ECE 3331 - 301

Final Report for Project Lab 1 “Soccer Robot”

Atharva Undire, R#11900139

Sam Graeber, R11902699

Sergio Dominguez, R#12050148

Texas Tech University

Electrical and Computer Engineering Department

May 2026

Abstract

The Control Alt Defeat team developed an autonomous soccer robot for the Raider Red Soccer competition. The final system differs from the earlier design by moving from a large belt-driven conveyor concept to a compact under-chassis intake and sorting path built around the Dagu 5 rover chassis. The updated mechanism is arranged as a funnel, a ball motor fitted with multiple rubber-band strands, a gutter, a servo gate with a color sensor mounted near the decision point, and a collector. The funnel guides the ping-pong ball into the ball motor, the rotating rubber-band strands pull the ball through the intake, the gutter holds or stages the ball, and the servo gate either accepts the ball into the collector or rejects/reverses it toward the appropriate goal path.

The perception system now uses an ArduCAM OV2640 camera module configured for RGB565 160 x 120 image capture. The camera performs background calibration, temporal averaging, foreground color thresholding, blob grouping, centroid calculation, and surface classification to locate red and green balls while avoiding false positives. A TCS34725 color sensor gives a second, close-range color decision after a ball reaches the gutter. Two VL53L1X time-of-flight sensors support obstacle detection, three downward-facing IR sensors detect boundary tape, and a BNO08x IMU provides yaw feedback for heading correction. A Raspberry Pi Pico 2 runs the C/C++ control code, polling sensors, updating pose estimates, selecting targets, controlling two drive motors, running the ball motor, and moving the servo gate using PWM.

The code implements a sense-plan-act loop with mode selection, goal color selection, camera calibration, search behavior, target centering, stepwise approach, funnel commit, color-sensor confirmation, collector management, boundary avoidance, obstacle avoidance, and

timed final scoring. Red balls are accepted into the collector until capacity is reached, while non-red balls are rejected or shot toward the ‘defend goal’. The collector can hold up to four balls, and the final scoring window triggers release behavior. This report preserves the structure of the earlier submission while updating the mechanical, electrical, vision, software, and testing sections to reflect the current ArduCAM/color-sensor/servo-gate/ball-motor architecture.

Acknowledgments

The Control Alt Defeat team thanks Dr. Ben Esser for guidance, project requirements, and access to lab resources. The team also acknowledges the contributions of each team member during mechanical prototyping, electrical integration, firmware development, sensor debugging, and field testing. References include manufacturer datasheets, course resources, project code, weekly progress presentations, and testing notes.

Table of Contents

List of Figures

List of Tables

1. Introduction

The Raider Red Soccer competition requires an autonomous robot to search for colored ping-pong balls, stay inside the playing field, avoid obstacles, and manipulate balls according to competition scoring rules. The current robot uses the Dagu 5 rover chassis as the base platform and places the entire intake and sorting mechanism underneath the rover. This preserves a low and compact profile while allowing balls to be guided through a controlled path from intake to decision point.

1.1 Competition Context and Objectives

The robot must operate without driver input after the start command. Its success depends on four connected capabilities: detecting valid target balls, approaching them without crossing field boundaries, pulling them into a physical intake, and then accepting or rejecting each ball based on color. The current design focuses on a reliable single-ball path: funnel -> ball motor -> gutter -> servo gate and color sensor -> collector or rejection path.

1.2 Updated Design Rationale and Subsystem Overview

The updated robot is organized around the following subsystems:

Vision System: An ArduCAM OV2640 camera is used for forward-facing ball detection. The camera is initialized through I2C/SPI and configured for RGB565 160 x 120 frames. The software calibrates the background, reads frames from the FIFO, filters motion/color, finds connected blobs, and selects target balls based on color, surface, and previous rejection history.

Close-Range Color Confirmation: A TCS34725 color sensor is mounted at the servo gate/gutter decision point. The camera identifies and approaches balls at range, but the color

sensor confirms the ball after it physically reaches the intake path. This prevents the robot from committing to a scoring action based only on long-range vision.

Ball Intake and Sorting: The underbody mechanism uses a funnel to guide the ball into a rotating ball motor. The ball motor has multiple rubber-band strands that act like compliant paddles, pulling the ball into the gutter. The servo gate controls whether the ball transfers into the collector or stays/reverses for rejection/scoring.

Mobility and Chassis: The robot uses the Dagu 5 rover chassis as the base. Two drive motors are controlled using PWM through motor drivers, while IMU feedback is used to maintain heading and improve repeatability during straight drives and turns.

Obstacle and Boundary Sensing: Two VL53L1X ToF sensors provide obstacle detection, and three downward-facing IR sensors detect field boundaries. The code immediately prioritizes boundary and obstacle handling before continuing with target pursuit.

Controller and Firmware: The Raspberry Pi Pico 2 runs the control loop in C/C++. The firmware initializes all sensors, calibrates the camera, polls sensors, updates the game state, chooses targets, controls the drive motors, actuates the ball motor, and moves the servo gate.

1.3 Reasoning for the Updated Approach

- The OV2640 camera provides a more manageable camera interface through the ArduCAM module and supports compact RGB565 frames suitable for Pico-side processing.
- The color sensor reduces scoring risk because the final accept/reject decision is made after the ball reaches the mechanism, not only from a camera frame.
- The funnel and rubber-band ball motor are mechanically simpler than a full conveyor and fit under the Dagu 5 chassis.

- The servo gate gives a clear binary mechanical decision: accept into collector or reject/hold for shooting.
- The ball motor can reverse for shooting/ejection, so the same actuator assists both collection and scoring.
- C/C++ is still preferred over MicroPython because the loop must poll sensors, process camera frames, update PWM outputs, and react to boundaries quickly.

1.4 High-Level Architecture

The robot follows a sense-plan-act architecture. Sensors collect data from the OV2640 camera, TCS34725 color sensor, VL53L1X LIDAR sensors, BNO08x IMU, IR boundary sensors, and mode/goal/start switches. The Pico 2 processes this information into a GameState structure containing mode, goal color, start side, pose, collected-ball count, current gutter ball, rejected target memory, and timing. The actuation layer drives left and right wheel motors, the rubber-band ball motor, and the servo gate.

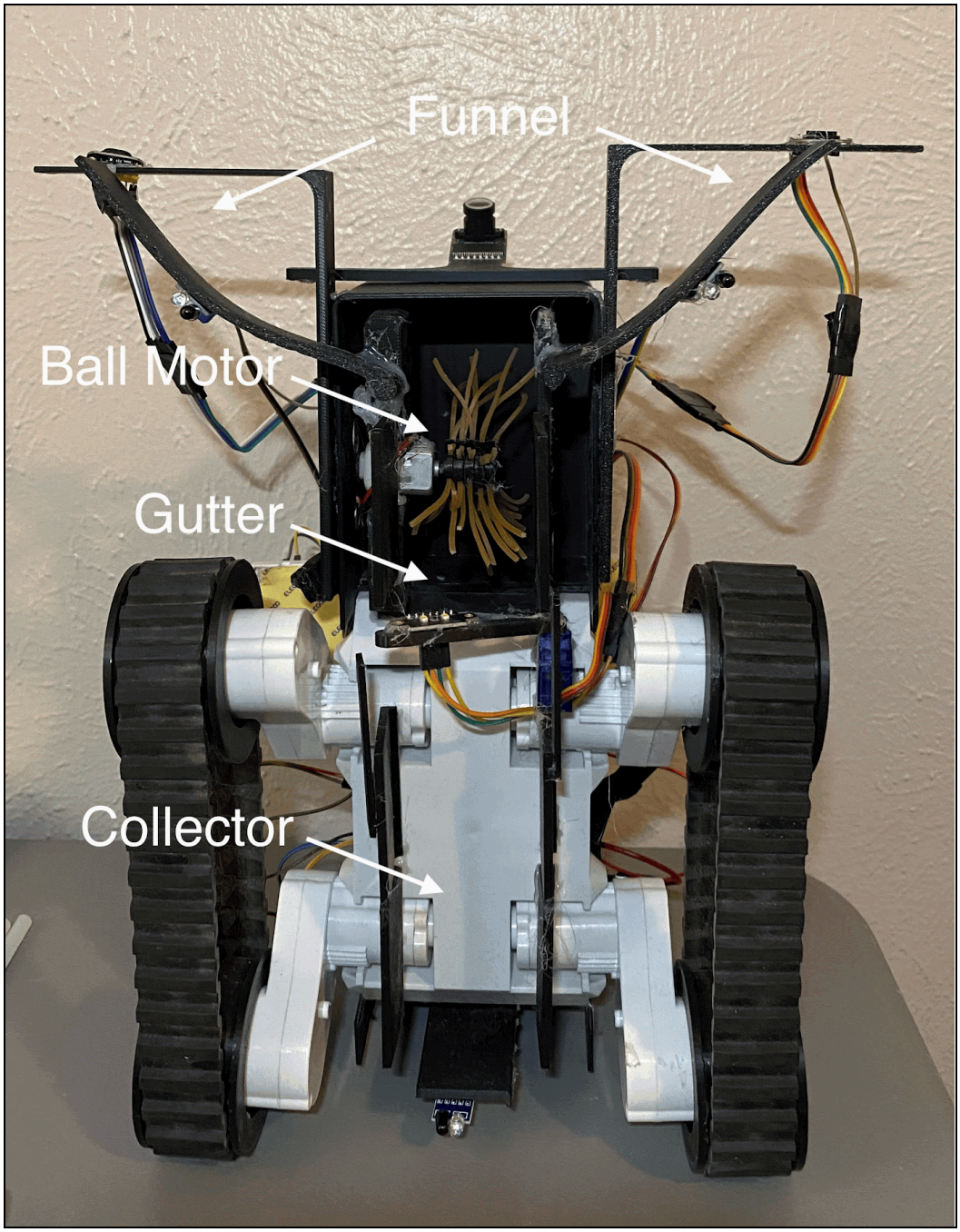

Figure 1. Dagu 5 rover chassis with underbody intake path

2. Body

The updated system was developed through iterative mechanical prototyping, sensor bring-up, code integration, and field testing. The main redesign replaced the previous belt conveyor with a compact underbody intake path that is easier to package beneath the chassis and easier to control in software.

2.1 Mechanical Design, Intake Path, and DFM

([Name Placeholder])

The current mechanical design is based on the Dagu 5 rover chassis with a custom mechanism mounted underneath. The main challenge is that the mechanism must fit inside the available underbody volume while still allowing a 40 mm ping-pong ball to enter, contact the moving rubber-band strands, and move smoothly into the gutter without jamming.

2.1.1 Additive Manufacturing Systems and Materials

FDM printing remains suitable for larger brackets, mounts, funnel walls, and chassis adapters because those parts benefit from fast iteration and reasonable strength. Resin printing remains useful for small, precise parts such as gate brackets, sensor mounts, and geometry test pieces where dimensional accuracy matters. The final mechanism should use smooth surfaces anywhere the ball touches the funnel or gutter.

Material	**Use in Updated Design**	**Reason**
PLA/FDM	Funnel walls, chassis adapters, ball-motor brackets	Fast iteration and adequate stiffness
Photopolymer resin/mSLA	Servo mount, color sensor bracket, small test pieces	Higher detail and smoother finish
Rubber bands	Ball motor strands/paddles	Compliant contact with ping-pong balls
Fasteners/spacers	Gate and sensor positioning	Adjustable alignment during testing

Table 1. Material properties and updated mechanical use

2.1.2 Dagu 5 Chassis Packaging and Clearance

Unlike the earlier custom-wheel/conveyor plan, the updated robot uses the Dagu 5 rover chassis as the primary drive platform. The ball handling system is therefore constrained by the chassis underside, wheel spacing, and available mounting points. The intake path must be low enough to contact the ball but high enough to avoid dragging on the field. The funnel opening should be wider than the ball diameter and taper toward the ball motor so that small alignment errors during approach still result in capture.

Important clearance and packaging constraints include:

- Funnel inlet asymmetric to prevent ball blockages during multi-ball approaches.
- Ball motor centered after the funnel to prevent the ball from hitting one side of the gutter.
- Color sensor placed on the servo arm so the sensed color corresponds to the staged ball.
- Servo gate travel kept clear of the ball motor and collector entrance.
- Sensor wiring routed above or beside the gutter so that wires cannot touch the ball path.

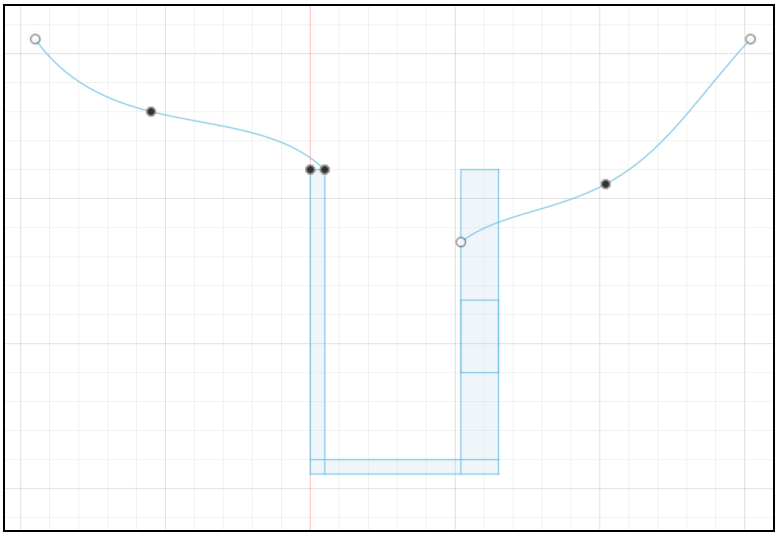

Figure 2. Funnel to ball-motor intake geometry

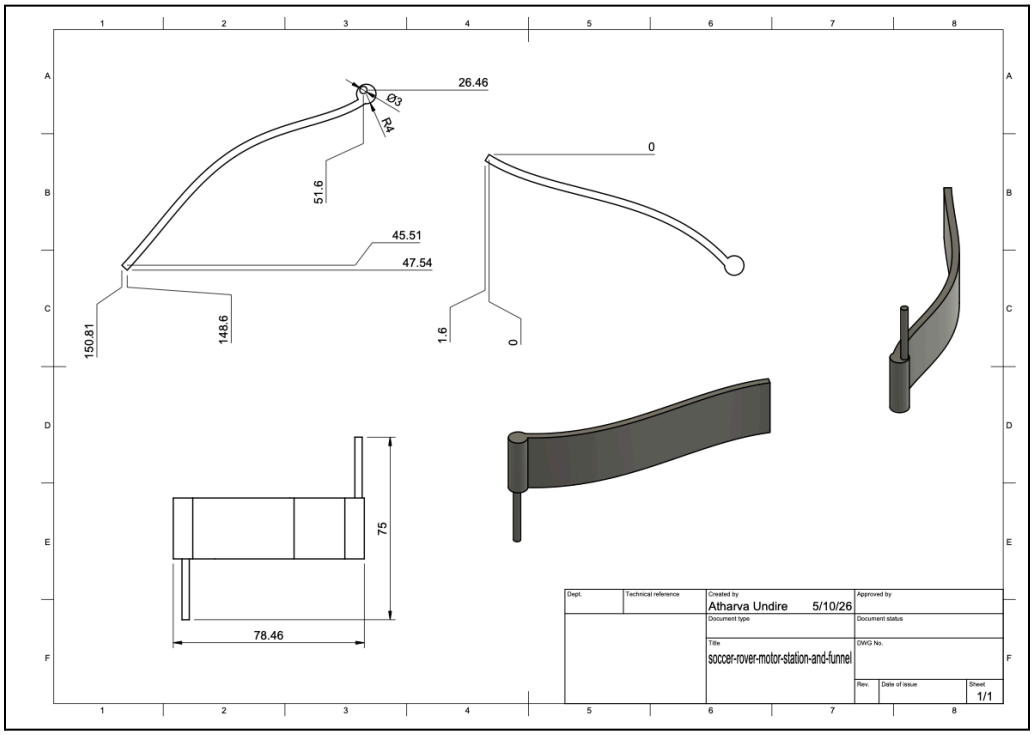

Figure 3. Asymmetric Funnel Drawing

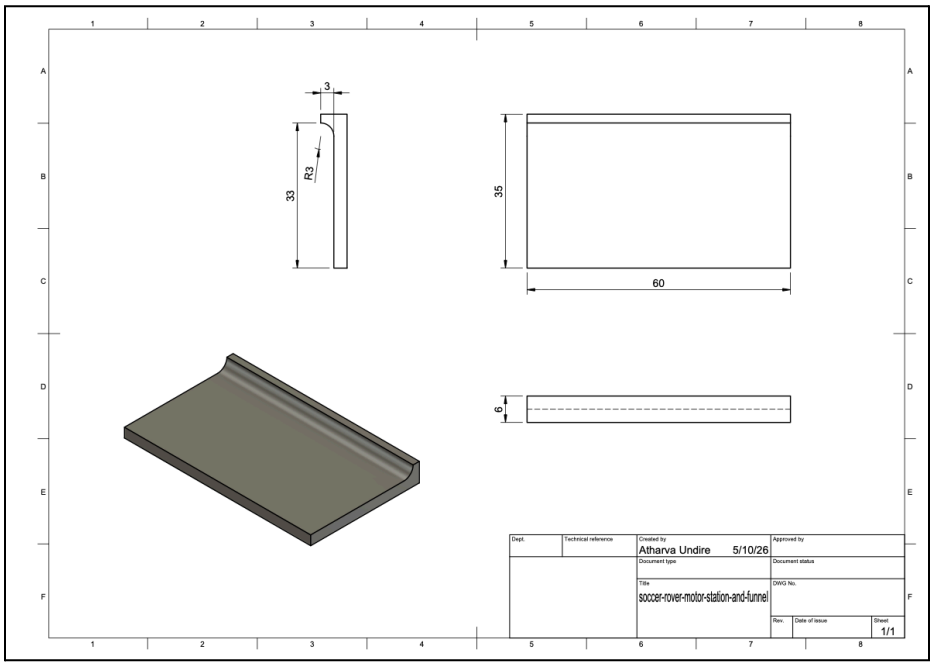

Figure 4. Gutter/Collector Wall Segment

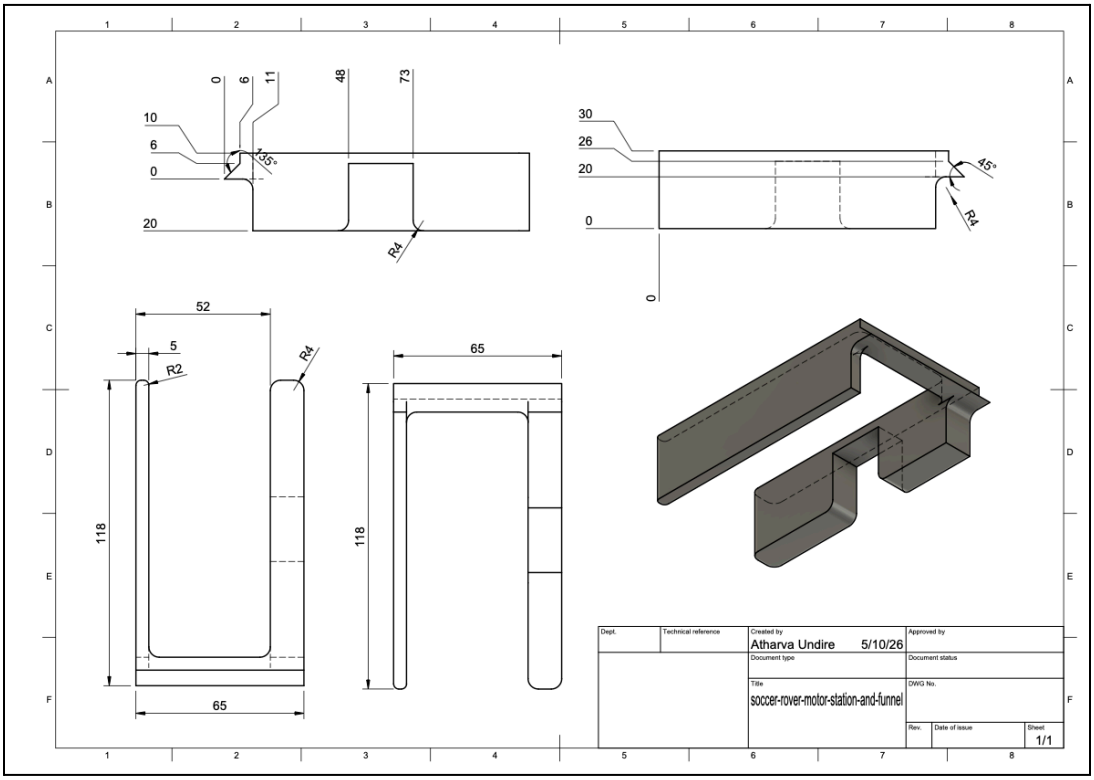

Figure 5. Ball Motor Bracket

2.1.3 Ball Motor, Rubber-Band Strands, Gutter, and Servo Gate

The ball motor replaces the previous large conveyor. Mechanically, the motor shaft carries multiple rubber-band strands. These strands behave like flexible fingers. When the ball reaches the motor, the strands deflect around the ball, generate frictional contact, and pull the ball into the gutter. This design is forgiving because the contact surface is compliant instead of rigid. It also reduces jamming risk compared with hard paddles, since the strands can bend if the ball is slightly misaligned.

After the ball motor, the gutter acts as a staging lane. The ball pauses or rolls to the servo-gate/color-sensor position. The color sensor provides the final ball color reading. The servo arm then changes the route. In the normal red-ball path, the gate opens to let the ball transfer into the collector. For non-red balls, the code keeps or lowers the gate and uses the ball motor reverse/scoring routine to reject or shoot the ball away from the collector.

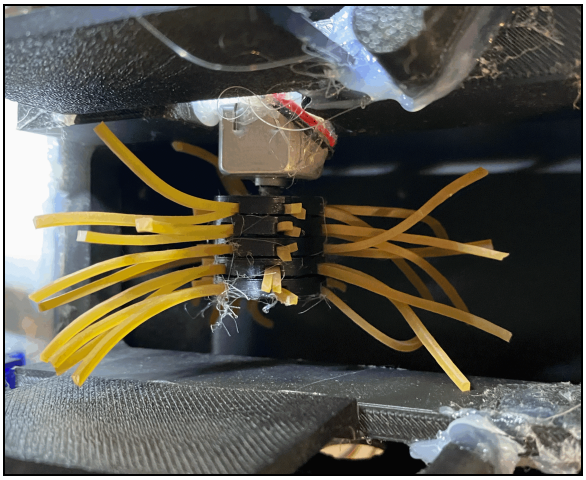

Figure 6. Ball Motor Rubber-band Strands

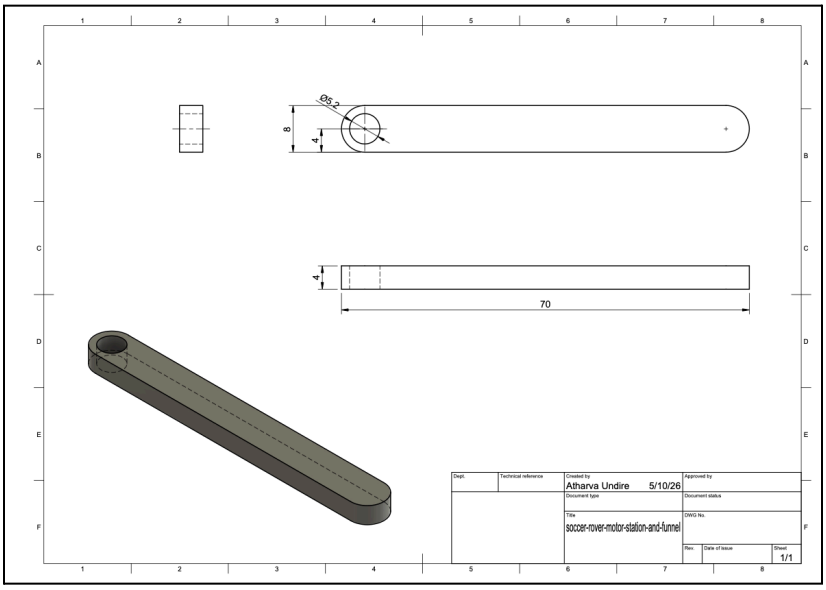

Figure 7. Servo gate + color sensor bracket

2.1.4 DFM Optimizations and Iteration

The updated mechanism places more emphasis on alignment, smoothness, and serviceability than on large custom printed structures. The funnel should be printed with enough wall thickness to resist bending, the gutter should be smooth and slightly oversized for the ball,

and the servo/color sensor mount should allow adjustment. DFM changes also include separating the funnel, motor mount, gutter, and sensor gate into smaller replaceable pieces so each part can be adjusted without reprinting the full assembly.

- Use fillets at funnel transitions to remove sharp edges that can stop the ball.
- Add elongated mounting holes for the color sensor and servo bracket to tune the detection point.
- Leave access to the ball motor shaft so rubber bands can be replaced quickly.
- Use modular screw-on pieces so the mechanism can be removed from the Dagu chassis during debugging.
- Keep the collector entrance large enough for reliable transfer when the gate opens.

2.1.5 Failure Modes and Mitigations

Failure Mode	**Likely Cause**	**Mitigation**
Ball jams in funnel	Inlet too narrow, rough edges or funnel too symmetric	Widen inlet, add fillets, sand or smooth ball-contact surfaces
Ball motor slips	Rubber strands too loose or too few contact points	Increase strand count/tension and test motor duty
Color sensor reads wrong ball/color	Sensor too far from ball or ambient light interference	Shield the sensor and move it closer to the gutter decision point
Servo gate blocks accepted balls	Gate angle or pulse width not tuned	Calibrate up/down pulses and check gate clearance
Collector overfills	Too many accepted balls without scoring release	Use software capacity limit and final scoring behavior
Mechanism drags on field	Underbody mounting too low	Add standoffs or revise bracket height

Table 2. Updated failure modes, causes, and mitigations

2.2 Motor Control, Power Electronics, and PCB

2.2.0 Overall Electrical Architecture

The robot electrical system is centered around the Raspberry Pi Pico 2 microcontroller. The Pico controls the drive motors, ball motor, servo gate, camera system, and sensors while also handling autonomous navigation logic. Multiple communication interfaces are used to separate sensing and control tasks. The LIDAR sensors and IMU are connected through one I2C bus, while the color sensor and ArduCAM sensor-control interface use a second I2C bus. The OV2640 camera additionally uses SPI communication for image/FIFO data transfer through the ArduCAM module.

PWM outputs from the Pico are used for the left and right drive motors, the ball motor, and the servo gate. Additional GPIO pins are used for IR boundary sensors, start switches, XSHUT control for the dual LIDAR setup, and mode-selection switches. Separating subsystems across multiple interfaces simplified debugging and reduced communication conflicts during development.

Subsystem	**Interface / Pins in Code**	**Purpose**
Right drive motor	GP0, GP1, GP2 sleep	Right-side rover motion
Left drive motor	GP3, GP4, GP5 sleep	Left-side rover motion
Ball motor	GP6, GP7	Collect/shoot using rubber-band strands
Start-side switch	GP8	Select starting side
Servo gate	GP9 PWM	Accept/reject gate control
ArduCAM SPI	GP10 SCK, GP11 TX, GP12 RX, GP13 CS	OV2640 frame/FIFO transfer
Mode/goal switches	GP14, GP15	Kicker/goalie mode and defend goal color
LIDAR/IMU I2C0	GP16 SDA, GP17 SCL	VL53L1X and BNO08x sensing
LIDAR XSHUT	GP18, GP19	Dual LIDAR address setup

Start button	GP20	Begin autonomous run
IR boundary sensors	GP21, GP22, GP28	Front-left, back, front-right boundary detection
Camera/color I2C1	GP26 SDA, GP27 SCL	OV2640 control and TCS34725 color sensor

Table 3. Pin assignments from current firmware

2.2.1 Custom DRV8876 Motor Driver PCB

A custom motor-driver PCB was designed around the Texas Instruments DRV8876 H-bridge motor-driver IC. The DRV8876 was selected because it supports bidirectional DC motor control, PWM operation, integrated current limiting, undervoltage protection, and thermal shutdown while remaining compatible with the robot battery voltage range.

Each PCB includes local decoupling capacitors, charge-pump capacitors, and motor power filtering placed close to the driver IC. These components help stabilize the motor supply voltage during rapid acceleration, braking, and direction changes. The board also includes dedicated sleep-control connections so the Pico can enable or disable the motor drivers during startup and debugging.

Several PCB revisions and soldering iterations were required during development because the DRV8876 uses a small surface-mount package with fine-pitch pins and a thermal pad underneath the IC. Early prototypes experienced unstable behavior caused by poor solder joints and electrical noise during motor startup. Additional bypass capacitors and improved grounding helped improve stability in later revisions.

The final custom PCB significantly reduced loose wiring problems compared with the original breadboard setup and made the electrical system more modular and easier to debug.

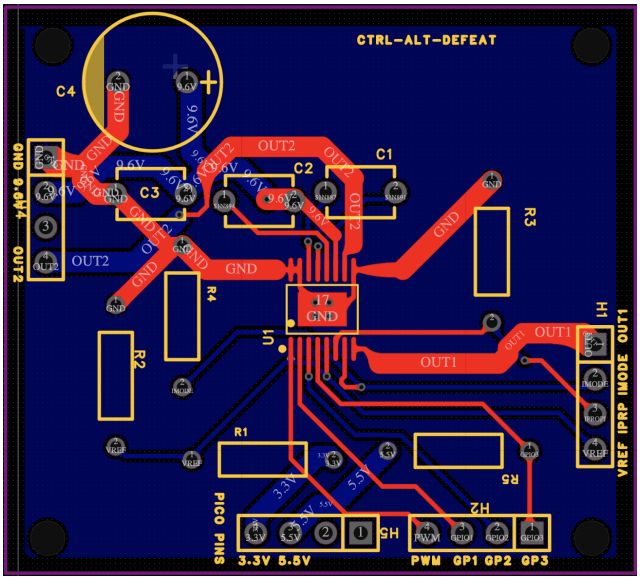

Figure 5.1. DRV8876 PCB layout

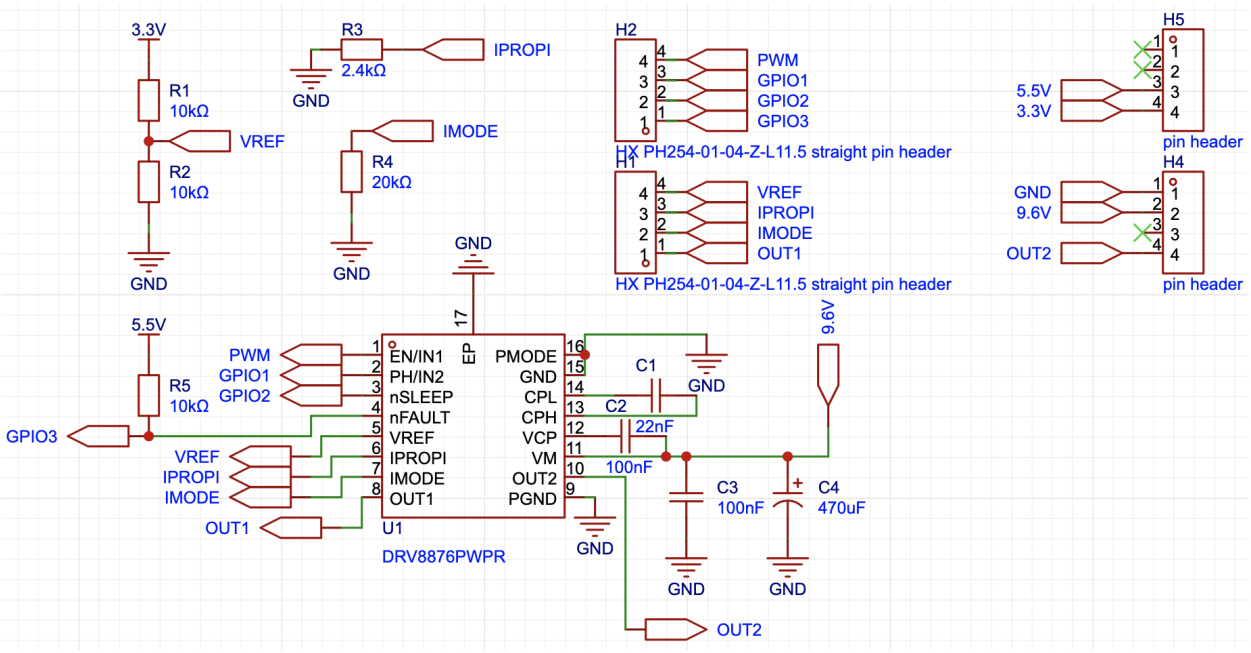

Figure 5.2. DRV8876 schematic and component placement

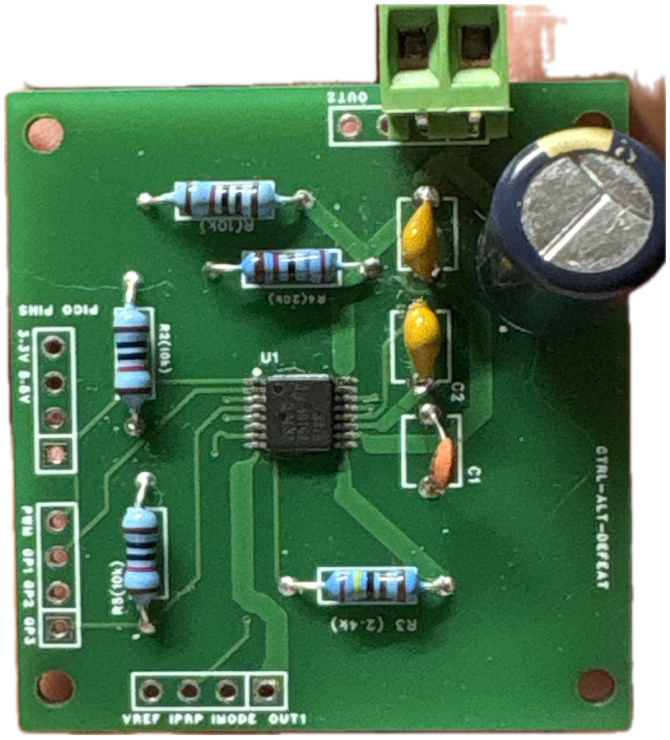

Figure 5.3. Assembled DRV8876 motor-driver PCB

2.2.2 Motor Driver and PWM Design

The drive motors use bidirectional PWM control using the DRV8876 motor drivers. Each wheel side has two input pins and a sleep/control pin connected to the Raspberry Pi Pico 2. PWM control was selected because it allows efficient motor-speed control while reducing unnecessary heat and power loss compared with linear motor-control methods.

The firmware configures the PWM system with calibrated duty-cycle values for the left and right motors to compensate for small mechanical differences between the drive sides. Independent left and right motor control allows the robot to move forward, reverse, rotate, and perform steering corrections during autonomous operation.

The ball motor uses a similar bidirectional PWM-control approach. During collection mode, the motor spins forward so the rubber-band strands pull ping-pong balls into the gutter system. During shooting or rejection mode, the motor reverses direction at high duty cycle while the robot backs away from the goal area.

2.2.3 Servo Gate PWM

The servo gate uses a 20 ms PWM period generated by the Raspberry Pi Pico 2. The firmware defines an up pulse of 1000 microseconds and a down pulse of 1583 microseconds. The servo_gate_set function selects the correct pulse based on whether the requested angle is closer to the up or down position.

The software uses 0 degrees for the gate-up position and 35 degrees for the gate-down position. During operation, the servo gate directs accepted balls into the collector while rejected balls remain in the shooting or rejection path.

Careful testing of the servo system was required to ensure that the gate mechanism did not interfere with the gutter, intake system, color sensor, or moving ball path during autonomous operation.

2.2.4 Multi-Output Buck Converter and Power Distribution

The robot electrical system uses a multi-output buck-converter design to supply stable operating voltages to the different subsystems. The drive motors operate directly from the main

9.6 V rechargeable battery supply, while the Raspberry Pi Pico 2, sensors, servo, and logic electronics require lower regulated voltages for stable operation.

The regulated outputs included logic power for the Raspberry Pi Pico 2 and sensors, as well as separate regulated rails for servo and motor subsystems. Separate regulated rails were used for the logic electronics and high-current motor subsystems. This separation helped reduce voltage sag, electrical noise, and Pico resets caused by rapid motor current spikes during acceleration and reversal. All subsystems share a common ground reference to maintain stable PWM signals and sensor communication between devices.

The power-distribution system was designed so that high-current motor wiring remained separated from sensitive logic and sensor wiring where possible. Additional decoupling capacitors were added near the motor drivers and power rails to reduce switching noise and improve voltage stability during autonomous operation.

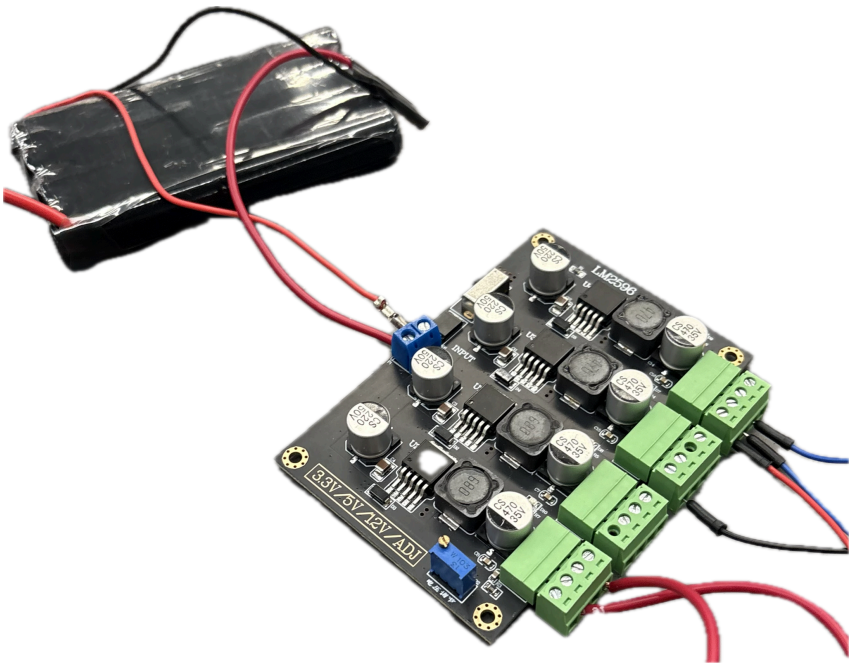

Figure 5.4. Multi-output buck converter subsystem

2.2.5 Electrical Design Constraints and Considerations

Important electrical-system constraints and design considerations included:

- Shared ground reference between logic and motor subsystems to maintain stable PWM and sensor communication.
- Separate regulated voltage rails for Pico logic, sensors, servo, and drive motors to reduce brownout resets.
- Decoupling capacitors placed near motor drivers to reduce switching noise during acceleration and motor reversal.
- Motor-driver sleep pins connected to the Pico for software enable/disable control during startup and debugging.
- High-current motor traces and power wiring kept short where possible to reduce voltage drop and electrical noise.
- Sensor, SPI, and I2C wiring routed away from motor power wiring to reduce interference with camera and sensor communication.
- Servo and sensor wiring routed above or beside the gutter system so wires could not interfere with the ball path or moving mechanisms.

2.2.6 Electrical Stability, Debugging, and Lessons Learned

Several electrical and integration challenges were encountered during development of the robot. Early breadboard prototypes experienced unstable motor-driver behavior caused by electrical noise, insufficient grounding, and unreliable solder connections. Because the motors

generate large transient currents during startup and reversal, additional bypass capacitors and improved grounding layouts were added in later revisions to improve stability.

The custom PCB and regulated power system significantly improved reliability compared with the original prototype wiring. Separating motor power from logic power reduced unintended Pico resets and improved sensor consistency during autonomous operation. Additional debugging effort was required to verify stable PWM behavior, proper motor-driver operation, and reliable communication between the Pico, sensors, and camera system.

The final electrical system provided a more modular and serviceable platform for integrating the drive system, sensing hardware, camera subsystem, and ball-handling mechanisms into a single autonomous robot.

2.3 ArduCAM OV2640 Vision System and Ball Detection

2.3.1 ArduCAM OV2640 Hardware Setup and Circuit

The updated vision system uses an ArduCAM module with an OV2640 sensor. The code verifies the ArduChip SPI test register, probes the sensor over I2C, and initializes the camera for RGB565 160 x 120 output. The camera is connected through SPI1 for image/FIFO reads and I2C1 for sensor register configuration. This is more practical than directly reading a parallel camera bus on the Pico because the ArduCAM module handles image capture buffering.

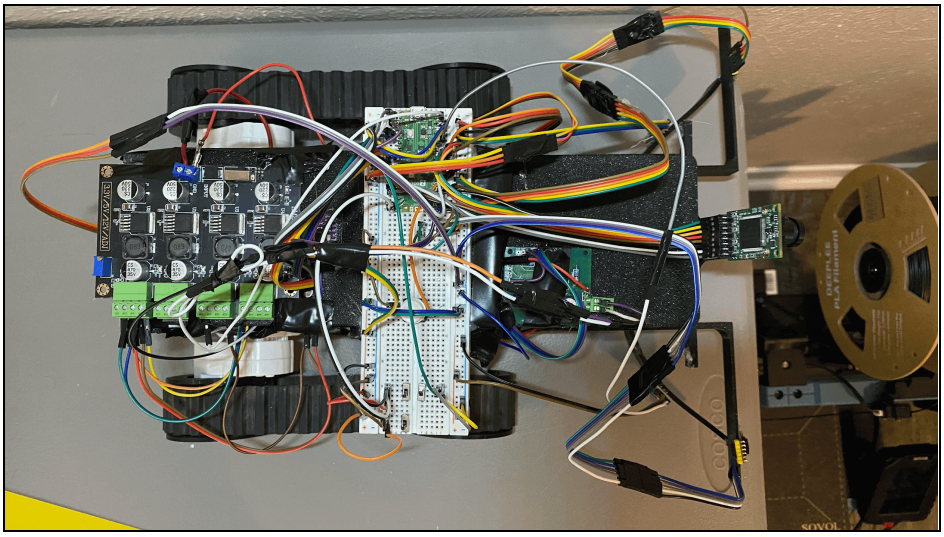

Figure 6. Wired Breadboard Image

2.3.2 Background Calibration and Temporal Averaging

Before the game loop relies on camera detections, the firmware captures calibration frames to build per-pixel background estimates. The blob code stores background red, green, and blue values and a local variation band. During detection, the current frame is compared to this background, so only pixels with sufficient change and saturation are considered foreground ball candidates. Temporal averaging smooths frame noise by keeping a small rolling history of RGB values.

2.3.3 RGB565 Color Thresholding and Blob Grouping

Each RGB565 pixel is expanded into 8-bit red, green, and blue values. The code computes normalized color ratios and saturation. Low-saturation or low-brightness pixels are rejected. Foreground pixels are classified as red, green, or blue based on threshold rules. The code then groups adjacent pixels into connected components, computes area, centroid, bounding box, and average color, and stores the largest useful ball candidates.

2.3.4 Surface Classification and Centerline Avoidance

The camera logic also samples surface colors such as black, white/gray, yellow, and purple. This helps identify whether a detected ball is on a legal or risky surface and supports goal/centerline logic. In the game loop, blue centerline detections are treated as a safety condition, especially for goalie behavior, so the robot can escape or avoid illegal field regions.

2.3.5 Target Selection and Distance Estimate

The target-selection code rejects balls if a ball is already in the gutter, if the collector is full, if the color is unknown, if blue balls are disabled, or if the ball matches a recently rejected target at nearly the same image position. The selected ball is centered by comparing its centroid to the image center. If the centroid is too far left or right, the robot performs small rotations. Once centered, distance is estimated using a constant divided by apparent ball size, clamped between minimum and maximum approach distances.

2.4 Sensor Integration: TCS34725, LIDAR, IMU, and IR

2.4.1 TCS34725 Color Sensor at the Gate

The TCS34725 color sensor is initialized on I2C1 at address 0x29. The code checks the sensor ID, enables power and ADC, sets integration time and gain, and then reads clear, red, green, and blue channels when new data is available. The classification function normalizes RGB values and requires enough raw intensity before returning red, green, blue, or unknown. In the game loop, this color reading is the physical confirmation that a ball has reached the gutter/gate region.

The software decision is simple: a red sensed ball is accepted into the collector if the collector is not full; otherwise the robot begins attack-goal shooting behavior. A non-red sensed ball is sent to the defend-goal routine. This makes the sensor/gate assembly the final sorting authority.

2.4.2 ToF LIDAR Array for Obstacle Avoidance

Two VL53L1X sensors are initialized with XSHUT control so they can be assigned different addresses. The code polls both sensors, applies calibration corrections, and considers an obstacle present when a measured distance is below the stop threshold. If an obstacle is detected, the robot stops or executes bypass logic before continuing.

2.4.3 BNO08x IMU for Orientation and Drift Correction

The BNO08x IMU provides yaw, pitch, roll, accelerometer data, and calibrated gyroscope data. The game state stores a yaw zero at start and updates the robot heading from the relative yaw. During straight drives, heading error is converted to left/right duty corrections so

the robot does not drift significantly while approaching the ball or moving toward scoring positions.

2.4.4 Downward-Facing IR Boundary Array

Three IR boundary sensors are used in the current code: front-left, front-right, and back. The logic builds an IR boundary state from these inputs. Forward motion is blocked by front sensors and reverse motion is blocked by the back sensor. When a boundary event occurs, the robot prioritizes escape behavior before resuming ball pursuit.

2.5 Control Software and C/C++ Implementation on Raspberry Pi Pico 2

2.5.1 Firmware Initialization

On boot, the firmware initializes USB serial output, I2C0, I2C1, SPI1, the LIDAR sensors, IMU, TCS34725, ArduCAM OV2640, wheel motors, ball motor, mode switches, start button, and servo gate. The servo starts in the down position and all motion is stopped before the autonomous loop begins. The robot then waits for the game start input.

2.5.2 Game State and Input Modes

The GameState structure stores the selected rover mode, defend goal color, attack goal color, start side, estimated pose, gutter ball color, collected-ball array, collected count, IMU yaw zero, search progress, rejected target memory, and several flags for obstacle/boundary events. Mode and goal switches allow the same codebase to run goalie or kicker behavior and to assign the appropriate defend/attack goal colors.

2.5.3 Main Autonomous Loop

After the start button, the robot lowers the gate, starts the ball motor in collect mode, zeros pose/yaw, and enters the timed game loop. Every loop cycle polls LIDAR and IMU, periodically reads IR, color sensor, and camera data, prints status, checks game time, and then chooses the highest-priority action. The priority order is safety first, then final scoring/collector-full behavior, then color-sensor handling, then boundary/obstacle handling, then camera target pursuit, and finally search behavior.

2.5.4 Camera Target Pursuit and Funnel Commit

When a valid camera blob is found, the robot rotates until the target is near the camera center. It then estimates distance from apparent ball size and drives in capped steps. After the ball

leaves the camera view, the code intentionally drives a remaining distance into the funnel. If a boundary is hit during this commit, the target is rejected. If the color sensor detects a ball, the ball is handled using the color-sensed routine. If no color sensor hit occurs, the code may fall back to the last camera target color.

2.5.5 Servo Gate, Collector, and Shooting Logic

The ball motor has three modes: off, collect, and shoot. Collect mode spins the motor forward to pull balls into the gutter. Shoot mode reverses the motor for ejection. The servo gate has two meaningful states: up and down. For a red ball, the code opens the gate and drives a transfer distance so the ball moves into the collector; then it lowers the gate again. The collector is tracked in software with a maximum of four balls. For non-red balls, the robot runs a defend-goal shot routine instead of adding the ball to the collector. During the final scoring window or when the collector is full, the robot releases stored balls toward the attack goal.

2.5.6 MicroPython vs C/C++ Tradeoff Evaluation

The earlier tradeoff still applies, but the current code makes the need for C/C++ more obvious. The firmware must read from an SPI camera FIFO, classify image pixels, maintain camera calibration buffers, poll multiple sensors, run PWM motor outputs, and respond to boundary events with low latency. C/C++ provides direct control over memory and timing, which is important on the Pico 2.

2.6 Rover Mobility and Testing

2.6.1 Dagu 5 Rover Assembly and Drive Tests

Mobility testing focuses on whether the Dagu 5 rover can move predictably while carrying the underbody intake mechanism. Drive calibration in the code maps motor duty and timing to field units, with separate left and right duty limits to compensate for asymmetry. IMU correction is used during longer moves to reduce drift.

2.6.2 Intake, Gate, and Color Sensor Tests

The updated mechanism requires staged testing. First, the ball motor should be tested alone to confirm that rubber-band strands pull the ball from the funnel into the gutter. Second, the color sensor should be tested with stationary balls at the decision point to tune thresholds and shielding. Third, the servo should be tested with red and non-red balls to verify that the gate opens only for accepted balls. Finally, the full funnel -> ball motor -> gutter -> servo/color sensor -> collector path should be tested during autonomous approach.

2.6.3 Integrated Autonomous Tests

Integrated tests should verify that camera detections produce useful approach motions, that the robot commits into the funnel after losing the ball from view, that the color sensor detects the ball after intake, that red balls are collected, and that non-red balls are rejected or shot toward the defend goal. Boundary and obstacle tests should be run between intake tests because the main loop prioritizes those safety behaviors.

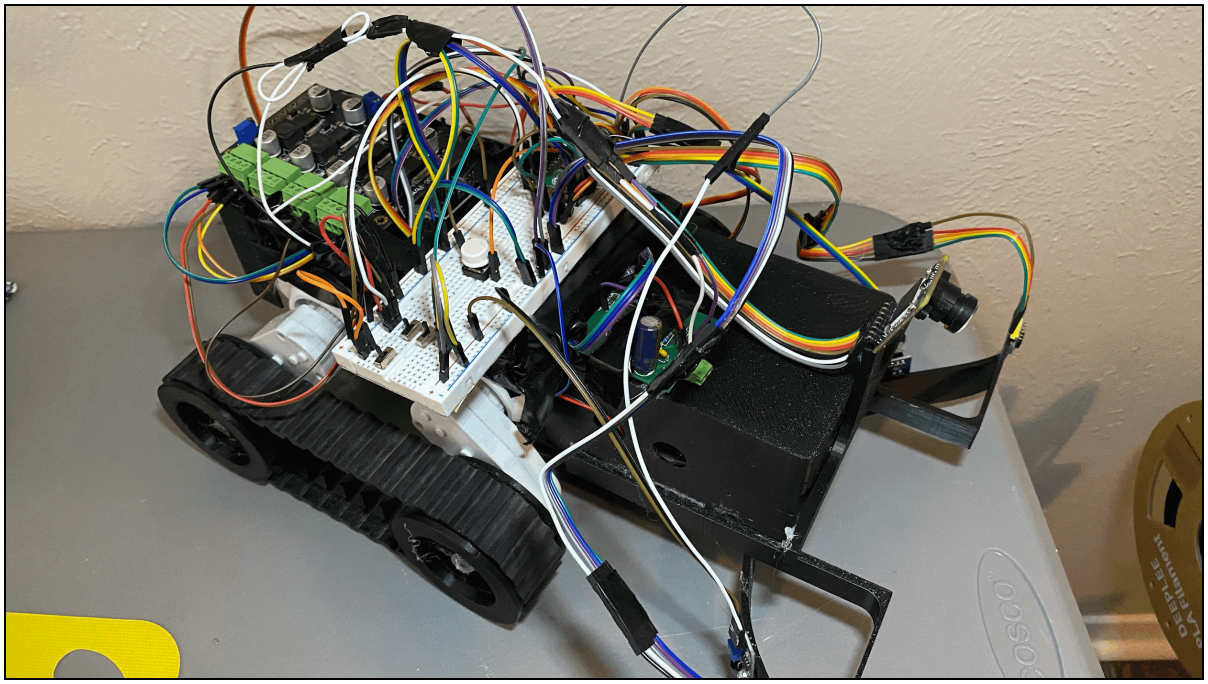

Figure 7. Complete Rover Image

3. Engineering Standards, Specifications, and Intellectual Property Considerations

The updated robot should follow relevant electrical and mechanical design practices including safe wiring, current-limited motor testing, secure battery mounting, strain relief on moving wires, and adequate spacing around rotating parts. PCB and wiring choices should follow standard trace/current guidelines and manufacturer datasheets for the motor drivers, sensors, and Raspberry Pi Pico 2. The software uses common embedded control concepts such as PWM motor control, finite-state decision logic, sensor polling, and image thresholding. Any custom CAD, intake geometry, firmware, or test procedures should be documented by the team for future reuse and attribution.

4. Safety, Public Health, and Welfare Considerations

Safety considerations are mainly electrical and mechanical. The ball motor contains moving rubber-band strands, so fingers, loose wires, and clothing should be kept away during operation. The servo gate can pinch or strike nearby parts if the mechanism is misaligned. Batteries and motor drivers can heat under load, so current draw and temperature should be checked during extended tests. The robot should always have a reliable stop procedure before testing autonomous code.

5. Global, Cultural, Social, Environmental, and Economic Factor Considerations

5.1 Economic

The updated design is economically favorable because it reuses the Dagu 5 chassis and replaces a complex conveyor with a simpler ball motor, funnel, servo, and color sensor. This reduces the number of precision printed parts and makes the mechanism easier to repair.

5.2 Environmental

The modular intake reduces waste because individual parts can be reprinted instead of replacing a larger assembly. Reusing the chassis also reduces the need for large custom printed structural parts.

5.3 Social and Global

The project demonstrates accessible robotics design using low-cost embedded hardware, 3D printing, simple sensors, and practical control logic. The same concepts can be adapted to educational robotics, warehouse sorting demonstrations, and low-cost autonomous mechanisms.

5.4 Sustainability

The design is sustainable from a maintenance perspective because the rubber-band strands are easy to replace, the servo gate is a simple actuator, and the sensor modules are reusable. Future iterations should label wiring and maintain spare printed parts for fast repair.

6. Integration and Looking Forward

The main integration path is to finalize the underbody mechanism, tune the color sensor thresholds with the actual lighting and ball colors, verify the servo gate positions, and run repeated end-to-end tests. The code already contains the core logic for initialization, searching, targeting, collection, rejection, boundary handling, obstacle handling, and final scoring. The remaining work is mainly mechanical reliability, field calibration, and test-driven tuning.

6.1 Updated State Machine and Sensor Fusion Architecture

SETUP/WAIT: Initialize hardware, read mode/goal/start switches, and wait for the start button.

CALIBRATE_CAMERA: Capture background frames and prepare the OV2640 blob detector.

SEARCH: Rotate and scan for ball blobs; if needed, move toward centerfield and repeat the search.

TARGET_CENTER: Use camera centroid error to rotate until the ball is near image center.

APPROACH: Drive in bounded steps based on apparent ball size while checking boundaries and obstacles.

FUNNEL_COMMIT: Continue forward after the target leaves the camera view so the ball enters the funnel.

COLOR_CONFIRM: Use the TCS34725 at the gutter/gate position to classify the ball.

ACCEPT_COLLECT: For red balls, raise the servo gate, roll forward, transfer to collector, lower gate, and resume collect mode.

REJECT/DEFEND_SHOT: For non-red balls, reverse the ball motor and shoot/eject toward the defend goal path.

FINAL_ATTACK_RELEASE: When time or collector capacity triggers scoring, navigate to the attack goal and release collected balls.

AVOID/ESCAPE: Interrupt any motion when IR boundaries or LIDAR obstacles require corrective action.

7. Conclusion

The final robot design now centers on a compact, code-driven intake and sorting architecture under the Dagu 5 rover chassis. The updated hardware combines an ArduCAM OV2640 camera for long-range ball detection, a TCS34725 color sensor for final close-range color confirmation, a servo gate for mechanical sorting, and a rubber-band-strand ball motor for both intake and shooting. This approach is simpler and more packageable than the original belt conveyor while still supporting autonomous search, approach, collection, rejection, and scoring. The firmware integrates camera blob detection, sensor polling, motor control, gate control, obstacle avoidance, boundary response, and game-timer behavior into one C/C++ control loop. Future work should focus on mechanical tuning, color threshold calibration, repeated field tests, and reliability improvements in the funnel-to-gutter transfer path.

References

1. Raspberry Pi Ltd. (2024). Raspberry Pi Pico 2 Datasheet. August 2024. PDF Link

2. OmniVision Technologies. OV2640 CameraChip Datasheet. PDF Link (OmniVision/UCTRONICS)

3. ArduCAM. ArduCAM Mini Module Camera Shield / OV2640 Documentation. Product Page & Documentation

4. AMS/TAOS. (2012). TCS3472 Color Light-to-Digital Converter Datasheet. PDF Link (Adafruit/TAOS)

5. CEVA. BNO080/BNO085 9-Axis IMU Datasheet. PDF Link (CEVA)

6. STMicroelectronics. VL53L1X Time-of-Flight Ranging Sensor Datasheet. PDF Link (ST)

7. Texas Instruments. DRV8876 H-Bridge Motor Driver Datasheet. PDF Link (TI)

8. International Table Tennis Federation (ITTF). Table Tennis Ball Specifications (diameter 40 mm). Technical Leaflet T3 Reference (Official rules detail spherical ball with 40 mm diameter and 2.7 g weight).

Appendix A: Budget

The final budget should be updated with actual purchased parts. The updated mechanism likely requires the following categories:

Project Lab 1	***Running Total***			***Total Estimate***		
Direct Labor:						
Category or individual:	*Rate/Hr*	*Hrs*		*Rate/Hr*	*Hrs*	
Sergio	15	80	$1,200.00	15	80	$1,200.00
Atharva	15	80	$1,200.00	15	80	$1,200.00
Sam	15	80	$1,200.00	15	80	$1,200.00
DL Subtotal (DL)		Subtotal:	$3,600.00		Subtotal:	$3,600.00
Labor Overhead	*rate:*	0%	$0.00	*rate:*	0%	$0.00
Total Direct Labor (TDL)			$3,600.00			$3,600.00
Contract Labor:						
Instructor	0	0	$0.00	0	0	$0.00
Total Contract Labor (TCL)			$0.00			$0.00
Direct Material Costs:						
Arducam Camera			$32.00			$32.00
Multichannel Switching PSU module			$12.00			$12.00
Laser Ranging Sensor Module			$22.00			$22.00
Custom Wheels			$1.68			$1.68
Custom PCB Motor Driver x5			$45.00			$45.00
3D printing filament			$20.00			$20.00
Bn085 Sensor			$17.00			$17.00
Motor Driver IC DRV8876PWR			$70.00			$70.00
Belt motor			$2.00			$2.00
IR Sensors x4			$6.00			$6.00
9G servo			$2.00			$2.00
TCS3472 Color Sensor			$13.00			$13.00
Metallic wire (for rods)			$20.00			$20.00
Misc. Items			*$30.00*			*$30.00*

Total Direct Material Costs: (TDM)			$292.68			$292.68
Equipment Rental Costs:	Value	Rental Rate		Value	Rental Rate	
	$0.00	0.20%	$0.00	$0.00	0.00%	$0.00
Total Rental Costs: (TRM)			$0.00			$0.00
Total TDL+TCL+TDM+TRM			$3,892.68			$3,892.68
Business overhead		0%	$0.00		0%	$0.00
Total Cost:		**Current**	$3,892.68		**Estimate**	$3,892.68
SUMMARY:	**Current**	**Est. Final**				
Labor + OH	$3,600.00	$3,600.00				
Materials & Equip Total	**$292.68**	$292.68				
Materials & Equip Rental	$0.00	$0.00				
TOTAL	**$3,892.68**	$3,892.68				

Table 4. Project budget categories

Appendix B: Progress Chart

	%			
Task Name	done	Atharva	Sergio	Sam
MILESTONES				
Interim Report 14 Mar				
Final Presentation - 12 May				
Demo Day - 5 May				
SUMMARY	100.0			
Sensor Circuit	**100.0**			
Motor Driver Circuit	**100.0**			
Camera/Vision Circuit	**100.0**			
All Coding	**100.0**			
Arm/Bucket mechanism	**100.0**			
3D Printing	**100**			
Compile Entire Project	**100.0**			
DETAILED				
Sensor Circuit	**100.0**			
Reseach Circuit	100	xx	xx	xx
Construct Virtual Circuit	100	xx		
Test Virtual Circuit	100	xx		
Obtain Necessary Components	100	xx		
Construct Actual Circuit	100	xx		
Troubleshoot Circuit	100	xx	xx	xx
Construct Finalized Circuit	100	xx		
Motor Driver Circuit	**100.0**			
Reseach Circuit	100		xx	
Construct Virtual Circuit	100		xx	
Test Virtual Circuit	100		xx	
Obtain Necessary Components	100		xx	
Construct Actual Circuit	100	xx	xx	xx
Troubleshoot Circuit	100	xx	xx	xx
Construct Finalized Circuit	100	xx	xx	xx
Camera/Vision Circuit	**100.0**			
Reseach Circuit	100	xx		
Construct Virtual Circuit	100	xx		
Test Virtual Circuit	100	xx		
Obtain Necessary Components	100	xx		
Construct Actual Circuit	100	xx		xx
Troubleshoot Circuit	100	xx	xx	xx
Construct Finalized Circuit	100	xx		xx

All Coding	**100.0**			
Research Code	100	xx	xx	xx
Construct Flow Chart	100	xx	xx	xx
Write Code	100	xx	xx	xx
Test Code	100	xx	xx	xx
Compile Code	100	xx	xx	xx
Debug Code	100	xx	xx	xx
Arm/Bucket Mechanism	**100.0**			
Research Design	100			xx
Create Design in CAD	100	xx		xx
Test Print Design	100	xx		
Iterate/Update Design	100	xx		xx
Print Final Design	100	xx		
Troubleshoot Issues	100	xx		xx
Compile Entire Project	**100.0**	xx	xx	xx
Troubleshoot all software	100	xx	xx	xx
Compile all circuits together	100	xx	xx	xx
Troubleshoot and resolve all issues	100	xx	xx	xx

Table 5. Project progress

Appendix C: Main Control Logic Summary

1. Initialize I2C0, I2C1, SPI1, LIDARs, IMU, TCS34725, ArduCAM OV2640, motors, switches, and servo gate.

2. Wait for the start button and read rover mode, defend goal color, attack goal color, and start side.

3. Calibrate the camera background and enable the motion camera watch if calibration succeeds.

4. Start the ball motor in collect mode and keep the servo gate down.

5. In the main loop, poll LIDAR and IMU continuously and periodically poll IR, TCS34725, and camera frames.

6. If the stop time is reached, stop all motion and lower the servo gate.

7. If a boundary or obstacle is detected, interrupt motion and execute escape/avoid behavior.

8. If the final scoring window is reached or the collector is full, release gutter/collector balls toward the appropriate goal.

9. If the color sensor confirms a ball at the gate, handle it as red accept/collect or non-red defend-shot.

10. If the camera sees a valid target, center the target, approach by distance estimate, and commit into the funnel.

11. If the camera loses the target near the robot, drive a remaining commit distance so the ball reaches the funnel.

12. If no target is visible, rotate/search, perform larger scans, or reposition toward centerfield.

Appendix D: ArduCAM Blob Logic Summary

1. Verify ArduChip SPI communication and sensor I2C presence.

2. Initialize the OV2640 for RGB565 160 x 120 frames.

3. Capture frames through the ArduCAM FIFO.

4. Build a calibrated background from multiple frames.

5. Maintain temporal RGB averages to reduce noise.

6. Compare current pixels against background bands and reject low-change pixels.

7. Classify saturated foreground pixels as red, green, or blue using normalized RGB thresholds.

8. Group adjacent color pixels into connected components.

9. Reject very small components and store the largest ball candidates.

10. Compute centroid, bounding box, size, average RGB, region, and surface color for each ball.

11. Sort detections by area and pass the result to the main game logic.