Pre-Alpha Build Report

Haoyuan Xu
Computer Engineering Department
University of Florida
Gainesville, United States
haoyuan.xu@ufl.edu

Jingxuan Xiao
Computer Engineering Department
University of Florida
Gainesville, United States
j.xiao@ufl.edu

***Abstract*—This report outlines the pre-alpha build of the Rover Master project, detailing the efforts in software and hardware development, architectural elements, and information handling systems. It includes the research and selection of models for monocular depth estimation and object detection, the design of the main control board PCB, and the development of internal systems such as the motion planning system and IMU fusion sensor.**

***Index Terms*—Rover Master, Monocular Depth Estimation, Object Detection, PCB Design, ROS2, IMU Fusion Sensor, Motion Planning System**

I. Introduction

The Rover Master project aims to develop an autonomous robot capable of navigating and interacting with its environment using advanced sensing and control systems. This pre-alpha build report details the efforts made in both software and hardware development, architectural design, and the implementation of internal systems that contribute to the robot's functionality.

II. Effort

A. Software

We researched the newest open-source models for monocular depth estimation and object detection. By comparing the results between multiple potential models and evaluating the feasibility of each model for our project, we identified the best models that fit our goals:

- **Monocular Depth Estimation**: https://marigoldmonodepth.github.io/
- **Object Detection**: https://github.com/ultralytics/ultralytics

B. Hardware

1) Web Scraper for Building KiCAD Library: We utilized a web scraper (https://github.com/zhangyx1998/easyeda-scraper) to build a KiCAD library by scraping electronic components information from LCSC.com. The program extracts critical data from the API call and formats it into a component library compatible with KiCAD. Although in its primitive stage, requiring the component part number to be specified for every run, this tool ensures that every part in our design corresponds to an existing electronic component available for purchase online.

2) Main Control Board PCB Design: The main control board PCB design is available at https://github.com/Rover-Master/RoverMaster-PCB.

- An STM32F05 microcontroller is used for communication with ESCs, voltage, and current sensing. Initially, an ESP32C3 was considered due to its Wi-Fi and Bluetooth capabilities; however, since wireless communication is not required at this stage, the STM32 was chosen to minimize size.
- A BNO055 sensor is used for gyroscope and acceleration data.
- The PD charging unit is under research. Two chips are selected, TPS25750 and BQ25792. TPS25730 was considered. The 50 version allows more configuration using GPIOs, while the 30 version's configuration is more restricted. Multiple resistor ladder is required for configuring 30. BQ25792 is a mature solution for charging 1-4s batteries while providing power to system load. The difficulty lies on how to design the peripheral circuit based on our needs, and how to design the circuit for configuration.
- The Battery Management System(BMS) is also under research. Passive balancing is not considered due to efficiency. Capacitor balancing is chosen over inductor balancing because it is easier to set up and control. For the capacitors within BMS, aluminum electrolytic capacitors are chosen over ceramic capacitors. Although electrolytic capacitors are bigger in size, they provide better stability under constant DC bias, and they have bigger capacitance. The difficulty lies on how to design the mosfet bridges and how to simplify the full bridges to half bridges.

III. Architectural Elements

A. External Interface

1) Camera: Captures frames showing objects and the environment for processing by the object detection and depth estimation models.

2) Robot Arm: Equipped with multiple sensors and can be controlled to pick up or release objects.

3) Charging: The robot accepts 20V PD charging via a Type-C cable, with the BMS managing the charging process.

4) Control: Currently, a controller is connected to a laptop, and control information is forwarded from the laptop to the robot.

5) Website: The website provides the following features:

a) Dashboard Overview: Displays the robot's current status (active, idle, error) and real-time metrics like battery level and connection status.

b) Live Video Feed: Streams live video from the robot's camera to view its surroundings.

c) Sensor Data Display: Shows real-time sensor readings, such as depth estimation results and detected objects with their locations.

d) Action Logs: Records actions performed by the robot, including items picked up or dropped, navigation history, and errors encountered.

e) User Controls: Allows users to send commands to the robot, like moving to a location or picking up objects.

f) Data History: Provides access to past performance data and action logs.

g) Settings: Enables users to adjust the robot's operation settings.

h) Alerts: Notifies users of critical issues, such as low battery or errors.

B. Persistent State

1) 3D Map: Stores the environment, object locations, depth estimates, navigation paths, and model calibration data. This allows the robot to recall information in future tasks without reprocessing everything from scratch.

2) Grasping Points and Object Manipulation Data: Stores information about the best way to grasp specific objects, avoiding the need to recalculate each time.

3) Environment Interaction Data: Stores data about how the robot interacts with the environment when performing tasks, such as successful navigation paths or avoidance strategies based on depth and object information.

C. Internal Systems

1) Object Detection Model: Responsible for identifying and classifying objects within frames captured by the robot's camera using advanced algorithms like convolutional neural networks (CNNs). Outputs include bounding boxes and labels indicating the type of object detected.

2) Depth Estimation Model: Measures the distance to detected objects using data from the camera, employing techniques like stereo vision or monocular depth estimation. Calculates spatial relationships between the robot and objects, crucial for navigation and collision avoidance.

3) Motion Planning System: Leverages data from the depth estimation and object detection models to determine precise arm movements. Calculates optimal position, orientation, and trajectory for tasks like picking up or releasing objects, considering factors such as object size, shape, distance, and environmental obstacles.

4) IMU Fusion Sensor: Provides 9 DOF data by integrating accelerometer, magnetometer, and gyroscope data into a stable three-axis orientation output. Enables accurate posture and motion tracking of the robot.

5) MCU on the Main Board: The microcontroller manages the Electronic Speed Controllers (ESCs) for motor control, ensuring precise motor speed and direction adjustments. It also monitors voltage and current from the power distribution system, using sensors to track battery health and power consumption in real-time for efficient energy management.

IV. Information Handling

1) Communication: We utilize Robot Operating System 2 (ROS2) to facilitate communication between the different modules of the robotic system. ROS2 provides a publish-subscribe architecture, allowing modules to send and receive messages asynchronously. Each module communicates via topics, where data such as camera feeds and object detection results are published and subscribed to as needed. ROS2 also supports services for synchronous communication, enabling modules to request specific data or actions and receive responses.

2) Integrity and Resilience:

a) Data Integrity: We ensure accuracy and consistency between the camera, depth estimation model, object detection model, and robot arm by:

- Using checksums to validate camera frames before processing.
- Including timestamps to maintain data order.
- Cross-referencing object detection results with depth information to detect inconsistencies.

b) System Resilience: We manage benign interference from environmental factors using sensor fusion for reliable data input. We implement automatic retries or fallback algorithms in case of detection errors.

c) Recovery Mechanisms: We support graceful degradation to operate in limited capacity during component failures and enable rollback to a valid state, retrying operations after data corruption or processing errors.

d) Logging and Alerts: We maintain error logs for communication failures and sensor malfunctions and generate alerts for critical issues requiring immediate operator attention.

V. Links

- **Project Repository**: https://github.com/haoyuanxu430/Rover-Master-Logs/
- **Video Overview**: https://youtu.be/6k8LeupdFZE