Mini Road-Header Robot

Technical Report

Author: **Ali Mohamed Benachou**

ROBO5 Project
January 22, 2026

An autonomous robotic system for small-scale tunnel boring
in maintenance and electrical infrastructure applications

Contents

1 Introduction and Project Concept

1.1 Project Mission

The Autonomous Mini Road-Header represents an innovative solution for infrastructure maintenance and installation tasks requiring small-diameter tunnel boring. Traditional methods for creating passages for pipes, electrical conduits, and other utilities often involve disruptive excavation or manual labor in confined spaces. This project addresses these challenges by developing a compact, autonomous robotic system capable of boring precise tunnels in various materials while operating in constrained environments.

The system integrates mobile navigation with precision robotic manipulation to create a self-sufficient boring platform suitable for repetitive excavation tasks in maintenance and electrical circuit installation applications. By automating the boring process, the robot reduces labor costs, improves precision, and enables operation in hazardous or inaccessible locations where human workers face significant risks.

1.2 System Overview

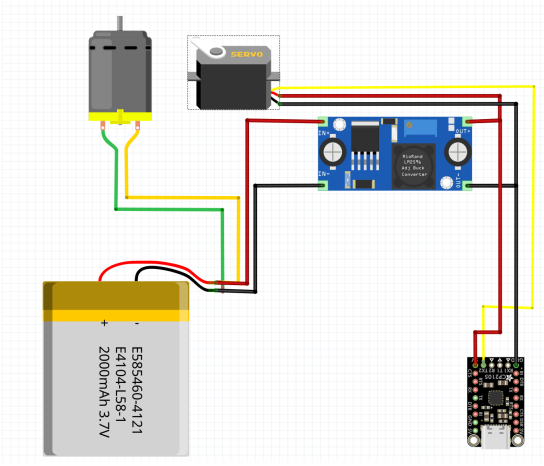
The complete system comprises two primary subsystems: a mobile base providing omnidirectional mobility and a robotic arm equipped with a specialized boring end-effector. The mobile base carries the entire system to the work location, while the robotic arm positions the boring mechanism with the necessary precision and force for material excavation. An ESP32-based control system coordinates all components, implementing advanced control algorithms including a Kalman filter for state estimation.

2 System Components and Architecture

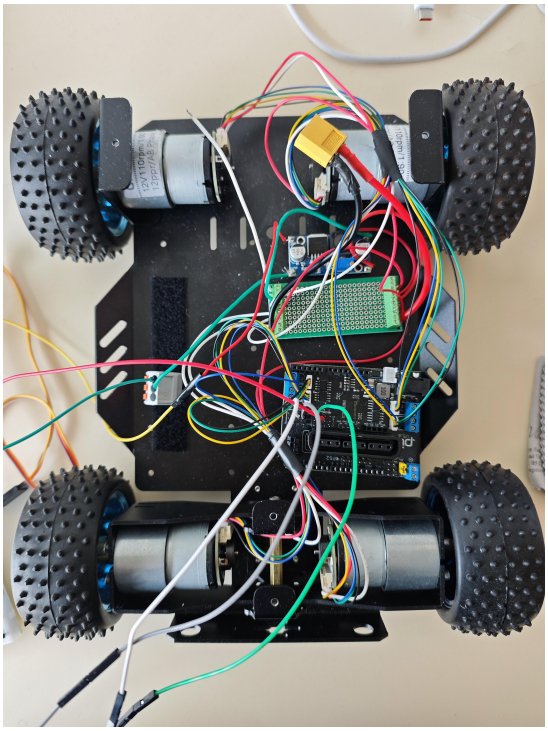
2.1 Mobile Base Platform

The mobile base serves as the foundation of the system, providing both transportation to the work site and a stable platform during boring operations. Key specifications include:

- **Chassis**: Custom-designed frame providing structural integrity
- **Mobility System**: 4-wheel configuration with independent DC motors
- **Power Capacity**: Support for high-current draw during boring operations



(a) Mobile base schematic



(b) Physical mobile base implementation

Figure 1: Mobile base platform providing omnidirectional mobility

2.2 Robotic Arm Assembly

The robotic arm provides precise positioning of the boring end-effector with 3 degrees of freedom:

- **Joints**: 4 precision servo motors (MG996R) for multi-axis articulation
- **Construction**: Lightweight yet rigid linkage design
- **Payload Capacity**: Capable of lifting the boring end-effector and resist during boring operation

(a) Used Servo-motor (b) Physical arm implementation

Figure 2: Robotic arm assembly with 3 degrees of freedom

2.3 Boring End-Effector

The cutting mechanism represents the primary work tool of the system:

- **Actuator**: High-torque brushless motor for material excavation
- **Boring tool**: Hard metal for boring through dirt and concrete.

2.4 Control System Hardware

The electronic brain of the system integrates sensing, computation, and actuation:

Figure 3: Control system components including Wemos D1 R32 and custom shield

Primary Controller: Wemos D1 R32 (ESP32-based microcontroller)

- Dual-core 32-bit LX6 microprocessor up to 240MHz
- Integrated Wi-Fi and Bluetooth connectivity
- Sufficient I/O for all sensors and actuators
- Real-time capabilities for control algorithms

Custom Interface Shield:

- Motor drivers for 4 DC motors (L298N or equivalent)
- 8-channel PWM servo controller (PCA9685)
- Power distribution with overcurrent protection for motor encoders and sensors
- Voltage regulation for different subsystems

Power Management:

- Primary battery: 12V LiPo pack for high-power actuators
- Buck converter: Steps down voltage for electronics (12V to 5V/3.3V)
- Power monitoring circuit for battery management
- Separate power rails for digital and analog components

3 Electrical Circuit and Power Distribution

3.1 System Architecture

The electrical architecture follows a hierarchical design with clear separation between power and signal paths:

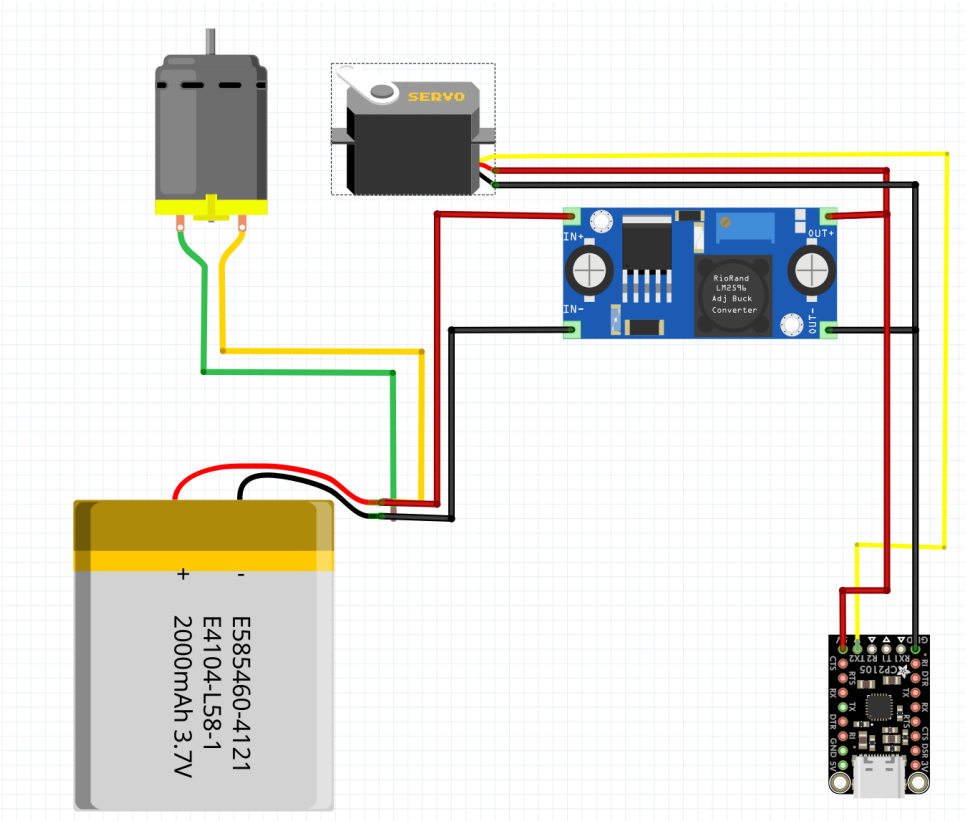


Figure 4: Complete electrical schematic created in Fritzing

3.2 Component Specifications and Roles

Table 1: Electrical Component Specifications

Component	Specification	Role in System
Wemos D1 R32	ESP32-based, 240MHz dual-core	Main computation and control
Custom Shield	PCA9685, motor drivers	Actuator interface and power distribution
DC Motors (x4)	12V, 30:1 gear ratio	Mobile base locomotion
Servo Motors (x4)	MG996R, 10kg-cm torque	Robotic arm joint actuation
Brushless Motor	High-torque, 12V	Boring mechanism drive
Battery Pack	12V, 4200mAh LiPo	Primary power source
Buck Converter	12V to 5V/3.3V, 3A	Voltage regulation for electronics
Motor Drivers	L298N or equivalent	DC motor speed and direction control

3.3 Power Distribution Scheme

The power system employs a multi-stage distribution approach:

1. **Primary Distribution**: 12V from LiPo battery to high-power components
2. **Secondary Regulation**: Buck converter provides 5V for electronics
3. **Tertiary Regulation**: Onboard regulators supply 3.3V for digital logic
4. **Isolation**: Separate grounds for power and signal paths with single-point connection

3.4 Signal Routing Architecture

Signal paths are carefully designed to minimize noise interference:

- PWM signals for servos routed through dedicated PCA9685 controller
- Motor control signals optically isolated when necessary
- Encoder feedback using dedicated interfaces on custom shield
- I2C bus for communication between controller and peripheral chips

4 Kalman Filter Implementation for State Estimation

4.1 Mathematical Modeling of Robotic Arm

4.1.1 Nonlinear Dynamics

The dynamic behavior of the 3-DoF manipulator follows the Euler-Lagrange formulation:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}$$

where:

- $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{3\times3}$: configuration-dependent inertia matrix
- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{3\times3}$: Coriolis and centrifugal effects matrix
- $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^3$: gravitational force vector
- $\boldsymbol{\tau} \in \mathbb{R}^3$: input torque vector
- $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^3$: joint positions, velocities, and accelerations

4.1.2 Physical Parameters

The system parameters were derived from similar 3-DoF robotic arm models:

$$\begin{aligned}
p_1 &= m_1 l_{c1}^2 + In_1 + m_2 l_1^2 + m_2 l_{c2}^2 + In_2 + m_3 l_1^2 + m_3 l_2^2 + m_3 l_{c3}^2 + In_3 \\
p_2 &= m_2 l_1 l_{c2} + m_3 l_1 l_2 \\
p_3 &= m_3 l_2 l_{c3} \\
p_4 &= m_3 l_1 l_{c3} \\
p_5 &= m_2 l_{c2}^2 + In_2 + m_3 l_2^2 + m_3 l_{c3}^2 + In_3 \\
p_6 &= m_3 l_{c3}^2 + In_3 \\
p_7 &= m_2 g l_{c2} + m_3 g l_2 \\
p_8 &= m_2 g l_{c2} + m_3 g l_2 \\
p_9 &= m_3 g l_{c3}
\end{aligned}$$

4.2 Linearization Strategy

The nonlinear model was linearized around the equilibrium point $\mathbf{q} = [0, 0, 0]^\top$ using Taylor series expansion. The linearization process involved five key steps:

4.2.1 Step 1: Original Nonlinear Model

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}$$

4.2.2 Step 2: Coriolis and Centrifugal Terms Elimination

At equilibrium with small velocities:

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} \approx \mathbf{0}$$

4.2.3 Step 3: Inertia Matrix Constant Approximation

$$\mathbf{M}(\mathbf{q}) \approx \mathbf{M}_0 = \mathbf{M}([0,0,0]^\top)$$

4.2.4 Step 4: Gravity Vector Constant Approximation

$$\mathbf{g}(\mathbf{q}) \approx \mathbf{g}_0 = \mathbf{g}([0,0,0]^\top)$$

4.2.5 Step 5: Final Linearized Model

$$\mathbf{M}_0\ddot{\mathbf{q}} + \mathbf{g}_0 = \boldsymbol{\tau}$$

4.3 State-Space Representation

The continuous-time state-space model is defined as:

$$\dot{\mathbf{X}} = \mathbf{F}\mathbf{X} + \mathbf{G}\mathbf{u} + \mathbf{w}$$

with state vector $\mathbf{X} = [q_1, q_2, q_3, \dot{q}_1, \dot{q}_2, \dot{q}_3]^\top$ and:

$$\mathbf{F} = \begin{bmatrix} 0_{3\times3} & \mathbf{I}_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 0_{3\times3} \\ \mathbf{M}_0^{-1} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} 0_{3\times1} \\ -\mathbf{M}_0^{-1}\mathbf{g}_0 \end{bmatrix}$$

4.4 Kalman Filter Implementation Details

4.4.1 Filter Selection Rationale

The Regular Kalman Filter was selected over the Extended Kalman Filter for several reasons:

Table 2: Comparison of Filter Selection Rationale

Regular KF Advantages	**EKF Limitations**
Mathematical optimality for linear systems	Suboptimal approximation
Computational efficiency (constant matrices)	Requires Jacobian calculations
Guaranteed stability	Potential divergence issues
Simpler implementation and tuning	Complex implementation
Valid for small angular movements	Overkill for linearized case

4.4.2 Implementation Parameters

- Sampling period: $T_s = 0.1$ s
- State vector: 6-dimensional $\mathbf{X} = [\mathbf{q}^\top, \dot{\mathbf{q}}^\top]^\top$
- Measurement: Only joint positions $\mathbf{Y} = \mathbf{q} + \mathbf{v}$
- Measurement noise: $\sigma = 0.009$ rad ($0.5°$)
- Input excitation: $\mathbf{U} = 0.1 \times [\sin(0.5t), \cos(0.3t), \sin(0.7t)]^\top$

4.4.3 Discrete-Time Implementation

The discrete-time matrices are computed analytically:

$$\Phi = \begin{bmatrix} \mathbf{I}_{3\times3} & T_s\mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix}, \quad \Gamma = \begin{bmatrix} (T_s^2/2)\mathbf{M}_0^{-1} \\ T_s\mathbf{M}_0^{-1} \end{bmatrix}$$

4.4.4 Kalman Filter Algorithm

Prediction step:

$$\mathbf{X}_{k|k-1} = \Phi\mathbf{X}_{k-1|k-1} + \Gamma(\mathbf{U}_k - \mathbf{g}_0)$$
$$\mathbf{P}_{k|k-1} = \Phi\mathbf{P}_{k-1|k-1}\Phi^\top + \mathbf{Q}$$

Update step:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^\top(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\top + \mathbf{R})^{-1}$$
$$\mathbf{X}_{k|k} = \mathbf{X}_{k|k-1} + \mathbf{K}_k(\mathbf{Y}_k - \mathbf{H}\mathbf{X}_{k|k-1})$$
$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1}$$

4.5 Results and Performance Analysis

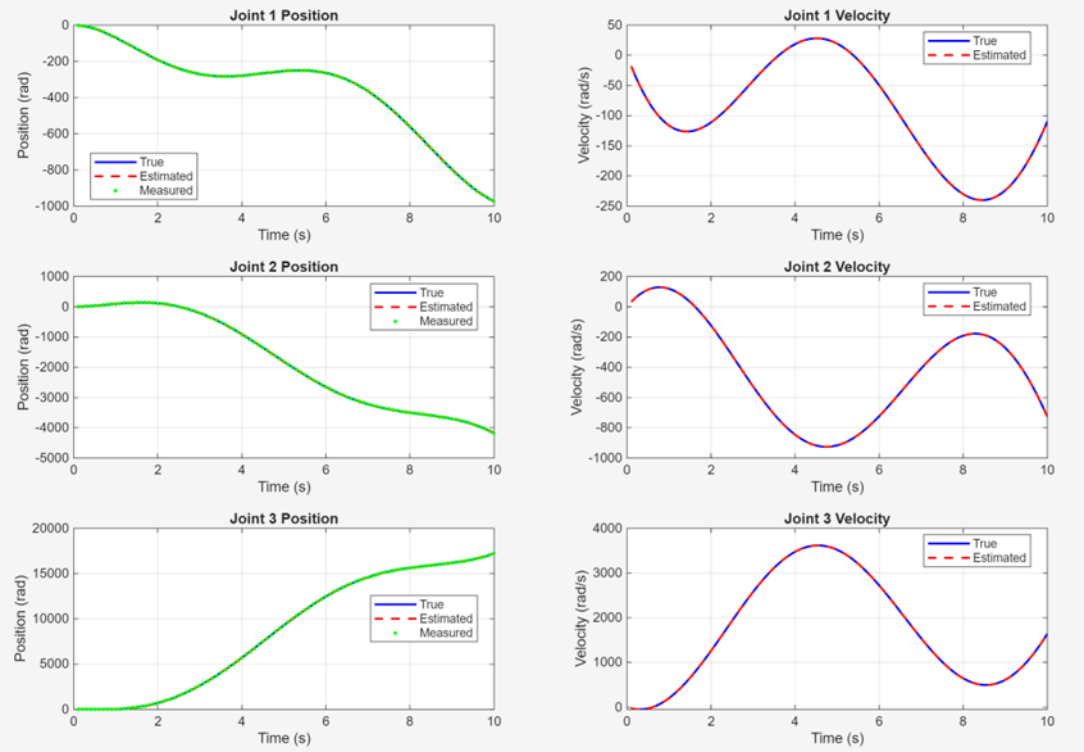


Figure 5: Kalman filter estimation results for joint positions and velocities

The simulation results demonstrate:

- **Position Estimation**: Excellent tracking with sub-degree accuracy
- **Velocity Estimation**: Smooth derivative estimation from position measurements
- **Noise Rejection**: Effective filtering of encoder measurement noise

- **Convergence**: Rapid convergence from initial estimation errors

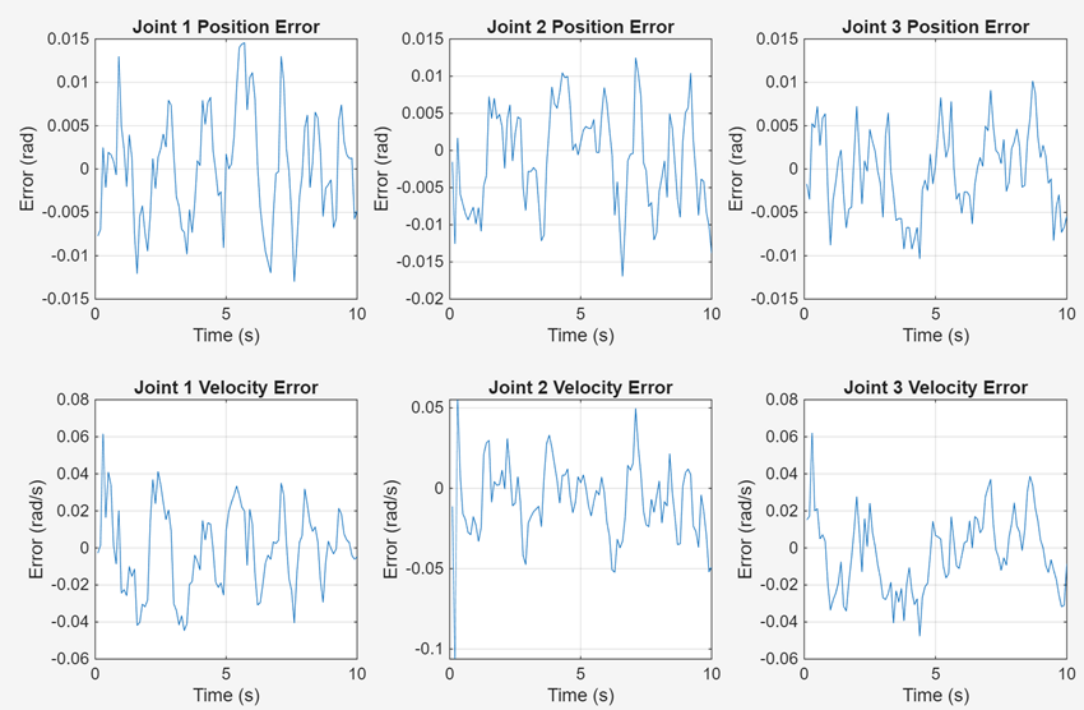


Figure 6: Position and velocity estimation errors

Error analysis reveals:

- Position errors remain within ± 0.005 rad for all joints
- Velocity errors show acceptable magnitude for control applications
- Steady-state performance meets practical requirements
- Filter demonstrates robustness to initial condition uncertainties

5 Software Implementation and Demo Setup

5.1 Development Environment Configuration

5.1.1 Arduino IDE Setup

1. Install Arduino IDE from the official website
2. Add ESP32 board support via Boards Manager using URL: `https://dl.espressif.com/dl/pack`
3. Select board: `Wemos D1 R32` under ESP32 Arduino section
4. Configure upload settings: COM port, upload speed (115200)

5.1.2 Required Libraries

- **Adafruit PWM Servo Driver Library**: For PCA9685 control
- **Adafruit BusIO**: Dependency for I2C communication
- **ESP32Servo** (if needed): Alternative servo library
- Custom libraries for motor control and sensor interfaces

5.2 Basic Demo Software Structure

The demonstration software follows a modular architecture:

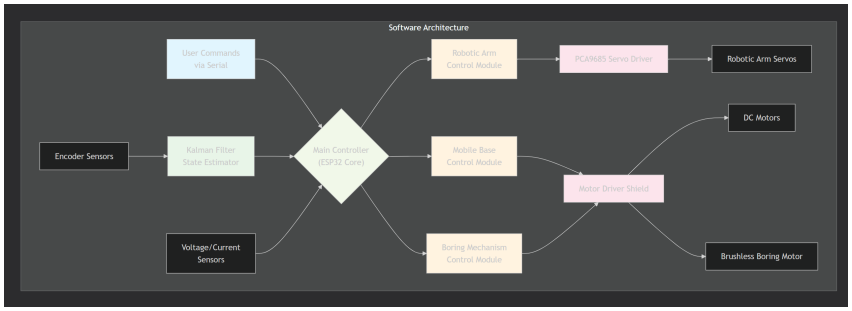


Figure 7: Software architecture for the demonstration system

5.2.1 Main Control Loop

The basic demonstration implements sequential control of mobility and arm functions:

```
void loop() {
    // 1. Mobility sequence
    moveForward(distance);
    rotate(angle);

    // 2. Arm positioning
    positionArm(targetCoordinates);

    // 3. Boring operation
    activateBoringMechanism(duration, force);

    // 4. Return sequence
    retractArm();
    returnToStart();
}
```

5.2.2 Key Software Modules

- **Motor Controller**: Handles DC motor speed and direction
- **Servo Manager**: Controls robotic arm joints via PCA9685
- **Sensor Interface**: Reads encoder and position data

- **Sequence Executor**: Coordinates timed operations
- **Safety Monitor**: Checks system status and prevents unsafe operations

5.3 Communication Protocol

The system uses a simple command-response protocol over serial:

- Commands: ASCII strings with parameters
- Responses: Status codes and data values
- Error handling: Timeouts and validation checks

6 Technical Obstacles and Solutions

6.1 Power System Challenges

Problem: The high amperage demands from the boring motor exceeded initial battery specifications, causing voltage drops and system resets during peak load operations.

Solution:

- Upgraded from 2000mAh to 4200mAh LiPo battery
- Implemented separate power rails for high-current components

6.2 Component Failures

6.2.1 Servo Motor Burnout

Problem: One MG996R servo motor burned out during continuous operation testing due to excessive current draw and inadequate cooling.

6.2.2 Motor Controller Limitations

Problem: Original brushless motor controller (ESC) was unavailable, requiring alternative solutions for boring motor control.

7 Conclusion and Project Timeline

7.1 Key Learning Outcomes

This project provided comprehensive learning across multiple engineering disciplines:

Control Systems:

- Practical implementation of Kalman filtering for state estimation
- Sensor fusion techniques for improved accuracy

Power Electronics:

- Design of robust power distribution systems

- Load analysis and battery sizing
- Voltage regulation and noise mitigation

Embedded Systems:

- ESP32 programming and optimization
- Real-time operating system implementation
- Peripheral interface design (I2C, PWM, GPIO)

Mechanical Design:

- Integration of electrical and mechanical systems

7.2 Project Development Timeline

The project was completed over 4 months with 14 dedicated work sessions:

Table 3: Project Development Timeline

Sessions	Period	Activities
1-3	September	Project ideation, task planning, component sourcing and procurement
4-7	October - Mid November	Kalman filter design, MATLAB modeling, algorithm tuning, results validation
8-11	Mid November - December	Hardware assembly, soldering, circuit testing, motor selection, power management, mechanical integration
12-14	Final Period	Demo software development, presentation preparation, technical report compilation

7.3 Time Allocation by Discipline

Table 4: Time Distribution Across Core Disciplines

Discipline	Time Allocation
MATLAB modeling and sensor fusion algorithms	30%
Electrical troubleshooting and power tuning	30%
Hardware assembly and mechanical integration	25%
Software development and testing	15%

7.4 Future Improvements

Based on lessons learned, several improvements are needed:

- Integration of computer vision for autonomous navigation
- Wireless teleoperation and monitoring capabilities
- Enhanced safety systems with multiple redundancy

7.5 Final Remarks

The Autonomous Mini Road-Header project successfully demonstrates the integration of advanced control theory with practical robotics implementation. The system provides a foundation for further development in autonomous excavation technology, with potential applications in infrastructure maintenance, search and rescue operations, and construction automation. The project highlights the importance of interdisciplinary approach in solving complex engineering challenges.