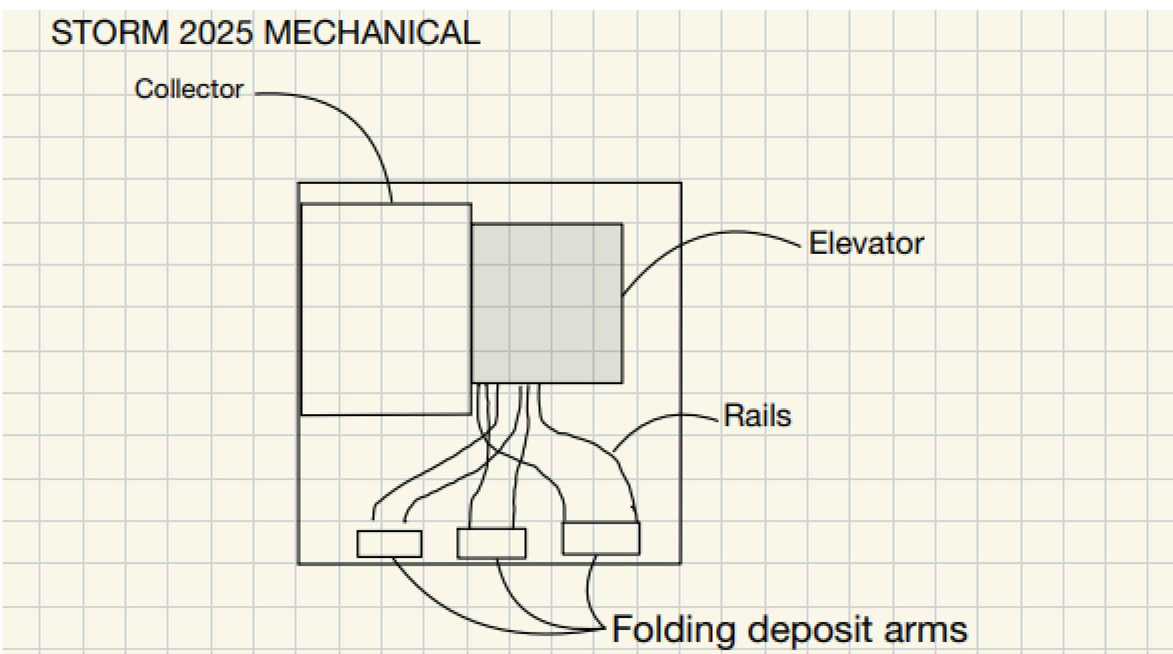
Introduction

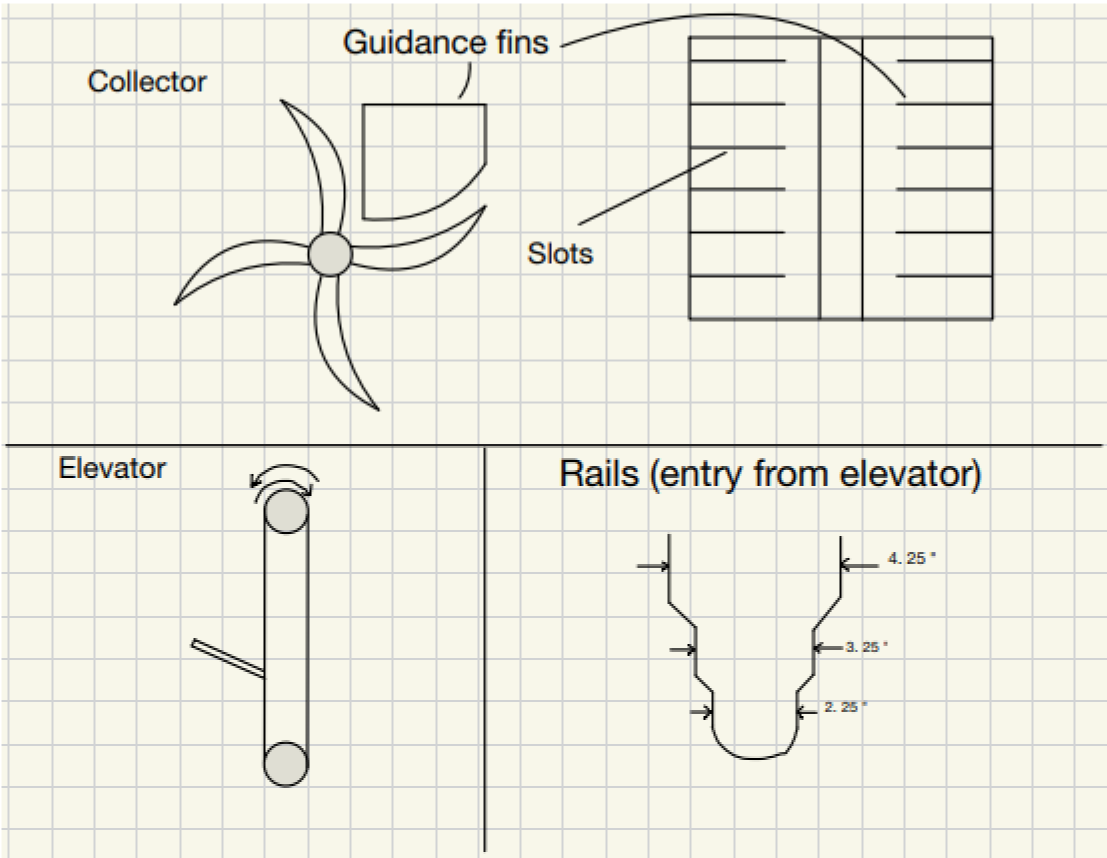
We are Wildcat Robotics, the Robotics Club at Kansas State University. Sponsored by the Department of Electrical and Computer Engineering, Department of Mechanical Engineering, and Department of Computer Science, we compete in various national collegiate robotics competitions. Our club consists of multiple teams: us, the Autonomous Vehicle Competition (AVC) team, and numerous combat robotics teams who participate in and host BattleBots competitions. We are proud to have been one of the inaugural teams competing in the STORM competition last year, and we look forward to participating again this year. The 2024-2025 AVC team consists of:

Name	**Contributions**
John Woods	Team Captain, programming
Josh Caldwell	Programming
Jonathan Hoepner	Electrical design
Jacob Mitchell	Robot design and construction
Jack Lilley	CAD, robot design and construction

Robot Layout



Mechanical Design



This bot uses a standard mecanum wheel drivetrain.
At the beginning of the match, arms for depositing hail fold upward. Hail is collected using a slotted, aptly named collector and transported to the elevator. This process uses fins that the slotted connector “screens” through to guide the hail. The hail is then lifted to an entry point for rails that guide it to its appropriately sized and located storage to be released onto the deposit arms.

Electrical Design

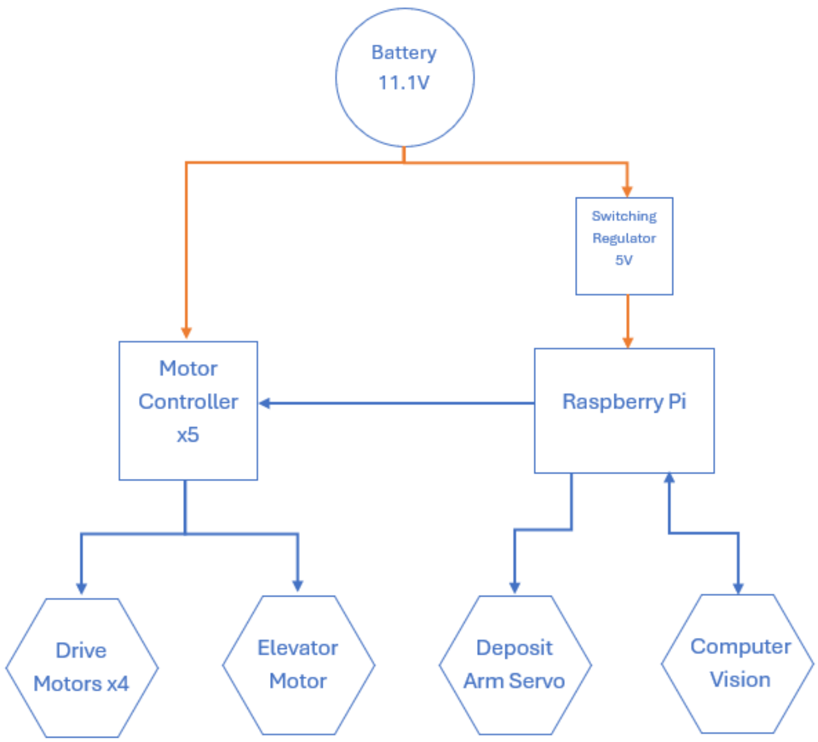
The main brain of the robot is going to be a Raspberry Pi 5. This allows for ease of access to cameras and connection to Wi-Fi to control the robot remotely. The GPIO pins can drive a servo motor for the deposit arm allowing for the release of hail into the bins. Lastly, the Pi will control five motor controllers that will drive the four main drive motors and the fifth will control the elevator motor.

<u>Sensors</u>

The only sensor we are using this year is the camera that supplies the operator with visual information of the course. The camera we are using is a standard Raspberry Pi branded camera.

Power Delivery

To power the robot, we will be using two Tattu 11.1V lithium polymer batteries wired in parallel for more battery life. To power the Raspberry Pi, the battery voltage will be run through a DC-DC switching buck converter to supply the Pi with its required 5V. The deposit arm servo will be supplied using 5V from the GPIO pins on the Pi. All other components will be supplied with the full battery voltage to supply the motors with their required power.



Software Design

Our codebase runs on a Raspberry Pi 4 model B and is split into two parts: one part is the code which runs on the robot, and the other part is the code for the driver interface webserver.

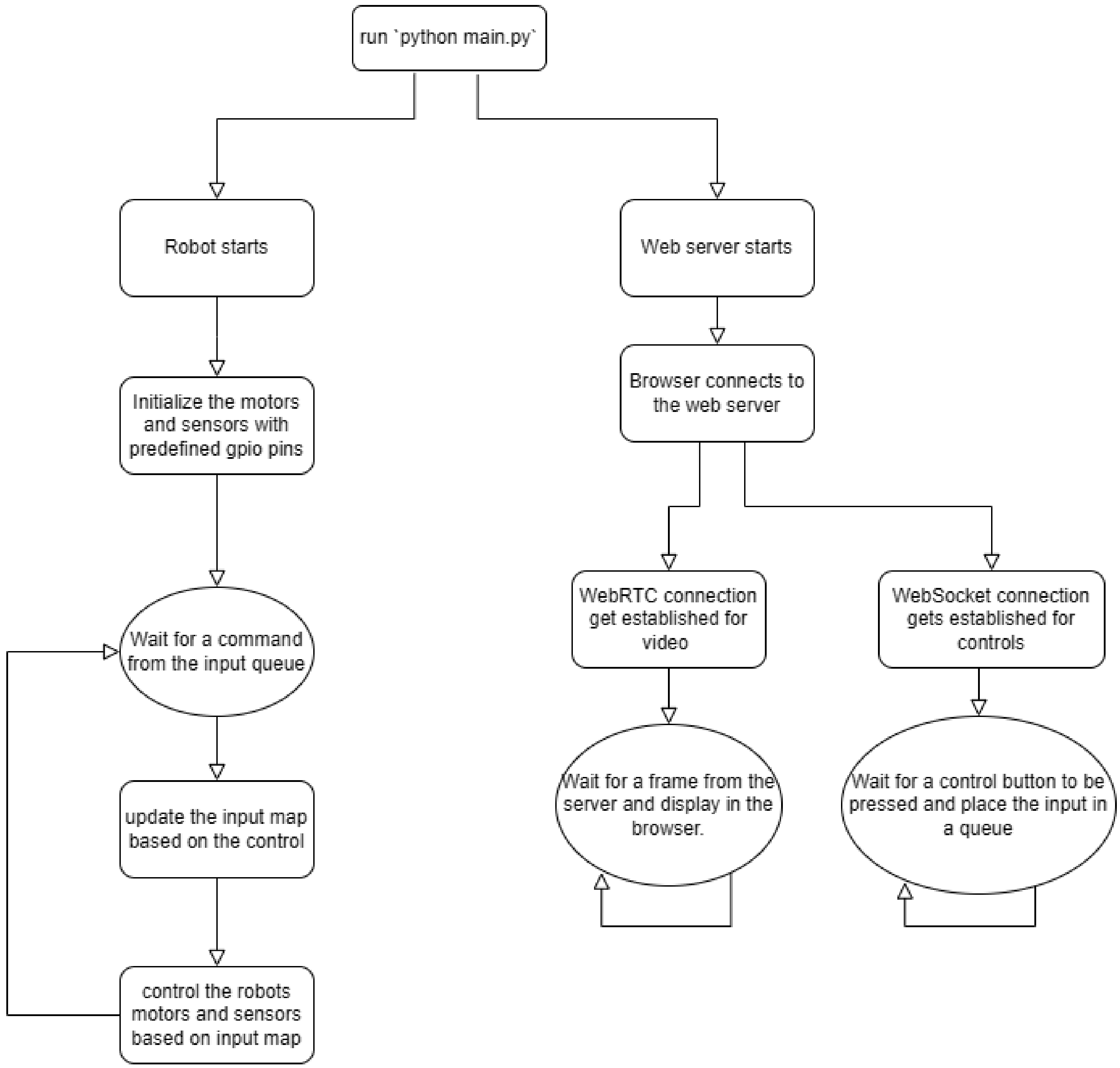
Robot Code

The robot code is written entirely in Python and responsible for controlling all the systems of the robot. The motors are controlled with the gpizero library and get their instructions from controller data sent by the driver interface via a websocket. To implement Mecanum drive, we use some math and logic to convert the controller input into percent power values for each of the four motors. Camera video data is received by the Picamera2 library, pre-processed into the right color space with OpenCV, and sent to the driver interface with WebRTC, which provides a fast and high-quality stream for the driver.

Driver Interface Code

The driver interface runs off a webserver hosted on the Raspberry Pi, written in Python with the aiohttp library. The driver can access the interface from Firefox by connecting to the robot's IP. For the UI, the driver interface simply contains a WebRTC video stream from the camera of the robot. Client-side JavaScript embedded in the driver interface webpage is used to collect controller input from the driver and send it to the robot via a websocket.

Below is a flowchart outlining the robot and webserver startup and running loop:


run `python main.py`
Robot starts
Web server starts
Initialize the motors and sensors with predefined gpio pins
Browser connects to the web server
WebRTC connection get established for video
WebSocket connection gets established for controls
Wait for a command from the input queue
Wait for a frame from the server and display in the browser.
Wait for a control button to be pressed and place the input in a queue
update the input map based on the control
control the robots motors and sensors based on input map

Safety and Reliability

For the mechanical design of the robot, we gravitated towards a simpler design that minimizes points of failure to ensure the robot is reliable. We are also machining parts for the robot with no sharp edges so you can't cut yourself, making the robot safer.

For the electrical design, we are putting the main electronics in a closed compartment to shield them from damage during matches, ensuring reliability. We are also putting care into how things are wired to ensure that there are no exposed wires and that wires are cable managed properly for a safer robot.

For the software, we largely rely on asynchronous programming to ensure that robot and driver communication is as snappy and reliable as possible. We have also designed the software in such a way that if the driver loses connection to the robot, the robot or any of its parts cannot move to ensure safety.

Bill of Materials

Item No.	Description	Quantity	Unit Cost	Amount
1	2" Styrofoam balls (30 ct.)	1	9.99	9.99
2	3" Styrofoam balls (24 ct.)	1	21.49	21.49
3	4" Styrofoam balls (24 ct.)	1	22.99	22.99
4	80mm Mecanum wheels (4 ct.)	1	33.99	33.99
TOTAL				88.46

These are the new parts we purchased this year for the robot, the rest of the robot was constructed from reused parts.