiver full bridge, and finally to the vehicle battery.

Table 1: Power and control design alternatives.

Design Issue	Alternatives Considered	Selected Approach	Justification or Corrective Action
WPT frequency control	Fixed 64 kHz operation; manual sweep; position-adaptive PFM.	DSP-based position-adaptive PFM using a calibrated LUT.	Fixed-frequency operation could not compensate for coupling changes. The LUT maps corrected position to switching frequency while preserving a 64 kHz fallback.
Position-to-frequency mapping	Sparse measured-point lookup; one-dimensional fit; two-dimensional polynomial fit; dense LUT interpolation.	RBF-generated 5 mm LUT with bilinear interpolation.	The measured calibration data vary in both directions, so the firmware uses a dense two-dimensional table. The current LUT has 0.1943 kHz MAE and $R^2 = 0.9498$.
Vehicle actuation	Raspberry Pi directly commands motor PWM; STM32 local motor loop.	Raspberry Pi sends velocity commands; STM32 closes the motor-speed loop.	The STM32 updates encoder-based PI control every 10 ms, while the Raspberry Pi remains responsible for navigation-level decisions.
Charging test input	Routine operation near 15 V input; reduced 12 V input for calibration.	12 V routine testing with one higher-input headroom trial.	The near-15 V trial showed rated-power headroom, but input current exceeded 3.5 A. The team returned to 12 V input to reduce thermal stress and battery-life risk while the draft verification remains incomplete.

Table 2: System integration and sensing design alternatives.

Design Issue	Alternatives Considered	Selected Approach	Justification or Corrective Action
Controller partitioning	Raspberry Pi directly drives DSP data path; DSP handles communication; ESP32 station bridge.	ESP32 receives Raspberry Pi TCP packets and forwards raw bytes to the DSP.	The bridge keeps the Wi-Fi/TCP stack off the DSP and keeps high-frequency PWM control local to the charging station.
Wireless protocol	UDP datagrams; TCP socket connection.	TCP connection from ESP32 client to Raspberry Pi server.	The data rate is low, but packet order and delivery matter for safe frequency updates. TCP reliability is more useful than UDP's lower overhead.
Vision pose estimation	Full PnP pose solve; calibrated image-plane offset.	Calibrated image-plane offset with yaw from the marker top edge.	The docking geometry is fixed, so the simpler method reduces processing complexity. Measured zero offsets of $x_0 = 5.01$ mm and $y_0 = 49.38$ mm are subtracted before transmission.

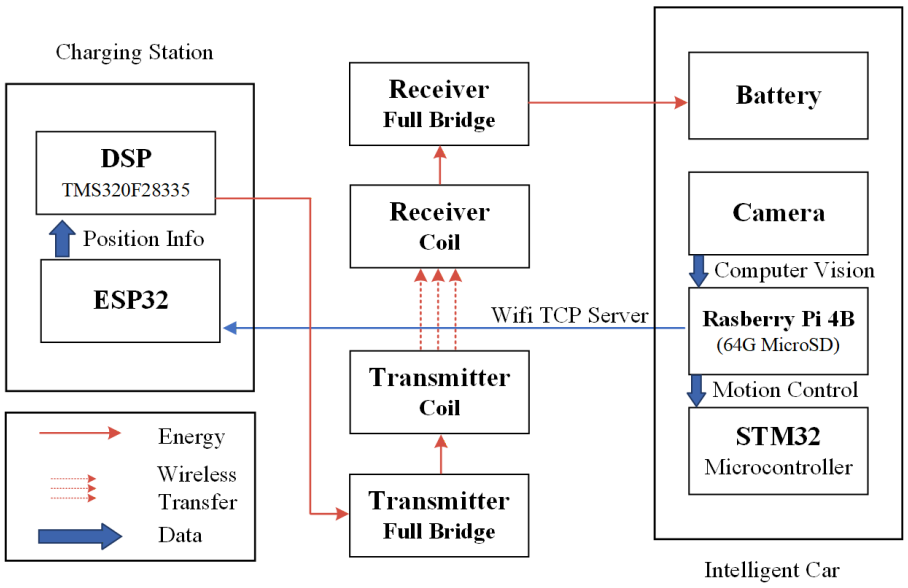


Figure 2: Wireless charging subsystem architecture with station-side DSP control, ESP32 communication, coil-to-coil wireless transfer, and vehicle-side battery charging path.

Table 3 summarizes the measured coil and compensation-component values. The transmitter and receiver coils have similar inductance, with $L_p = 168.34\ \mu$H on the transmitter side and $L_s = 159.75\ \mu$H on the receiver side, which is consistent with an approximately 1:1 coil turns ratio. The measured series-aiding and series-opposing inductances were $L_+ = 596.4\ \mu$H and $L_- = 57.4\ \mu$H. For coupled coils,

$$M = \frac{L_+ - L_-}{4}, \qquad k = \frac{M}{\sqrt{L_p L_s}}, \tag{1}$$

which gives a measured mutual inductance of $M = 134.75\ \mu$H and coupling coefficient $k = 0.822$ at the tested alignment. The average value $(L_+ + L_-)/2 = 326.9\ \mu$H is close to the separately measured sum $L_p + L_s = 328.09\ \mu$H, which supports the consistency of the coupled-inductor measurement.

Using the measured coil inductances and compensation capacitors, the simple single-side LC estimates are

$$f_p = \frac{1}{2\pi\sqrt{L_p C_p}} = 85.26\ \text{kHz}, \qquad f_s = \frac{1}{2\pi\sqrt{L_s C_s}} = 85.09\ \text{kHz}. \tag{2}$$

These values show that the transmitter and receiver compensation networks are closely matched. However, the single-side LC resonance does not directly determine the final operating frequency because the two resonators are strongly coupled. With a high measured coupling coefficient, the coupled system exhibits frequency splitting: the original single-resonator peak separates into a lower-frequency mode and a higher-frequency mode.

Table 3: Measured coil and compensation parameters.

Parameter	Value	Notes
Transmitter coil inductance, L_p	168.34 μH	Charging-station Tx coil.
Receiver coil inductance, L_s	159.75 μH	AGV Rx coil.
Series-aiding inductance, L_+	596.4 μH	Used to estimate mutual inductance.
Series-opposing inductance, L_-	57.4 μH	Used to estimate mutual inductance.
Tx compensation capacitor, C_p	20.7 nF	Measured transmitter-side capacitor.
Rx compensation capacitor, C_s	21.9 nF	Measured receiver-side capacitor.
Estimated mutual inductance, M	134.75 μH	$(L_+ - L_-)/4$.
Estimated coupling coefficient, k	0.822	At the tested coil alignment.

For two closely matched resonators, the approximate split frequencies can be interpreted as

$$f_{\text{low}} \approx \frac{f_0}{\sqrt{1+k}}, \qquad f_{\text{high}} \approx \frac{f_0}{\sqrt{1-k}}. \tag{3}$$

Using $f_0 \approx 85.2$ kHz and $k = 0.822$, the lower split mode is approximately 63 kHz, which is consistent with the experimentally selected DSP operating region near 62–67 kHz. Therefore, the 85 kHz value in Equation (2) describes the isolated component-level LC tanks, while the adaptive control range follows the lower resonance peak of the loaded, strongly coupled WPT system.

The lithium battery rated charging point is 12.6 V at 2 A, so the rated charging power is

$$P_{\text{rated}} = 12.6\ \text{V} \times 2\ \text{A} = 25.2\ \text{W}. \tag{4}$$

During the reported calibration measurements, the power-stage input was held at 12 V to reduce thermal and electrical risk while validating the position-to-frequency behavior. The measured output power therefore remained below the full battery-rated charging power. The team also performed a higher-input test near 15 V, which confirmed the available headroom for the rated charging point. However, the input current exceeded 3.5 A

at this operating condition, increasing thermal stress and battery-life risk. For long-term operation and safety margin, the system was returned to 12 V input for routine testing and data collection.

The DSP generates PWM signals for the gate drivers and updates the timer period when a new frequency command is selected. If f_{clk} is the DSP timer clock and f_{sw} is the target switching frequency, the timer period is set according to

$$\text{TBPRD} = \frac{f_{\text{clk}}}{f_{\text{sw}}} - 1. \tag{5}$$

The DSP should disable PWM output during faults such as overtemperature, invalid alignment packets, or output overcurrent.

2.4 Vision and Navigation Subsystem

The Raspberry Pi performs the high-level vehicle functions. It receives frames from a USB camera, detects ArUco marker corners using OpenCV, computes the relative coil offset, and decides when to transition from route following to marker-guided docking. In the updated architecture, this same Raspberry Pi also owns navigation decisions and sends motion commands to the STM32 during precision parking.

The implemented image-recognition node runs as the ROS node `aruco_docking_node`. It subscribes to the Boolean topic `/start_detect`; when the docking manager publishes `True`, the node opens camera 0 at 1280×720 resolution and searches for a marker from OpenCV's 4-by-4, 50-marker ArUco dictionary. The code supports both newer and legacy OpenCV ArUco detector APIs, which makes the same script usable across the OpenCV versions installed on the Raspberry Pi.

Rather than solving a full camera pose with PnP, the current implementation uses a calibrated image-plane offset. For the first detected marker, the Raspberry Pi averages the four marker-corner coordinates to obtain the marker center. It subtracts the image center to compute `dx_pixel` and `dy_pixel`, then multiplies by the measured scale factor 0.035 cm/pixel. The marker yaw estimate is computed from the angle of the top marker edge. Before transmission, the centimeter offsets are converted to millimeters and corrected by the measured visual zero offset, $x_0 = 5.01$ mm and $y_0 = 49.38$ mm:

$$x_{\text{corr}} = 10\Delta x_{\text{cm}} - 5.01, \qquad y_{\text{corr}} = 10\Delta y_{\text{cm}} - 49.38. \tag{6}$$

To reduce frame-to-frame noise, one detection trigger performs five independent samples. Each sample lasts up to 1.2 s and is accepted once at least eight valid marker frames have been collected. The node then averages the accepted sample means and publishes the corrected result on `/aruco_offset` as a `Vector3`, where x and y are corrected millimeter offsets and z is the marker angle in degrees. It also publishes text state messages such as `DETECTING`, `DONE`, `NO_MARKER`, and `ERROR` on `/aruco_status`. If no valid marker is found, the Raspberry Pi sends a marker-lost status frame instead of an OK frame.

The camera is mounted on the AGV and directed toward the charging station marker. No image frames need to leave the Raspberry Pi during normal operation. Only compact numeric data such as Δx, Δy, $\Delta\theta$, packet sequence number, and validity status are transmitted to other controllers. The node also writes a small text log and one debug image on the Raspberry Pi for field debugging when ROS console output is delayed.

2.5 Motion Control Subsystem

The STM32 microcontroller performs the low-level vehicle motion control [7]. The Raspberry Pi remains responsible for navigation and docking decisions, while the STM32 handles time-critical motor actuation, encoder feedback, PWM generation, and local safety behavior. During approach and precision docking, the Raspberry Pi publishes velocity commands on the ROS `/cmd_vel` topic. The ROS base node converts these commands to an 11-byte serial frame and sends them to the STM32 at 115200 baud.

The command frame contains the desired chassis velocity in `linear.x`, `linear.y`, and `angular.z`. The values are scaled by 1000 before transmission, giving 1 mm/s linear-speed resolution and 0.001 rad/s yaw-rate resolution. For the differential-drive docking motion used in this project, `linear.y` is kept at zero; forward and backward corrections use `linear.x`, while heading corrections use `angular.z`. The STM32 parses the command, converts it to left- and right-wheel targets, and applies the appropriate motor polarity and PWM output.

Wheel speed is regulated locally on the STM32 using encoder feedback. The firmware reads the encoder timer counts every 10 ms and updates an incremental PI speed controller,

$$PWM[k] = PWM[k-1] + K_P(e[k] - e[k-1]) + K_I e[k], \tag{7}$$

where $e[k]$ is the difference between measured and target wheel speed. The calculated PWM command is clamped before being written to the motor driver; the main control path limits the motor command to 6900 counts, and the PI controller includes a hard limit of 7200 counts. The STM32 also sends a 24-byte feedback frame to the Raspberry Pi containing chassis velocity, IMU data, battery voltage, frame markers, and an XOR checksum.

The motion-control path includes basic fail-safe behavior. Frames use fixed markers and an XOR checksum, so malformed packets are rejected instead of being interpreted as valid velocity commands. When the ROS base node shuts down, it sends a zero-velocity command before closing the serial port. The STM32 firmware also sets the motor PWM to zero under undervoltage conditions, and the docking logic publishes zero velocity when marker detection is lost or final parking tolerance is reached.

Table 4: Key motion-control implementation parameters.

Parameter	Value	Purpose
Serial baud rate	115200 baud	Raspberry Pi–STM32 communication.
Command frame length	11 bytes	Velocity command from ROS to STM32.
Feedback frame length	24 bytes	Odometry, IMU, and battery feedback.
Velocity scale factor	1000×	m/s to mm/s; rad/s to mrad/s.
Motor control period	10 ms	Encoder read and PI speed update.
Command wire time	0.95 ms	$11 \times 10/115200$.
Feedback wire time	2.08 ms	$24 \times 10/115200$.
Main PWM clamp	6900 counts	Prevents excessive motor command.
Hard PI clamp	7200 counts	Prevents PI windup and saturation.

2.6 Charging Station Communication and DSP Control

The ESP32 is the network bridge at the charging station. It connects to the Raspberry Pi through Wi-Fi and TCP, with the ESP32 acting as a TCP client for the Raspberry Pi server at `192.168.0.100:5000`. The Raspberry Pi script binds the server socket to `0.0.0.0:5000`, accepts ESP32 connections in a background thread, and replaces the old socket whenever a new ESP32 connection arrives. The ESP32 receives the Raspberry Pi data stream and transparently forwards the raw bytes to UART2 without modifying or rebuilding the frame. UART2 is configured as 115200 baud, 8N1, with GPIO17 as TX and GPIO16 as RX. This design keeps the TCP stack off the DSP and allows the DSP firmware to focus on deterministic PWM and protection logic.

TCP was selected instead of UDP because the transmitted alignment packets contain low-bandwidth but safety-critical control information. These packets include corrected marker offsets, packet sequence number, and status flags required by the downstream DSP frequency-control subsystem. Packet loss or out-of-order delivery could lead to incorrect resonant-frequency updates or invalid docking behavior. TCP provides reliable, ordered delivery, automatic retransmission, and built-in connection management while adding negligible bandwidth overhead for the compact packets used in this project. Since the system sends numerical alignment results rather than image streams, the TCP latency overhead is small compared with the overall docking cycle.

Another advantage of the TCP client-server structure is that the Raspberry Pi only needs to maintain a single persistent socket after the ESP32 client connects. Once connected,

the Raspberry Pi can transmit docking packets through the same socket without tracking dynamically changing ESP32 source ports, which simplifies the communication software and improves robustness during long-duration experiments. Table 5 summarizes the protocol tradeoff.

Table 5: Comparison between TCP and UDP for docking communication.

Feature	TCP	UDP
Reliable delivery	Yes	No
Ordered packets	Yes	No
Connection management	Yes	No
Packet retransmission	Yes	No
Docking-control data	Suitable	Limited
High-bandwidth streaming	Moderate	Excellent

Table 6 lists the 13-byte position frame sent by the Raspberry Pi and forwarded by the ESP32. The ESP32 code also parses the frame header, floating-point offsets, and XOR checksum for serial-monitor debugging; however, the control path remains transparent raw-byte forwarding from TCP to UART2. The firmware includes Wi-Fi reconnect, TCP reconnect, received-byte statistics, and LED connection-state indication. After a TCP disconnect, the ESP32 retries the Raspberry Pi server approximately every 2 s.

On the Raspberry Pi side, each result is sent as a short burst instead of a single packet. The script transmits the same final status and corrected offset at 30 Hz for 1 s while incrementing an 8-bit sequence number. This burst behavior was chosen because the ESP32-to-DSP UART path is one-way and does not provide an acknowledgement. The Raspberry Pi currently sends `STATUS=0` for a valid marker result and `STATUS=1` for marker lost; `STATUS=2` is reserved for link-down handling in the broader protocol.

Table 6: Position frame sent from Raspberry Pi to ESP32 and forwarded to the DSP.

Field	Format	Purpose
Header	`AA 55`	Two-byte frame marker.
SEQ	uint8	Sequence number for detecting dropped or repeated packets.
STATUS	uint8	0 = OK, 1 = marker lost, 2 = link down.
X	float32 little-endian	Corrected x offset in millimeters.
Y	float32 little-endian	Corrected y offset in millimeters.
XOR	uint8	XOR checksum of SEQ, STATUS, X, and Y body bytes.

On the DSP side, the F28335 implementation is based on TI's ePWM up-down action-qualifier example. The DSP receives the ESP32-forwarded position frame through SCI-C at 115200 baud. A byte-wise receive interrupt implements a state-machine parser for the `AA 55` header, sequence number, status byte, little-endian floating-point `x_mm` and `y_mm` fields, and XOR checksum. When the checksum passes and the status byte is OK, the firmware updates the latest valid position sample.

The latest firmware uses a high-resolution position-to-frequency LUT instead of the earlier polynomial formula. The table `FitLutKHz` is a 32×36 grid generated offline from the corrected Sheet2 calibration data using a multiquadric radial basis function fit with $\epsilon = 100$ and $\lambda = 10^{-4}$. Rows cover $y = -90$ mm to 65 mm, columns cover $x = -80$ mm to 95 mm, and both axes use 5 mm spacing.

For a corrected position (x, y), the DSP first checks that each coordinate is within ± 110 mm and that the radial offset $\sqrt{x^2 + y^2}$ is no more than 115 mm. Accepted coordinates are clipped to the LUT domain, converted to fractional table indices, and evaluated with bilinear interpolation:

$$\begin{aligned} f_{\mathrm{LUT}} = (1-\alpha)(1-\beta)F_{j,i} + \alpha(1-\beta)F_{j,i+1} \\ + (1-\alpha)\beta F_{j+1,i} + \alpha\beta F_{j+1,i+1}, \end{aligned} \tag{8}$$

where α and β are the fractional positions within the 5 mm grid cell. The interpolated frequency is clamped by the firmware frequency limits, currently 60–68 kHz, before it is applied to the ePWM period registers. If the position is invalid or stale, the controller returns to the 64 kHz baseline frequency.

The DSP configures ePWM1 and ePWM2 for full-bridge drive using up-down counting, 50% duty cycle, synchronized phase-shift control, and 500 ns dead time. ePWM1 is the reference bridge leg and ePWM2 is shifted by 180 degrees through the phase register. The commanded frequency is not applied as a step; instead, a 10 ms control loop applies a 0.05 kHz per tick slope limit so that the power stage tracks the new target gradually.

Fault handling is implemented through communication watchdogs, stale-result timeout, range checks, ePWM Trip Zone behavior, and a GPIO12-driven disable signal. During abnormal conditions, the DSP disables power output or returns to the 64 kHz baseline. The firmware also preserves CCS Watch manual-frequency mode and sweep mode for resonant-point debugging, power-stage response testing, and future recalibration.

3 Verification

3.1 Verification Procedure

Verification should demonstrate that the final system works as an integrated AGV charging platform rather than as isolated modules. The most important tests are charging power and efficiency, vision accuracy, docking accuracy, TCP-to-DSP communication latency, and output voltage stability.

3.2 Results

Table 7 summarizes the top-level verification plan and the current final-demo evidence. The results distinguish between global navigation to the charging-station area and residual coil-offset compensation through ArUco sensing and DSP frequency adaptation.

3.3 Discussion

The motion-control verification is included because the STM32 is the actuator layer for vehicle movement. The basic command-path test starts the base serial node, sends low-speed `/cmd_vel` commands, verifies wheel response, sends a zero-velocity command before disabling the vehicle, and checks that malformed serial frames are rejected. This command path is verified for the current draft. The ArUco stage is used as a residual-offset sensing stage rather than a closed-loop steering-correction stage, so final coil alignment is handled by measuring the remaining offset and adapting the DSP switching frequency.

The vision test should match the implemented Raspberry Pi behavior. A test fixture can place the marker at known image-plane offsets and angles, then trigger the ROS topic `/start_detect`. Each trigger should produce five sample windows of up to 1.2 s, with at least eight valid marker frames per accepted sample. The reported x and y values are checked after the zero-offset correction in Equation (6). The measured position offset error is no more than 0.5 mm, which satisfies the ± 5 mm position requirement. The no-marker case is also verified: stale marker-derived offsets are rejected, and the DSP uses the default 64 kHz output instead of applying an invalid adaptive update.

The latency requirement applies to the communication and DSP update path after the Raspberry Pi has produced its averaged alignment result. The recognition routine intentionally takes multiple samples for a steadier final docking measurement, so the complete trigger-to-result time is longer than the packet-transfer latency. For the communication test, the expected Raspberry Pi behavior is a 30 Hz, 1 s burst of 13-byte frames with monotonically increasing 8-bit sequence numbers. The final-result transfer latency is verified to meet the 50 ms requirement.

The communication verification is different from the original design document because the frequency-control path now crosses a Wi-Fi/TCP link before reaching the station-side ESP32 and DSP. ESP32-to-DSP residual-offset forwarding was verified during the

Table 7: Verification summary for the updated architecture.

Requirement	Test Method	Result
Support the 12.6 V, 2 A battery-rated charging point	Connect a load or battery interface; measure output voltage and current at 12 V input and during the high-input trial.	12 V calibration: 15.70–18.80 W. Near-15 V trial showed power headroom but input current exceeded 3.5 A.
Maintain at least 75% efficiency under ± 2 cm offset	Measure input and output power at known offsets with adaptive control enabled.	16–17 W output: about 60%. Resonant point: 1–2 W output with $>90\%$.
Vision accuracy within ± 5 mm and $\pm 3^\circ$	Compare Raspberry Pi offset and angle estimates with fixture measurements.	Offset error ≤ 0.5 mm.
Residual parking offset after navigation is characterized	Measure final Tx–Rx coil offset after LiDAR navigation reaches the charging-station area.	LiDAR parking is not direct coil-over-coil; residual offset is handled by ArUco measurement and DSP adaptation.
Motion-control command path is safe and repeatable	Send ROS `/cmd_vel` commands through the base node; verify wheel response, stopping behavior, and malformed-frame rejection.	Verified.
Final-result transfer latency no more than 50 ms	Timestamp Raspberry Pi result frame, ESP32 TCP receipt, wired forwarding, and DSP PWM update.	Verified.
Output ripple no more than 0.5 V peak-to-peak	Measure receiver output with an oscilloscope during charging.	366 mVpp steady-state; 3.76 Vpp full BW with probe/scope ringing.

Table 8: ROS commands used for motion-control bring-up.

Step	Command	Purpose
Start base	`roslaunch turn_on_wheeltec_robot turn_on_wheeltec_robot.launch car_mode:=mini_diff`	Opens the STM32 serial base node.
Move slowly	`rostopic pub -r 10 /cmd_vel geometry_msgs/Twist linear.x=0.05, angular.z=0.0`	Verifies low-speed wheel response.
Stop	`rostopic pub -1 /cmd_vel geometry_msgs/Twist linear.x=0.0, angular.z=0.0`	Sends a final zero-velocity command.

Table 9: Additional motion-control verification items.

Requirement	Verification method	Result
STM32 accepts valid docking commands and rejects malformed serial data.	Send valid `/cmd_vel` commands and intentionally corrupted serial frames; verify that valid frames move the wheels and corrupted frames do not update motor PWM.	Verified.
Motor response is repeatable for low-speed docking.	Command $v = 0.05$ m/s and $v = -0.05$ m/s for fixed time windows; measure distance traveled and final stopping error over at least five trials.	Verified for command response.
Yaw command path is controllable near the charger.	Command small positive and negative ω_z values; confirm that the STM32 motion path accepts low-speed angular commands.	Command path verified; ArUco stage is sensing-only.
Loss of valid marker or offset prevents stale WPT updates.	Publish a marker-lost state and verify that stale offset data are not applied to DSP adaptive frequency updates.	Verified; DSP falls back to 64 kHz.

frequency-sweep measurement sequence, with more than 90 consecutive measurement groups transmitted successfully from the ESP32 to the DSP. Stale-packet rejection and marker-lost behavior were also verified by confirming that the DSP returns to the default 64 kHz output when no valid marker-derived offset is available.

The corrected coil calibration data contain 133 measured offset points. One raw workbook entry was corrected from 54.4 kHz to 64.4 kHz before fitting and evaluation. The measured offsets span $x = -75.81$ mm to 93.40 mm and $y = -87.40$ mm to 61.40 mm. The measured optimal switching frequencies span 61.75–66.42 kHz, and the measured output powers span 15.70–18.80 W with an average of 17.01 W under the 12 V input calibration setup. These data do not represent the maximum battery charging capability; they were collected under the reduced input voltage selected for routine testing. The efficiency measurement showed a tradeoff between delivered power and transfer efficiency: when the system was tuned to maintain about 16–17 W output power, the measured efficiency was about 60%; when the frequency was adjusted back to the design resonance point, the efficiency increased to above 90%, but output power dropped to about 1–2 W. Therefore, the 75% efficiency requirement was not simultaneously met with the higher-power operating point in the present setup. The battery-rated charging point is 25.2 W. A high-input trial near 15 V showed that the system has the necessary charging-power headroom, but the input current exceeded 3.5 A. The team therefore returned to 12 V input to reduce battery aging, thermal loading, and long-term safety risk. The measured steady-state V_{out} ripple is 366 mVpp after excluding switching spikes and ringing, so the 0.5 Vpp steady-state ripple requirement is met. The full-bandwidth oscilloscope measurement reports 3.76 Vpp ripple and noise; the high-frequency ringing is attributed to interference from the oscilloscope and probe setup. For the current DSP LUT with 5 mm bilinear interpolation, the fitting evaluation gives 0.1943 kHz mean absolute error, 0.2661 kHz root-mean-square error, 0.5585 kHz 95th-percentile absolute error, 0.9780 kHz maximum absolute error, and $R^2 = 0.9498$. Appendix B includes the fitting-method comparison and a compact calibration-data table.

4 Costs

4.1 Parts and Labor Costs

The project reuses several major components from the previous generation or existing lab inventory, including the AGV chassis, motors, encoders, H-bridge motor driver, STM32 development board, TMS320F28335 DSP board, ESP32 module, and 12 V AGV battery. The two Raspberry Pi boards were newly purchased for this project rather than inherited from the previous design. Table 10 lists both the estimated retail value and the amount paid by the team or department. Lab-owned and reused parts are listed with a paid cost of \$0.00 so that the implementation cost and replacement value are both visible.

The reimbursement document lists parts purchases in RMB. These values were converted to U.S. dollars using an exchange rate of

$$1 \text{ USD} = 6.803 \text{ RMB}. \tag{9}$$

Thus, the parts-cost conversion is

$$C_{\text{USD}} = \frac{C_{\text{RMB}}}{6.803}. \tag{10}$$

The ECE 445 guideline recommends estimating labor cost for each partner as

$$\text{labor cost} = \text{ideal hourly salary} \times \text{actual hours spent} \times 2.5. \tag{11}$$

For this draft estimate, Table 11 uses an ideal engineering salary of \$38.00/h and 50 h per partner. These hours should be replaced with the final individual time records before submission if the team keeps a more detailed log.

Table 12 summarizes the resulting project cost. The paid total is the amount paid by the team or department, while the retail-equivalent total also counts reused lab-owned parts at their estimated replacement value.

No machine shop cost is expected. If a charging station enclosure is needed, it can be 3D printed through department resources. The current prototype is not intended for immediate mass production, so the cost estimate does not include a bulk-purchase or manufacturing-volume projection.

4.2 Schedule

Table 13 summarizes the semester schedule by week and team member. The schedule is written as a draft record of the work distribution used for the current report version; it should be updated if the final experiment dates or individual responsibilities change before submission.

Table 10: Draft parts cost with retail value and amount paid.

Item	Retail Cost (USD)	Paid Cost (USD)	Source or Status
Power-electronics components, connectors, and shipping from LCSC receipts	$94.59	$94.59	Purchased
Transmitter PCB fabrication	$7.44	$7.44	Purchased
Receiver PCB fabrication	$4.76	$4.76	Purchased
Raspberry Pi 4B, 8 GB development board	$126.56	$126.56	Purchased
Raspberry Pi 3B kit for development and backup testing	$53.81	$53.81	Purchased
USB-to-TTL serial adapter	$1.43	$1.43	Purchased
Raspberry Pi USB camera	$2.67	$2.67	Purchased
USB HD camera module	$13.23	$13.23	Purchased
AGV chassis, motors, encoders, H-bridge motor driver, STM32 board, and 12 V battery	$300.00	$0.00	Reused lab inventory
TMS320F28335 DSP board	$180.00	$0.00	Reused lab inventory
ESP32 station communication module	$8.00	$0.00	Reused lab inventory
Wiring, headers, fasteners, and mechanical mounting hardware	$20.00	$0.00	Lab stock
Parts subtotal	$812.50	$304.50	

Table 11: Draft labor cost estimate by team member.

Team Member	Hourly Rate	Hours	Labor Cost
Jingzhou Ding	$38.00/h	50	$4,750.00
Jinru Cai	$38.00/h	50	$4,750.00
Jiaxin Cao	$38.00/h	50	$4,750.00
Yaxin Li	$38.00/h	50	$4,750.00
Labor subtotal		200	$19,000.00

Table 12: Draft project cost summary.

Category	Paid Cost	Retail-Equivalent Cost
Parts subtotal	$304.50	$812.50
Labor subtotal	$19,000.00	$19,000.00
Grand total	$19,304.50	$19,812.50

Table 13: Draft weekly project schedule by team member.

Week	Jingzhou Ding	Jinru Cai	Jiaxin Cao	Yaxin Li
1	Reviewed DSP and WPT goals.	Audited coils and power hardware.	Audited AGV motion hardware.	Reviewed navigation requirements.
2	Defined adaptive-frequency control path.	Selected compensation and PCB direction.	Reviewed STM32 serial and motor code.	Defined Raspberry Pi software tasks.
3	Set up DSP ePWM baseline.	Prepared power-stage component list.	Tested low-speed motor commands.	Brought up camera and ArUco detection.
4	Defined packet fields for DSP parser.	Laid out transmitter and receiver boards.	Checked encoder feedback path.	Built ROS detection trigger flow.
5	Implemented SCI receive parser.	Assembled and inspected PCBs.	Tuned PI speed-control behavior.	Calibrated image-plane offset scale.
6	Added PWM update and clamp logic.	Brought up low-input WPT test.	Verified serial frame checks.	Prototyped TCP server on Raspberry Pi.
7	Added stale-packet fallback.	Measured coil inductance and capacitors.	Tested docking-speed limits.	Integrated ESP32 bridge link.
8	Tested manual frequency sweep.	Collected preliminary power data.	Integrated marker-lost stop behavior.	Added result burst transmission.
9	Generated high-resolution frequency LUT.	Collected coil-offset calibration data.	Supported AGV docking trials.	Logged corrected marker outputs.
10	Integrated LUT interpolation into DSP firmware.	Compared 12 V and near-15 V input tests.	Checked command shutdown behavior.	Tested TCP reconnect behavior.
11	Verified frequency clamp and slope limit.	Measured output power and efficiency tradeoff.	Prepared repeatability test method.	Prepared latency test method.
12	Integrated station-side controller.	Prepared ripple and thermal tests.	Integrated AGV parking sequence.	Integrated vision output with packet sender.
13	Analyzed LUT interpolation error.	Documented measured power-stage data.	Documented motion-control parameters.	Documented communication protocol.
14	Drafted DSP and verification sections.	Drafted WPT and cost data.	Drafted motion and safety material.	Drafted vision and communication sections.
15	Prepared final DSP test updates.	Prepared final charging test updates.	Prepared final docking test updates.	Prepared final vision test updates.

5 Conclusions

The updated prototype separates the system into vehicle-side autonomy and station-side charging control. The Raspberry Pi handles navigation and ArUco recognition, the STM32 handles motor control, the ESP32 bridges Wi-Fi/TCP data at the station, and the DSP controls the CLLC converter frequency.

At the current draft stage, the strongest verified result is the position-adaptive frequency-control path. The corrected WPT calibration data contain 133 measured coil-offset points at 12 V input. The measured optimal switching frequency spans 61.75–66.42 kHz, the output power spans 15.70–18.80 W, and the firmware LUT has a 0.1943 kHz mean absolute error against the measured optimal frequencies. A higher-input trial near 15 V showed headroom toward the 25.2 W battery-rated charging target, but input current exceeded 3.5 A. The 75% efficiency requirement has not yet been verified at the higher-power operating point: 16–17 W operation produced about 60% efficiency, while the high-efficiency resonant point delivered only 1–2 W.

5.1 Remaining Uncertainties

Remaining work includes final docking error, station forwarding, marker-loss, and fault tests. If rated-operation targets remain unmet, likely fixes include retuning the network, reducing coil spacing, improving parking, revising the operating range, or separating the power and efficiency requirements.

5.2 Future Work

Future work should first complete the remaining verification tests with quantitative pass or fail data. A useful hardware improvement is a small data screen for live charging status. The PCB already reserves a sampling-chip interface, so a future revision can sample voltage and current, compute $P = VI$, and display voltage, current, power, frequency command, alignment status, fault state, and real-time efficiency. The same feedback path could later support current limiting, overcurrent shutdown, and automatic derating.

5.3 Ethics and Safety

This project follows the IEEE Code of Ethics by prioritizing safety, truthful reporting, and responsible use of autonomous and wireless power technology [8]. This draft reports unmet efficiency and incomplete verification items rather than presenting the prototype as fully validated. Personnel should stay away from active coils, field exposure near 64 kHz should be checked against relevant guidance when possible [9], and the transmitter should disable PWM for invalid, stale, or out-of-range packets. Broader impacts include reduced connector wear, operator effort, and wasted energy.

Appendix A Requirement and Verification Table

Table 14 expands the main verification summary with concrete pass criteria and current final-demo evidence.

Table 14: Detailed requirement and verification table.

Requirement	Verification Method	Result
Battery-rated charging point is 12.6 V at 2 A.	Measure voltage and current; compute output power at 12 V input and during high-input trial.	Verified
Higher input should remain safe for long-term operation.	Raise input from 12 V to about 15 V and monitor input current and thermal behavior.	$>$3.5 A; 12 V used
Efficiency is at least 75% at offsets up to $\pm$2 cm.	Set known coil offsets; measure input and output power.	16–17 W: $\sim$60%; 1–2 W: $>$90%
Marker estimate is within $\pm$5 mm and $\pm3^\circ$.	Trigger `/start_detect` at known marker offsets; compare corrected ArUco estimates with fixture positions.	Offset error $\leq$0.5 mm
Residual parking offset after navigation is characterized.	Measure final transmitter-receiver coil offset after LiDAR navigation reaches the charging-station area.	ArUco + DSP
STM32 motion-control commands are safe and repeatable.	Send low-speed `/cmd_vel` commands; measure wheel response, stopping error, and malformed-frame rejection.	Verified
Loss of valid marker or offset prevents stale WPT updates.	Publish marker-lost state and verify that stale offset data are not applied to DSP adaptive frequency updates.	Verified; 64 kHz fallback
ESP32 reliably forwards alignment data to DSP.	Forward residual offset packets during the frequency-sweep measurement sequence and verify DSP-side reception.	Verified; $>$90 groups
Final-result transfer latency is no more than 50 ms.	Timestamp Raspberry Pi result frame, TCP receipt, UART forwarding, and PWM update.	Verified
Output ripple is no more than 0.5 V peak-to-peak.	Measure receiver output with oscilloscope.	366 mVpp steady; 3.76 Vpp full BW noise
Faults cause safe shutdown or hold behavior.	Inject marker-lost or invalid offset conditions and verify update rejection or fallback behavior.	Verified; 64 kHz fallback

Appendix B Frequency Mapping and Calibration

The corrected Sheet2 data contain measured output power and optimal switching frequency for 133 coil-offset positions. One raw data-entry value was corrected from 54.4 kHz to 64.4 kHz before fitting. These data were used to generate and compare the DSP's calibrated high-resolution LUT. Table 15 summarizes the frequency-mapping test results. Table 16 lists the corrected raw measurement data, and Tables 17–19 list the current DSP LUT used by the firmware.

Table 15: Frequency mapping method comparison. Error metrics are in kHz.

Method	MAE	RMSE	P95 $\lvert e \rvert$	Max $\lvert e \rvert$	R^2
Current DSP LUT, 5 mm bilinear	0.1943	0.2661	0.5585	0.9780	0.9498
RBF continuous model	0.1946	0.2669	0.5656	0.9784	0.9495
Old DSP quadratic formula	0.2707	0.3649	0.6985	1.5337	0.9056
Refit 2nd-order polynomial	0.2011	0.2852	0.5889	1.1398	0.9423
Refit 3rd-order polynomial	0.1951	0.2789	0.5475	1.0971	0.9448
Refit 4th-order polynomial	0.1818	0.2627	0.6371	0.9799	0.9510
IDW interpolation LOOCV	0.3743	0.5156	0.8917	2.3765	0.8115

Table 16: Corrected raw coil offset calibration data. Offsets are in mm, output power is in W, and optimal frequency is in kHz.

No.	x	y	P_o	f_o	No.	x	y	P_o	f_o	No.	x	y	P_o	f_o
1	-75.81	51.01	15.80	66.37	46	-10.91	-26.42	17.18	62.25	91	46.22	-31.80	17.35	62.72
2	-74.67	40.40	16.00	65.85	47	-9.46	20.01	17.15	62.43	92	46.23	-36.28	17.33	62.97
3	-71.10	49.36	16.10	66.10	48	-8.12	-37.93	17.22	62.40	93	47.18	56.78	16.20	64.57
4	-70.42	27.25	16.10	65.35	49	-7.46	52.74	17.70	63.37	94	48.38	-58.12	17.08	63.70
5	-69.63	47.84	15.90	65.92	50	1.73	-6.86	17.17	61.88	95	48.62	-9.67	17.25	62.45
6	-66.50	-67.13	16.47	66.20	51	6.51	-5.74	17.26	61.85	96	48.68	-2.11	17.10	63.00
7	-64.44	-63.85	16.65	65.90	52	7.52	-26.86	17.11	62.10	97	51.82	-30.43	17.40	62.95
8	-64.26	-57.58	16.64	65.50	53	8.03	47.81	17.41	63.30	98	52.17	-60.42	17.20	63.55
9	-63.55	-48.66	16.69	65.10	54	9.34	0.07	16.90	61.75	99	53.50	-27.97	17.10	62.77
10	-62.37	-82.87	16.20	66.20	55	10.43	-1.93	16.70	62.20	100	53.51	-16.21	17.24	62.73
11	-60.65	-87.40	16.20	66.42	56	12.53	31.41	16.89	62.38	101	55.60	-71.71	16.40	63.97
12	-60.20	-47.74	16.70	64.90	57	13.50	-27.69	17.18	61.95	102	55.77	-51.33	16.77	64.15
13	-58.23	-64.84	16.67	64.40	58	13.63	21.56	16.90	62.20	103	56.15	-50.77	16.99	63.70
14	-57.88	39.57	16.10	64.97	59	14.61	23.49	16.90	62.13	104	56.17	31.16	17.45	63.18
15	-54.91	56.69	16.45	64.98	60	15.17	-4.25	17.26	61.82	105	56.75	-31.38	17.67	63.15
16	-54.73	-42.24	16.79	64.40	61	17.09	-5.31	17.11	61.77	106	57.57	-52.59	16.85	63.85
17	-52.24	-72.59	16.40	65.22	62	17.66	-17.98	16.94	61.90	107	58.03	-13.93	17.34	62.85
18	-50.28	51.20	16.55	64.55	63	18.51	-41.93	16.90	62.42	108	58.82	-55.84	18.00	63.30
19	-50.01	-38.51	16.90	64.10	64	18.56	40.63	16.90	62.68	109	59.76	-24.59	17.20	62.77
20	-46.98	-80.12	16.30	65.35	65	19.70	-3.89	17.25	61.82	110	60.03	-41.92	18.80	63.45
21	-45.85	51.89	16.50	64.48	66	19.78	23.33	16.90	62.35	111	63.18	-31.41	17.30	62.97
22	-45.81	37.92	16.30	64.35	67	21.49	47.79	17.26	63.18	112	63.58	-69.62	16.85	63.92
23	-44.16	-33.64	17.70	63.60	68	22.59	-4.10	16.89	62.25	113	65.44	-34.60	17.69	63.57
24	-41.34	-60.67	16.40	64.30	69	22.89	-7.83	16.92	61.83	114	65.67	-20.04	17.30	62.95
25	-35.45	-34.47	18.59	63.30	70	25.90	32.39	16.90	62.40	115	71.86	61.40	15.70	65.70
26	-35.24	39.58	17.40	63.43	71	26.46	-6.12	17.24	61.90	116	73.29	-5.23	16.10	63.60
27	-34.52	43.06	16.20	64.00	72	26.80	12.63	16.97	62.93	117	73.46	-22.92	18.40	63.35
28	-32.99	-51.98	18.00	63.37	73	26.99	-7.14	16.20	62.40	118	84.68	5.96	16.00	64.60
29	-30.55	-71.58	16.50	64.42	74	27.85	-42.48	17.00	62.50	119	88.96	37.85	15.90	65.70
30	-28.40	57.00	16.80	64.00	75	30.69	-48.33	16.50	63.88	120	91.87	-27.96	16.00	64.70
31	-26.19	57.93	16.00	64.45	76	31.53	-5.69	17.22	62.00	121	93.40	-26.45	16.30	64.80
32	-24.84	39.94	17.70	63.40	77	32.10	-59.28	17.81	63.35	122	-41.82	49.33	16.65	64.30
33	-23.68	-31.89	17.23	62.80	78	32.21	-32.40	17.23	62.33	123	-50.57	-28.72	17.11	63.87
34	-23.59	-37.40	17.10	62.85	79	32.59	41.72	16.99	62.78	124	-35.14	-43.22	18.32	63.47
35	-23.10	27.37	17.36	62.85	80	34.32	-20.10	17.15	62.20	125	-32.70	-55.77	16.90	63.95
36	-22.56	-27.58	17.40	62.80	81	37.11	-41.33	17.39	62.87	126	-38.96	-66.07	16.74	64.62
37	-20.30	12.66	17.30	62.40	82	37.66	13.08	16.92	62.18	127	62.41	25.30	17.76	63.30
38	-20.02	-61.96	17.80	63.35	83	38.15	41.62	17.07	62.95	128	50.26	44.11	18.70	63.55
39	-18.70	8.20	17.20	62.50	84	39.91	-47.54	17.76	63.15	129	33.70	50.38	17.90	63.35
40	-16.35	19.42	17.28	62.30	85	40.49	-8.38	17.23	62.17	130	24.22	60.69	17.68	63.40
41	-16.33	32.45	17.21	63.93	86	40.59	-38.40	17.00	62.60	131	-60.71	-31.54	16.90	64.65
42	-14.99	-54.33	17.50	63.10	87	41.34	54.13	18.48	62.70	132	-50.16	-66.52	16.75	65.55
43	-14.03	60.15	16.20	64.17	88	41.91	-39.53	17.09	62.72	133	-67.34	35.18	16.63	64.95
44	-13.70	-31.46	16.90	62.50	89	44.18	33.48	16.99	62.85					
45	-13.54	10.11	17.20	62.75	90	44.79	-18.69	17.20	62.43					

Tables 17–19 list the current DSP firmware LUT `FitLutKHz`. Values are switching frequency commands in kHz rounded to 0.001 kHz. The table rows are $y = -90$ mm to 65 mm, columns are $x = -80$ mm to 95 mm, and both axes use 5 mm spacing.

Table 17: Current DSP LUT, $x = -80$ mm to -25 mm. Values are in kHz.

$y \backslash x$	−80	−75	−70	−65	−60	−55	−50	−45	−40	−35	−30	−25
−90	67.307	67.032	66.843	66.730	66.681	66.683	66.717	66.765	66.808	66.828	66.811	66.746
−85	67.148	66.802	66.538	66.352	66.233	66.169	66.144	66.140	66.139	66.123	66.079	65.996
−80	67.093	66.674	66.336	66.075	65.884	65.752	65.665	65.606	65.559	65.507	65.433	65.330
−75	67.124	66.634	66.223	65.889	65.626	65.427	65.279	65.167	65.075	64.985	64.883	64.757
−70	67.219	66.661	66.180	65.777	65.447	65.184	64.979	64.817	64.683	64.559	64.429	64.284
−65	67.351	66.730	66.185	65.717	65.326	65.007	64.751	64.546	64.375	64.222	64.070	63.908
−60	67.492	66.814	66.210	65.686	65.242	64.875	64.577	64.337	64.139	63.964	63.796	63.622
−55	67.613	66.884	66.231	65.659	65.171	64.766	64.437	64.173	63.956	63.768	63.592	63.412
−50	67.693	66.919	66.223	65.613	65.092	64.660	64.311	64.032	63.807	63.617	63.440	63.263
−45	67.713	66.900	66.170	65.531	64.987	64.539	64.180	63.898	63.675	63.490	63.322	63.153
−40	67.663	66.819	66.063	65.403	64.846	64.392	64.033	63.757	63.544	63.372	63.219	63.065
−35	67.545	66.676	65.900	65.229	64.667	64.214	63.864	63.600	63.404	63.252	63.119	62.984
−30	67.367	66.479	65.691	65.015	64.455	64.011	63.674	63.429	63.253	63.123	63.012	62.900
−25	67.144	66.243	65.450	64.775	64.224	63.793	63.473	63.248	63.094	62.986	62.898	62.807
−20	66.897	65.990	65.198	64.530	63.991	63.576	63.275	63.070	62.937	62.849	62.781	62.709
−15	66.651	65.744	64.958	64.302	63.778	63.380	63.097	62.910	62.794	62.722	62.669	62.613
−10	66.426	65.526	64.752	64.111	63.604	63.224	62.958	62.786	62.682	62.621	62.577	62.530
−5	66.244	65.357	64.599	63.976	63.488	63.125	62.873	62.712	62.615	62.558	62.517	62.473
0	66.117	65.248	64.511	63.910	63.441	63.094	62.854	62.699	62.604	62.545	62.501	62.453
5	66.054	65.208	64.496	63.917	63.468	63.137	62.906	62.753	62.655	62.589	62.536	62.478
10	66.054	65.236	64.551	63.997	63.567	63.250	63.025	62.872	62.768	62.691	62.623	62.552
15	66.112	65.325	64.668	64.139	63.729	63.424	63.205	63.049	62.936	62.845	62.760	62.672
20	66.216	65.460	64.832	64.328	63.938	63.644	63.429	63.269	63.145	63.039	62.937	62.831
25	66.352	65.627	65.027	64.547	64.174	63.891	63.678	63.514	63.380	63.259	63.140	63.017
30	66.501	65.806	65.233	64.774	64.416	64.142	63.932	63.764	63.621	63.488	63.354	63.216
35	66.647	65.979	65.429	64.989	64.645	64.379	64.171	64.000	63.850	63.707	63.563	63.416
40	66.778	66.132	65.602	65.177	64.844	64.584	64.377	64.205	64.051	63.903	63.755	63.603
45	66.881	66.254	65.739	65.325	65.000	64.744	64.540	64.368	64.214	64.066	63.919	63.771
50	66.954	66.339	65.834	65.428	65.108	64.856	64.653	64.484	64.333	64.191	64.052	63.916
55	66.995	66.388	65.889	65.487	65.169	64.919	64.719	64.554	64.411	64.281	64.158	64.041
60	67.011	66.408	65.910	65.509	65.191	64.942	64.746	64.588	64.457	64.343	64.243	64.154
65	67.012	66.408	65.910	65.507	65.188	64.940	64.749	64.600	64.484	64.393	64.322	64.267

Table 18: Current DSP LUT, $x = -20$ mm to 35 mm. Values are in kHz.

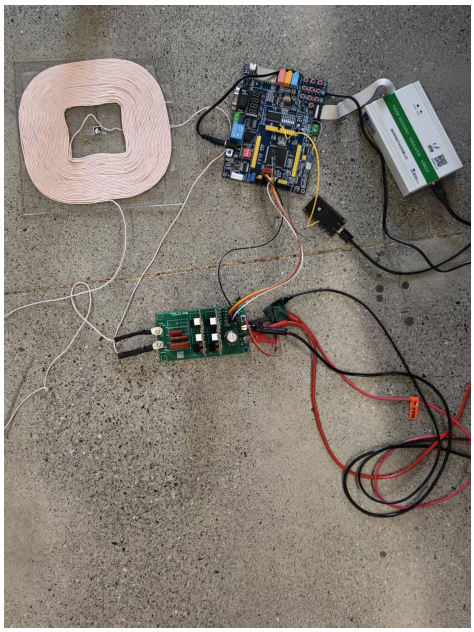
$y \backslash x$	−20	−15	−10	−5	0	5	10	15	20	25	30	35
−90	66.630	66.465	66.261	66.029	65.786	65.548	65.327	65.130	64.958	64.804	64.656	64.496
−85	65.870	65.705	65.509	65.296	65.085	64.890	64.725	64.598	64.506	64.441	64.387	64.322
−80	65.191	65.022	64.830	64.632	64.446	64.289	64.174	64.108	64.090	64.106	64.139	64.163
−75	64.605	64.428	64.238	64.050	63.883	63.756	63.682	63.669	63.713	63.800	63.909	64.011
−70	64.117	63.933	63.741	63.558	63.404	63.298	63.256	63.284	63.377	63.522	63.692	63.859
−65	63.729	63.537	63.342	63.160	63.013	62.921	62.900	62.956	63.085	63.271	63.487	63.702
−60	63.435	63.237	63.038	62.855	62.711	62.625	62.614	62.686	62.835	63.046	63.291	63.537
−55	63.222	63.021	62.820	62.635	62.489	62.403	62.394	62.469	62.626	62.846	63.102	63.362
−50	63.074	62.875	62.674	62.488	62.338	62.247	62.232	62.301	62.452	62.667	62.920	63.179
−45	62.973	62.780	62.582	62.396	62.242	62.144	62.118	62.174	62.309	62.508	62.746	62.991
−40	62.899	62.717	62.526	62.342	62.186	62.079	62.040	62.078	62.192	62.367	62.581	62.805
−35	62.836	62.669	62.488	62.310	62.153	62.039	61.987	62.006	62.096	62.244	62.431	62.629
−30	62.771	62.621	62.455	62.284	62.130	62.013	61.949	61.951	62.018	62.140	62.299	62.472
−25	62.699	62.568	62.416	62.257	62.109	61.991	61.921	61.909	61.957	62.057	62.193	62.343
−20	62.620	62.506	62.370	62.224	62.084	61.970	61.898	61.879	61.914	61.998	62.116	62.251
−15	62.539	62.440	62.319	62.186	62.058	61.952	61.883	61.862	61.891	61.965	62.074	62.201
−10	62.466	62.378	62.270	62.151	62.035	61.940	61.878	61.861	61.890	61.962	62.068	62.193
−5	62.413	62.332	62.233	62.125	62.022	61.939	61.888	61.879	61.913	61.988	62.096	62.225
0	62.391	62.312	62.218	62.119	62.028	61.956	61.917	61.917	61.959	62.040	62.153	62.288
5	62.409	62.327	62.234	62.140	62.057	61.995	61.966	61.975	62.025	62.112	62.230	62.371
10	62.471	62.381	62.285	62.192	62.114	62.059	62.037	62.052	62.106	62.196	62.317	62.460
15	62.577	62.474	62.372	62.277	62.199	62.147	62.127	62.144	62.198	62.286	62.403	62.543
20	62.719	62.604	62.492	62.391	62.310	62.256	62.234	62.247	62.294	62.373	62.480	62.610
25	62.889	62.761	62.639	62.531	62.444	62.384	62.355	62.358	62.392	62.455	62.545	62.659
30	63.076	62.936	62.805	62.690	62.596	62.528	62.487	62.476	62.492	62.535	62.603	62.696
35	63.266	63.120	62.983	62.863	62.762	62.686	62.633	62.605	62.601	62.620	62.664	62.732
40	63.451	63.304	63.167	63.045	62.942	62.859	62.796	62.753	62.729	62.725	62.744	62.789
45	63.624	63.484	63.354	63.238	63.138	63.054	62.985	62.930	62.890	62.868	62.867	62.893
50	63.783	63.658	63.543	63.442	63.353	63.276	63.209	63.151	63.103	63.069	63.056	63.072
55	63.931	63.831	63.741	63.664	63.596	63.536	63.481	63.430	63.385	63.351	63.336	63.352
60	64.076	64.009	63.955	63.912	63.876	63.843	63.812	63.781	63.751	63.730	63.728	63.757
65	64.229	64.206	64.196	64.197	64.203	64.210	64.215	64.216	64.217	64.223	64.247	64.302

Table 19: Current DSP LUT, $x = 40$ mm to 95 mm. Values are in kHz.

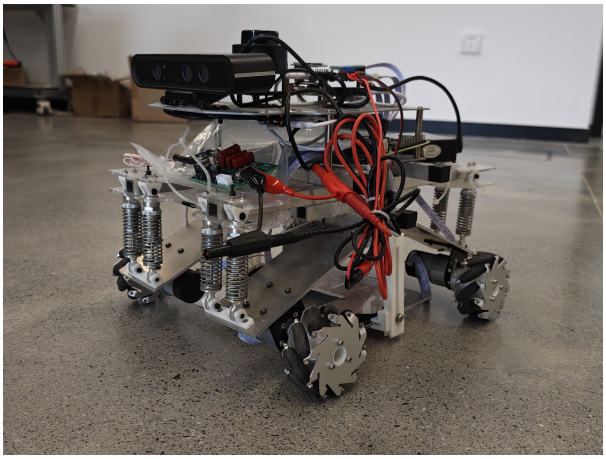
$y \backslash x$	40	45	50	55	60	65	70	75	80	85	90	95
−90	64.307	64.076	63.794	63.460	63.083	62.682	62.281	61.911	61.604	61.390	61.298	61.348
−85	64.227	64.082	63.876	63.607	63.284	62.925	62.557	62.213	61.927	61.733	61.659	61.729
−80	64.154	64.090	63.956	63.750	63.479	63.162	62.829	62.515	62.255	62.086	62.038	62.136
−75	64.078	64.086	64.018	63.869	63.647	63.373	63.076	62.795	62.568	62.431	62.417	62.548
−70	63.990	64.058	64.046	63.947	63.771	63.538	63.280	63.036	62.846	62.749	62.777	62.952
−65	63.881	63.997	64.029	63.973	63.837	63.643	63.426	63.223	63.078	63.028	63.105	63.332
−60	63.748	63.897	63.962	63.940	63.839	63.683	63.506	63.350	63.255	63.259	63.394	63.680
−55	63.590	63.757	63.844	63.847	63.776	63.657	63.523	63.415	63.376	63.441	63.640	63.991
−50	63.409	63.582	63.681	63.702	63.658	63.573	63.482	63.427	63.447	63.578	63.845	64.265
−45	63.211	63.381	63.484	63.518	63.497	63.446	63.400	63.399	63.481	63.678	64.015	64.505
−40	63.008	63.168	63.269	63.313	63.313	63.296	63.295	63.348	63.492	63.756	64.161	64.717
−35	62.812	62.958	63.057	63.109	63.131	63.146	63.190	63.296	63.499	63.825	64.292	64.907
−30	62.634	62.768	62.865	62.927	62.970	63.019	63.105	63.261	63.518	63.899	64.419	65.084
−25	62.489	62.614	62.712	62.786	62.852	62.933	63.059	63.260	63.563	63.989	64.550	65.249
−20	62.386	62.508	62.611	62.700	62.789	62.901	63.062	63.300	63.639	64.098	64.686	65.406
−15	62.332	62.455	62.568	62.674	62.787	62.927	63.118	63.385	63.749	64.227	64.827	65.551
−10	62.326	62.457	62.582	62.707	62.842	63.007	63.221	63.507	63.885	64.369	64.967	65.680
−5	62.364	62.505	62.646	62.788	62.944	63.128	63.359	63.656	64.036	64.513	65.095	65.785
0	62.435	62.588	62.743	62.902	63.074	63.273	63.513	63.812	64.186	64.647	65.203	65.859
5	62.526	62.688	62.854	63.027	63.211	63.419	63.663	63.959	64.319	64.758	65.282	65.898
10	62.618	62.786	62.961	63.142	63.335	63.548	63.792	64.080	64.425	64.838	65.329	65.904
15	62.699	62.868	63.045	63.230	63.428	63.643	63.886	64.166	64.495	64.885	65.345	65.883
20	62.759	62.922	63.096	63.283	63.482	63.700	63.942	64.217	64.535	64.907	65.343	65.851
25	62.794	62.946	63.115	63.300	63.501	63.722	63.967	64.242	64.557	64.920	65.340	65.829
30	62.811	62.950	63.111	63.293	63.498	63.725	63.979	64.263	64.583	64.948	65.365	65.845
35	62.827	62.951	63.104	63.286	63.497	63.737	64.007	64.308	64.645	65.022	65.449	65.933
40	62.865	62.976	63.123	63.309	63.532	63.791	64.085	64.413	64.776	65.178	65.625	66.125
45	62.955	63.057	63.204	63.399	63.640	63.925	64.251	64.614	65.014	65.450	65.928	66.452
50	63.126	63.228	63.383	63.593	63.860	64.178	64.542	64.948	65.391	65.870	66.385	66.942
55	63.410	63.520	63.691	63.926	64.224	64.581	64.991	65.444	65.936	66.461	67.019	67.612
60	63.830	63.961	64.157	64.424	64.760	65.161	65.619	66.123	66.665	67.239	67.842	68.473
65	64.404	64.566	64.798	65.104	65.484	65.933	66.440	66.996	67.589	68.211	68.857	69.525

Appendix C Prototype Photographs

Figures 3 and 4 show the implemented hardware used for system integration and verification.

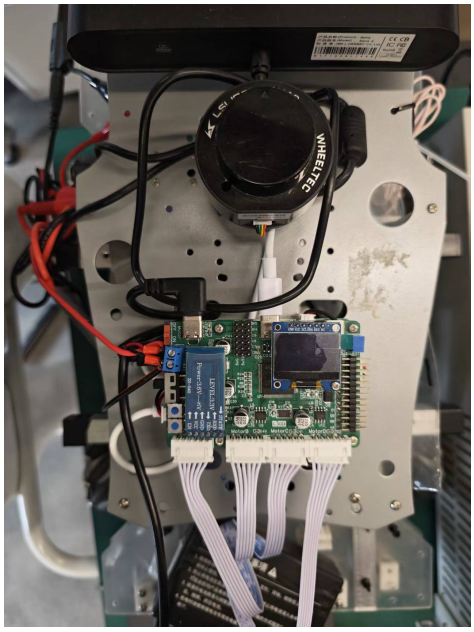


(a) Charging subsystem.

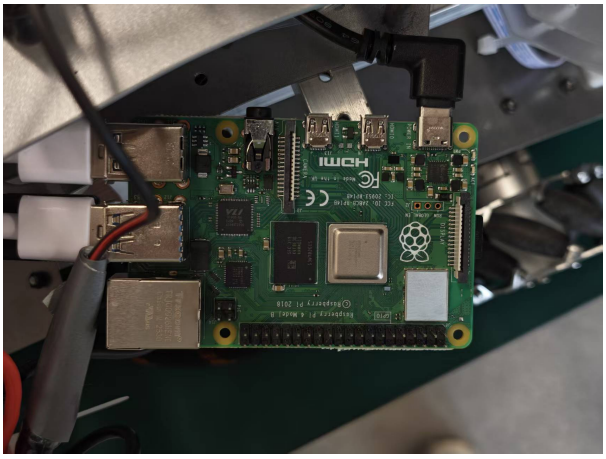


(b) Full AGV prototype.

Figure 3: Prototype photographs of the charging subsystem and full AGV.



(a) STM32 motion-control board.



(b) Raspberry Pi controller.

Figure 4: Prototype photographs of the vehicle-side control hardware.

References

[1] W. C. Brown, "The history of power transmission by radio waves," *IEEE Transactions on Microwave Theory and Techniques*, vol. 32, no. 9, pp. 1230–1242, 1984.

[2] X. Wu, H. Xu, J. Xiao, Y. Mo, N. Wu, and S. Chen, "Overview of wireless power supply technology for electric vehicles," in *2023 IEEE 6th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, 2023, pp. 66–69.

[3] M. Liu, X. Wang, and J. Xu, "Design methodology of SiC MOSFET based bidirectional CLLC resonant converter for wide battery voltage range," in *2021 IEEE Workshop on Wide Bandgap Power Devices and Applications in Asia (WiPDA Asia)*, 2021, pp. 423–427.

[4] X. Li and A. K. S. Bhat, "Analysis and design of high-frequency isolated dual-bridge series resonant DC/DC converter," *IEEE Transactions on Power Electronics*, vol. 25, no. 4, pp. 850–862, 2010.

[5] Raspberry Pi Foundation, *Raspberry pi 4 model B documentation*, 2024. [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[6] OpenCV. "Detection of ArUco markers. "[Online]. Available: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

[7] STMicroelectronics, *STM32 32-bit ARM Cortex MCUs reference manual*, 2024.

[8] IEEE. "IEEE Code of Ethics," Accessed: Feb. 8, 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[9] International Commission on Non-Ionizing Radiation Protection, "Guidelines for limiting exposure to electromagnetic fields (100 kHz to 300 GHz)," *Health Physics*, vol. 118, no. 5, pp. 483–524, 2020.