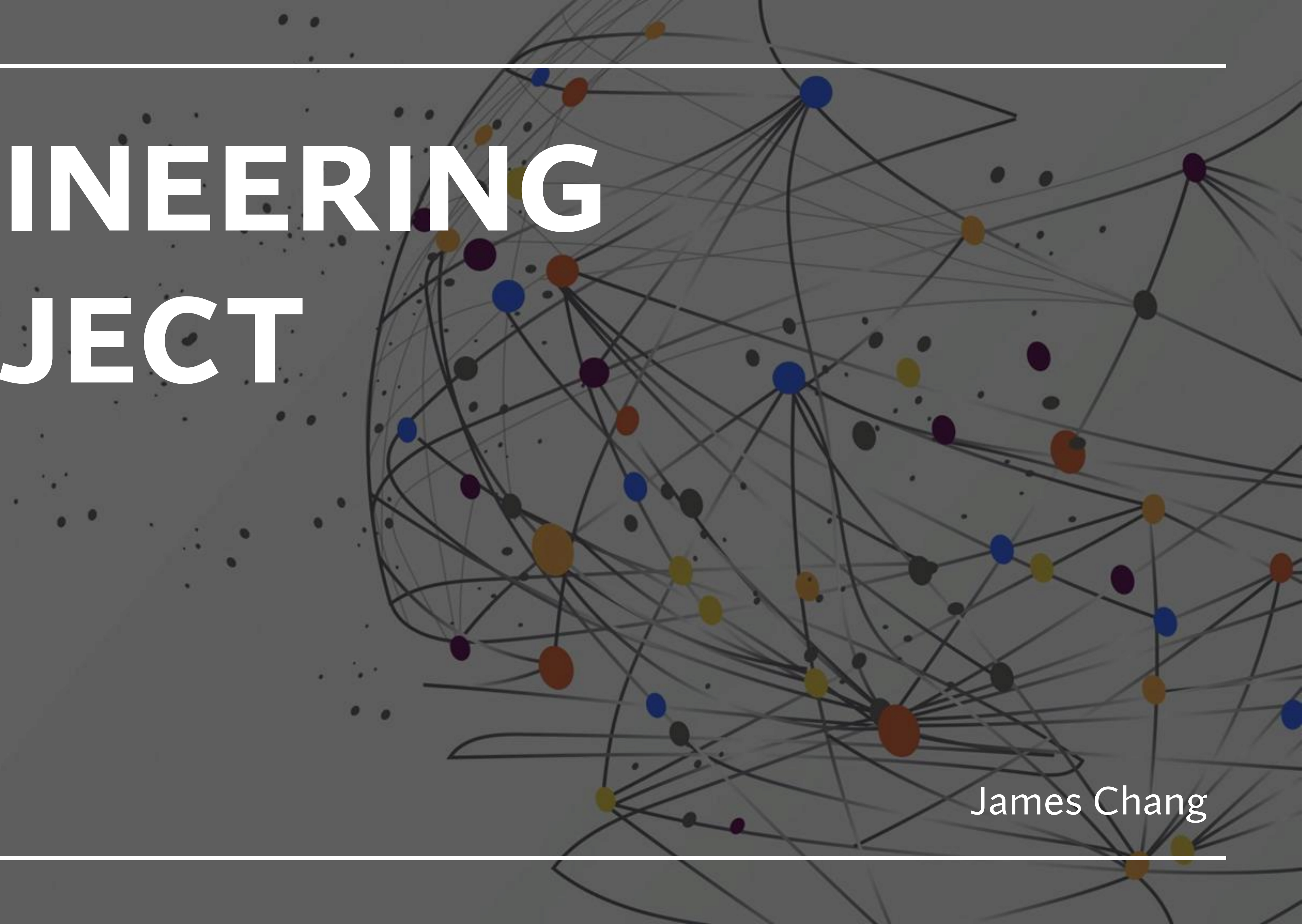


ENGINEERING PROJECT

James Chang

Academic project

Self project

Academic

Development of a Robotic Arm Interface with Image and Voice Database Integration

Project Overview

Collaborated with Hiwin Technologies to develop a real-time, intuitive robotic arm interface integrating both image and voice commands. The system combined PLC and HMI controls to simulate various electronic components, sensors, and motors, enabling multi-behavior task execution.

My Contributions

- Designed and built image and voice databases, and developed matching algorithms for command execution.
- Integrated gesture and voice recognition technologies to enable seamless multimodal interaction.
- Developed a client-server architecture separating recognition and motion control tasks.
- Implemented automation safety mechanisms to enhance the safety of human-robot interaction.

Key Features

Voice Recognition

Real-time speech-to-text processing that maps verbal commands to predefined robotic actions, enabling natural language interaction.

Gesture Recognition

Real-time hand gesture recognition using Yolov7 and MediaPipe.

Applied scaling and transformation matrices to accurately map image coordinates to robotic workspace coordinates.

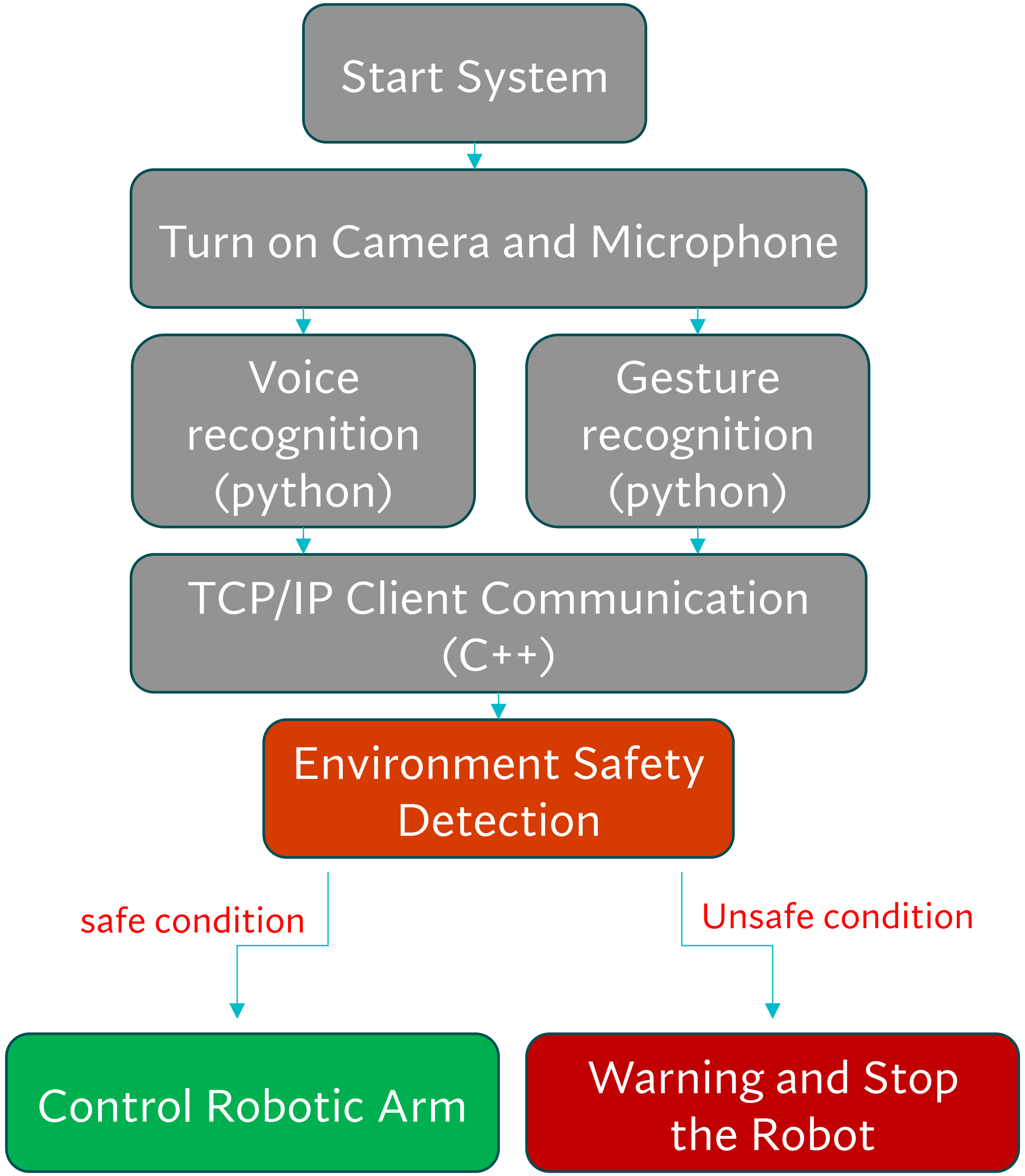
Client-Server Architecture

Python (Client): Handled voice and gesture recognition, translating inputs into control commands and coordinate data.

C++ (Server): Received commands via non-blocking TCP/IP sockets, executed inverse kinematics and motion interpolation, and controlled robotic movements through a robot control library.

Automation Safety Mechanism

Integrated proximity sensing and motion monitoring into the PLC system, implementing a collaborative safety alarm system that triggers real-time warnings or emergency stops to ensure safe human-robot interaction.


Start System
Turn on Camera and Microphone
Voice recognition (python)
Gesture recognition (python)
TCP/IP Client Communication (C++)
Environment Safety Detection
safe condition
Unsafe condition
Control Robotic Arm
Warning and Stop the Robot

Detailed System Integration Flow

1. **Start System**
   - •Initialize all modules and hardware.
2. **Turn On Microphone and Camera**
   - •Activate input devices:
     - •Microphone for voice capture.
     - •Camera for image capture.
3. **Voice Recognition (Python)**
   - •Capture live audio stream.
   - •Apply Speech-to-Text (STT) processing.
   - •Match recognized command with database.
   - •Generate corresponding robotic arm control command.
4. **Gesture Recognition (Python)**
   - •Capture real-time video frames.
   - •Detect hand landmarks using Yolov7 and MediaPipe.
   - •Analyze hand gestures.
   - •Apply scaling and transformation matrices to map 2D camera coordinates to 3D robot coordinates.
   - •Generate corresponding robotic arm control command.
5. **TCP/IP Client Communication**
   - •Package voice and gesture data into structured command format.
   - •Establish TCP/IP connection with server.
   - •Renew and upload command and coordinate data in real-time (non-blocking socket).
6. **Server Processing (C++)**
   - •Receive and parse incoming command data.
   - •Validate command structure and content.
   - •Calculate robotic arm trajectory using Inverse Kinematics (IK).
   - •Perform Motion Interpolation for smooth movement planning.
7. **Environment Safety Detection**
   - •Continuously read proximity sensor data via PLC.
   - •Analyze environment for obstacles or unsafe conditions.
   - •Predict potential collision based on current trajectory and environment mapping.
8. **Decision Making**
   - **•If Safe Condition Detected:**
     - •Approve and execute movement command.
     - •Send control signals to robotic arm via motion control library.
   - **•If Unsafe Condition Detected:**
     - •Trigger safety alarm.
     - •Immediately stop robotic arm movement (Emergency Stop).
     - •Notify user with warning signal (visual or audio).
9. **Control Robotic Arm**
   - •Send real-time commands to servo motors.
   - •Monitor arm position and movement feedback.
10. **System Feedback and Monitoring**
    - •Display system status on HMI (e.g., Running, Error, Stopped).
    - •Log all operations and safety events for record-keeping.

Academic

Automated Multi-Intersection Traffic Light Control System

Project Overview

This project focuses on designing an intelligent traffic light control system capable of adapting to real-time pedestrian and vehicle flow conditions across multiple intersections, specifically simulating typical traffic scenarios at intersections in Taiwan. The system leverages **Delta DVP48EH PLCs** and was programmed using **WPLSoft**, ensuring a highly reliable and industry-relevant solution for urban traffic management.

My Contributions

- Modeled typical Taiwanese urban intersections, incorporating pedestrian crossings, left-turn and right-turn lanes.
- Developed PLC control logic using **Delta DVP48EH Series PLC**.
- Implemented and debugged the program using **WPLSoft** (Delta's proprietary PLC programming environment).
- Integrated real-world timing logic based on Taiwan's traffic signal control standards.
- Installed directional and distance sensors to monitor pedestrian and vehicle flow.
- Simulated dynamic traffic patterns and validated signal timing adjustments under peak and off-peak conditions.

Key Features

Pedestrian Clearance Detection

Installed directional and distance sensors on pedestrian signal poles.

Detects pedestrian entry and exit during green light phases to and prevent premature signal changes.

Vehicle Flow Counting

Deployed vehicle flow sensors to count vehicles crossing the intersection during green lights.

Adjusted signal timing dynamically based on real-time vehicle counts.

Holiday and Peak Time Adjustment

Configured signal timing profiles based on weekday, weekend, holiday, and peak-hour schedules.

Automatically adapted green/red light durations for optimal flow during different periods.

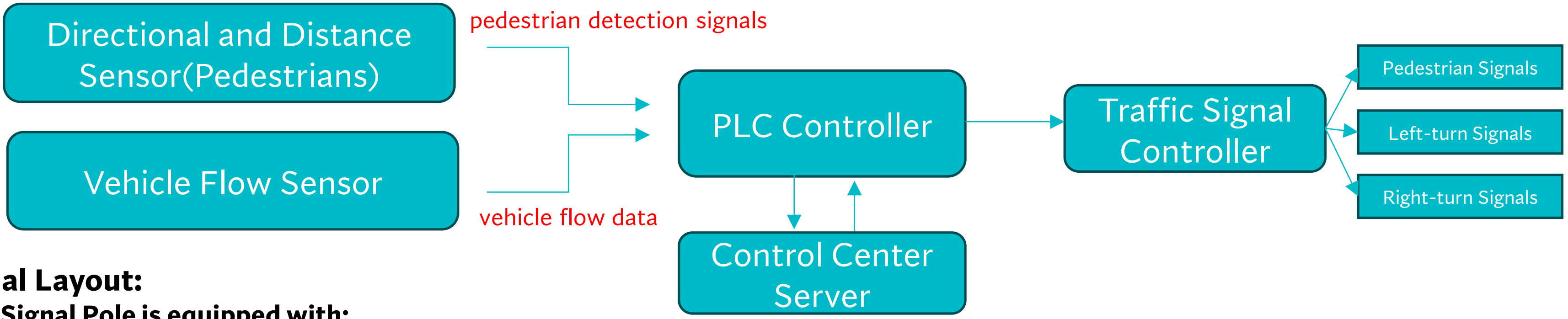
Control Center Notification

Sent real-time traffic flow data and system status updates to the control center via PLC communication interfaces.

Triggered alerts for abnormal traffic conditions such as high pedestrian density or vehicle congestion

Dedicated Turn Signals

Implemented independent left-turn and right-turn signal control to enhance traffic management flexibility and efficiency.



Displays real-time traffic conditions
Alerts abnormal traffic situations
Stores historical traffic flow data for analysis

Sensor and Signal Layout:

1. **Each Pedestrian Signal Pole is equipped with:**
- Directional Sensor (detects pedestrian movement direction)
- Distance Sensor (detects pedestrian presence)
- Camera

2. **Vehicle Flow Sensors:**
- Installed before zebra crossings to count passing vehicles.

3. Dedicated Turn Signals:
- Separate signals for left-turn and right-turn lanes.

Data Flow and Control:

1. **Pedestrian Detection:**
- Directional and distance sensors detect pedestrians entering or exiting the crosswalk.
- Data sent to PLC Controller to prevent premature signal change if a pedestrian is still crossing.

2. **Vehicle Flow Counting:**
- Vehicle sensors track the number of vehicles crossing during green light periods.
- PLC adjusts signal durations dynamically based on real-time vehicle flow.

3. **Timing Adjustment:**
- The PLC references a predefined schedule (weekday, weekend, holiday, peak-hour).
- Signal durations are automatically optimized based on real-time traffic data and schedule settings.

4. **Communication with Control Center:**
- PLC uploads real-time traffic and system status data to the control center.
- The control center receives alerts for anomalies and can issue override commands if necessary.

5. **Control Actions:**
- Dynamically update traffic light cycles.
- Activate/deactivate dedicated turn signals based on traffic density.

System Components Summary

Sensors

Directional and distance sensors for pedestrians.

Vehicle flow sensors (inductive loops and infrared counters).

Control Logic

PLC-based real-time control.

Combination of predefined schedules and adaptive control algorithms.

Networking

Communication between PLC and Control Center via wired or wireless (4G/5G) connections.

Traffic Signal Management

Independent control of main road signals, pedestrian crosswalk signals, and left/right-turn signals.

Academic

Power Grid Densification: Makara Beach Case Study

Project Overview

In response to the planned residential densification at Makara Beach, this project aimed to assess and enhance the existing low-voltage distribution network to accommodate increased load demand. Modeling and simulations were conducted using DIgSILENT PowerFactory to ensure that the network could reliably support the future load growth while maintaining compliance with voltage and thermal performance standards.

My Contributions

- Simulated both current and densified load scenarios to identify system constraints.
- Diagnosed critical network issues including transformer overloads, line congestion, and voltage instability.
- Proposed, modeled, and validated optimization strategies through iterative load flow simulations.
- Evaluated alternative configurations to ensure a balance between performance and cost.

Key Findings

Identified Issues

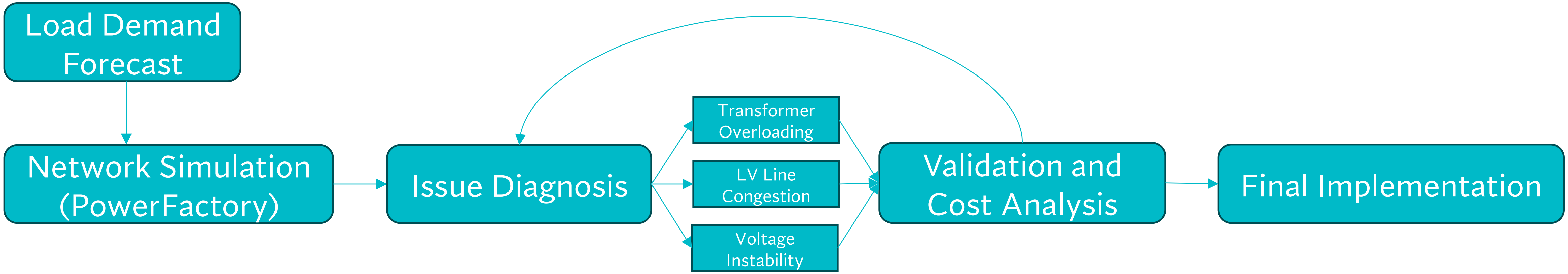
- Severe transformer overloading under full load conditions.
- Overloading of low-voltage distribution lines.
- Voltage drops beyond acceptable thresholds across multiple nodes.

Design Considerations

- **Thermal and Voltage Compliance**: Maintained operating conditions within thermal and voltage thresholds.
- **System Redundancy and Reliability**: Enhanced network robustness to support stable operation under varied load conditions.
- **Cost-Efficiency**: Selected solutions that deliver strong technical performance while minimizing capital expenditure.
- **Scalability**: Ensured the network layout remains simple and scalable for future residential or commercial expansions.

Solutions Implemented

- **Transformer Upgrades**
  Upgraded key transformers at strategic points in the network to accommodate the increased load and eliminate overloading risks.
- **Line Reinforcements**
  Paralleled and restructured low-voltage feeders to redistribute load, reduce congestion, and improve load balancing across the network.
- **Voltage Profile Optimization**
  Improved voltage profiles at all terminal nodes, ensuring that system voltages remained within regulated limits to enhance service quality and reduce losses.



Load Demand Forecast

Network Simulation (PowerFactory)

Issue Diagnosis

Transformer Overloading

LV Line Congestion

Voltage Instability

Validation and Cost Analysis

Final Implementation

Metric	Before	After
Max Transformer Loading	227.77%	<80%
Max LV Line Loading	>90%	<60%
Lowest Node Voltage	<0.96 p.u.	>0.975 p.u.
Total Upgrade Cost (NZD)		+10.7%

Remark:
All these improvements were achieved with only a **10.7% increase** in the original system cost — ensuring high performance at an optimized budget.

Academic

Educational Training Kit Redesign and Enhancement

Project Overview

This project focuses on the redesign and modernization of a microcontroller-based educational platform for university-level electronics courses. It aims to replace the outdated 8051-based system with a versatile and robust learning kit, featuring updated hardware, expanded modularity, and improved safety.

My Contributions

- Conducted system requirement analysis and translated them into design specifications.
- Selected and validated a modern microcontroller development board (STM32F411RE).
- Designed a high-precision 32-bit Digital-to-Analog Converter (DAC) module.
- Developed protective enclosures ensuring mechanical and electrical safety.
- Performed cost analysis to meet the budgetary constraints and maintain scalability.

Key Features

Microcontroller Upgrade

Replaced 8051 microcontroller with a modern ARM Cortex-M4-based development board, offering enhanced processing power, peripheral support, and industry-standard tools.

Peripheral Module Design

Designed a DAC module capable of high-resolution analog output, facilitating broader experimentation opportunities.

Protective Enclosure Engineering

Developed modular enclosures using commercial-grade ABS plastic to provide mechanical protection and ease of maintenance.

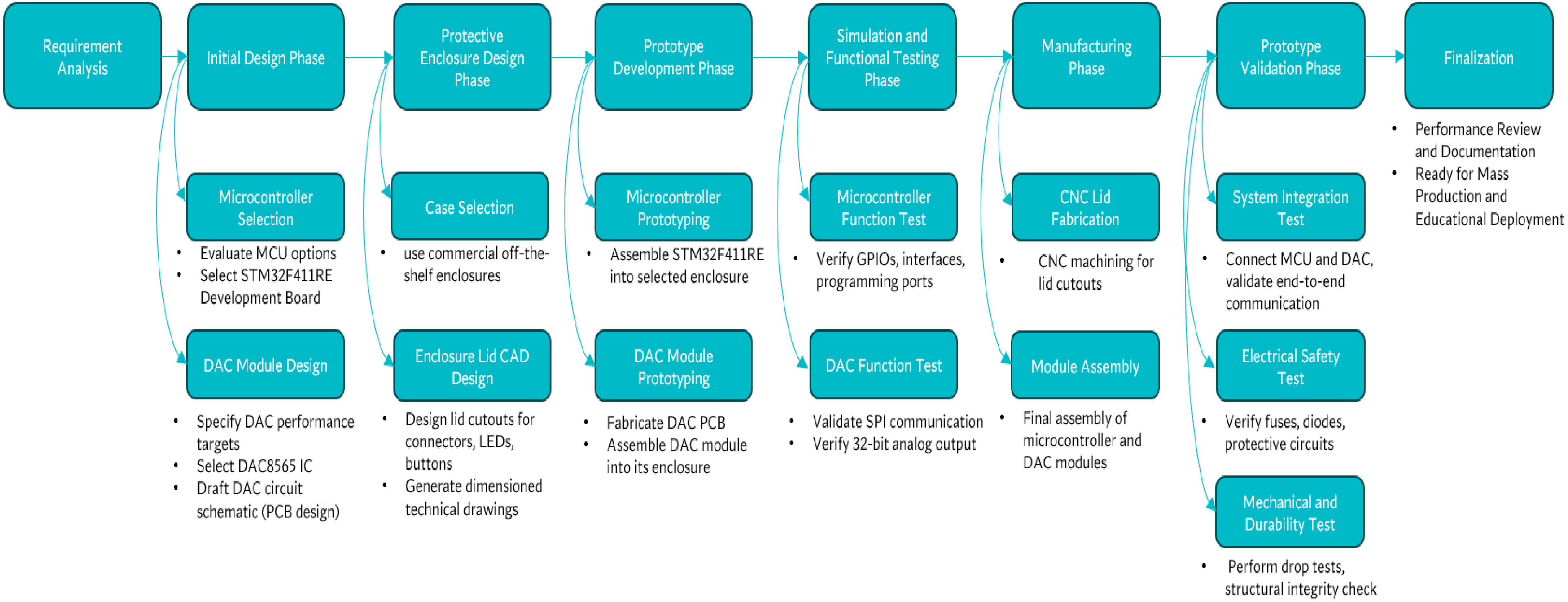
Safety and Reliability Enhancements

Incorporated fuses, diodes, and protective circuitry to ensure electrical safety and long-term durability.

Educational Versatility

Ensured compatibility with various auxiliary modules and industry-relevant development environments, supporting future curriculum expansion.

System Architecture Overview



Requirement Analysis

Initial Design Phase

Microcontroller Selection

- Evaluate MCU options
- Select STM32F411RE Development Board

DAC Module Design

- Specify DAC performance targets
- Select DAC8565 IC
- Draft DAC circuit schematic (PCB design)

Protective Enclosure Design Phase

Case Selection

- use commercial off-the-shelf enclosures

Enclosure Lid CAD Design

- Design lid cutouts for connectors, LEDs, buttons
- Generate dimensioned technical drawings

Prototype Development Phase

Microcontroller Prototyping

- Assemble STM32F411RE into selected enclosure

DAC Module Prototyping

- Fabricate DAC PCB
- Assemble DAC module into its enclosure

Simulation and Functional Testing Phase

Microcontroller Function Test

- Verify GPIOs, interfaces, programming ports

DAC Function Test

- Validate SPI communication
- Verify 32-bit analog output

Manufacturing Phase

CNC Lid Fabrication

- CNC machining for lid cutouts

Module Assembly

- Final assembly of microcontroller and DAC modules

Prototype Validation Phase

System Integration Test

- Connect MCU and DAC, validate end-to-end communication

Electrical Safety Test

- Verify fuses, diodes, protective circuits

Mechanical and Durability Test

- Perform drop tests, structural integrity check

Finalization

- Performance Review and Documentation
- Ready for Mass Production and Educational Deployment

Self Project

Big Data Multi-Platform Intelligent Integration System

Project Overview

Designed and implemented a fully integrated big data-driven network intelligence system for automated stock trading. The system legally aggregates data from multiple public platforms, applies self-developed classification and analysis algorithms, and executes algorithmic trades via banking and stock trading platform APIs. Special focus was placed on network security, system reliability, and real-time monitoring to ensure continuous, safe, and accurate trading operations.

My Contributions

- Developed a custom Python-based web crawler to legally collect financial and stock market data.
- Designed and implemented proprietary big data classification and analysis algorithms.
- Built an independent backtesting executable (EXE) program for historical trading strategy validation with customizable time frames.
- Integrated API connections with financial institutions and trading platforms for real-time automated transactions.
- Developed real-time notification mechanisms for remote system supervision and mechanical adjustment.
- Engineered robust system fault tolerance, considering potential network, hardware, and software failures, ensuring uninterrupted operation.
- Deployed the system for continuous operation with proven success over a year, including real-time report generation.

Key Features

Big Data Crawling and Analysis

- Automated collection of Taiwan stock market data, including technical indicators and trading volumes.
- Custom algorithms for data classification and analysis to identify target stocks efficiently and accurately.

Backtesting System (Custom EXE)

- Designed and implemented an independent backtesting tool.
- Supported historical data simulation with user-defined backtesting periods.
- Enabled pre-deployment strategy validation and optimization based on past market behavior.

System Integration

- Seamless API integration with banking and stock trading platforms for end-to-end trading automation.
- Real-time multitasking capability to aggregate and process data from multiple platforms simultaneously.

Network Security and Reliability

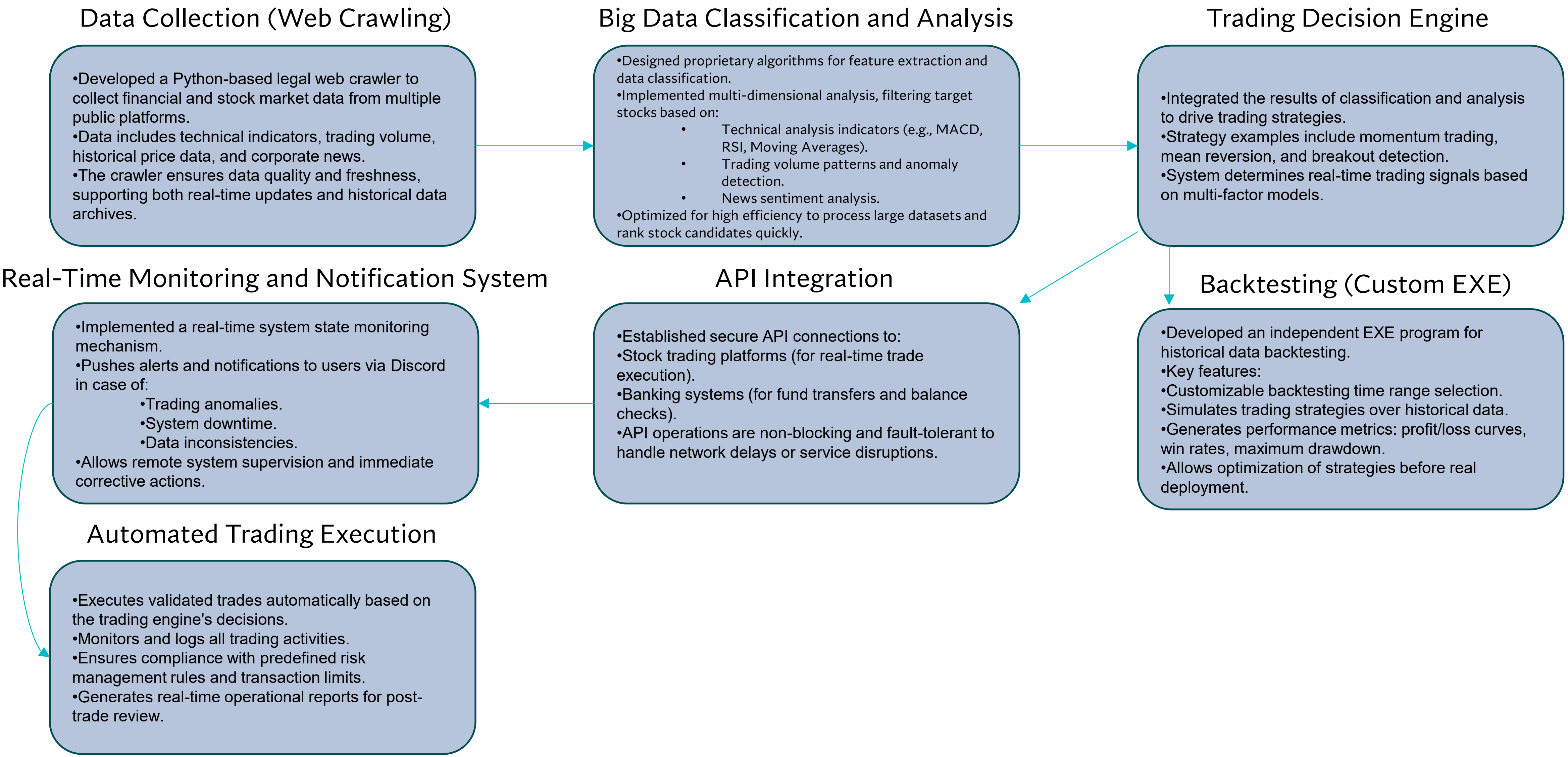
- Designed with security-first architecture to protect data transmission and system integrity.
- Implemented contingency handling for network failures, power outages, and data delays.

Real-Time Monitoring and Notifications

- Developed an internal notification mechanism for immediate alerts and remote adjustments.
- Enabled remote control and real-time status updates to ensure system flexibility and resilience.

Automated Trading Execution

- Executed trades based on predefined strategies with minimal human intervention.
- Generated real-time operational and performance reports for monitoring and review.



Data Collection (Web Crawling)

•Developed a Python-based legal web crawler to collect financial and stock market data from multiple public platforms.
•Data includes technical indicators, trading volume, historical price data, and corporate news.
•The crawler ensures data quality and freshness, supporting both real-time updates and historical data archives.

Big Data Classification and Analysis

•Designed proprietary algorithms for feature extraction and data classification.
•Implemented multi-dimensional analysis, filtering target stocks based on:

- Technical analysis indicators (e.g., MACD, RSI, Moving Averages).
- Trading volume patterns and anomaly detection.
- News sentiment analysis.

•Optimized for high efficiency to process large datasets and rank stock candidates quickly.

Trading Decision Engine

•Integrated the results of classification and analysis to drive trading strategies.
•Strategy examples include momentum trading, mean reversion, and breakout detection.
•System determines real-time trading signals based on multi-factor models.

Backtesting (Custom EXE)

•Developed an independent EXE program for historical data backtesting.
•Key features:
•Customizable backtesting time range selection.
•Simulates trading strategies over historical data.
•Generates performance metrics: profit/loss curves, win rates, maximum drawdown.
•Allows optimization of strategies before real deployment.

API Integration

•Established secure API connections to:
•Stock trading platforms (for real-time trade execution).
•Banking systems (for fund transfers and balance checks).
•API operations are non-blocking and fault-tolerant to handle network delays or service disruptions.

Real-Time Monitoring and Notification System

•Implemented a real-time system state monitoring mechanism.
•Pushes alerts and notifications to users via Discord in case of:
•Trading anomalies.
•System downtime.
•Data inconsistencies.
•Allows remote system supervision and immediate corrective actions.

Automated Trading Execution

•Executes validated trades automatically based on the trading engine's decisions.
•Monitors and logs all trading activities.
•Ensures compliance with predefined risk management rules and transaction limits.
•Generates real-time operational reports for post-trade review.