**Project Revision Meeting Minutes**

**Project**: Bluetooth Controlled Obstacle Avoidance Car  
**Team**: Team 5  
**Date**: 18 June 2025

**1. Technical Refactoring**

* **Language & Structure**
  + Current code is plain C without classes → will be rewritten in C++ with object-oriented design.
  + Discontinue use of Arduino; migrate all control to Raspberry Pi on Linux.
* **Non-compliant Technologies**
  + Remove Python/Flask web interface → replace with C++ terminal interface or basic UI.

**2. Real-Time System Improvements**

* Implement **event-driven architecture** using a state machine model.
* Add **RT latency measurement** and logging to evaluate responsiveness.

**3. Version Control & Build System**

* New Git repository link updated on team Wiki.
* Remove unclear file names (e.g., “test1”, “test\_final\_2”) → use clear, structured naming.
* Introduce **Makefile or CMake** to support consistent builds and compilation instructions.

**4. Testing & Validation**

* Add **unit tests** and **logs** for key modules.
* Create a **testing document** outlining procedures and expected results.

**5. Documentation & Presentation**

* Rewrite README: include project overview, component list, wiring diagrams, and build/run instructions.
* Add diagrams, hardware photos, and demo videos.
* Consider using **GitHub Pages** or other platforms for project presentation.

**Project Log – Day 2**

**Date**: 19 June 2025  
**Team**: Team 5  
**Project**: Bluetooth Controlled Obstacle Avoidance Car

**Hardware Refactoring & Improvements**

**Major Change**:  
 We have **completely stopped using Arduino**.  
 All hardware components are now **connected and controlled directly via Raspberry Pi** under a Linux environment, in compliance with course requirements.

**Specific Improvements**

1. **Power Supply Restructure**
   * Two **battery packs** are now used to power **two L298N motor driver modules**, which in turn control the four DC motors.
   * The **Raspberry Pi** is powered independently via a **portable power bank**, ensuring stable operation without Arduino involvement.
   * 桌子上放着笔记本电脑的人

     中度可信度描述已自动生成

**inner structure**

1. **Ultrasonic Sensors Upgrade**
   * Installed **three HC-SR04 ultrasonic sensors** (front, left, right) for multi-directional obstacle detection.
   * Sensors are wired and managed directly through Raspberry Pi GPIO.

|  |  |
| --- | --- |
| 吃东西的小孩  中度可信度描述已自动生成 | 图片包含 游戏机, 电缆, 电路  描述已自动生成 |
| **Reorganize Wiring** | **Camera Sensor Wiring** |

1. **GPIO Pin Optimization**
   * Encountered limitation: Raspberry Pi has only two 5V output pins and insufficient GND pins.
   * Solved by using **parallel wiring** to share power and ground across all modules.
2. **Camera Module Integration**
   * Successfully installed the **Raspberry Pi Camera Module V2.1 (8MP, 1080p)**.
   * Used a **CSI ribbon cable** for proper connection.
   * Resolved camera detection issue via raspi-config and proper cable alignment.

**Summary**

All hardware is now **Raspberry Pi–based**, with clean, centralized control. This marks a key step toward aligning our implementation with the required development environment and architecture.

**Project Log – Day 3**

**Date**: 20 June 2025  
**Team**: Team 5  
**Project**: Bluetooth Controlled Obstacle Avoidance Car

**Software Architecture Refactoring**

**Key Progress**:  
 We have successfully **refactored the entire project using object-oriented C++**, eliminating all C-style procedural code and Arduino dependencies.

**Implementation Highlights**

1. **Object-Oriented Structure**
   * All hardware components (motors, sensors, communication) are encapsulated into **independent C++ classes**, each managing its own memory and internal logic.
2. **Event-Driven Ultrasonic Sensing**
   * Developed a **fully event-driven obstacle detection system** using **callback interfaces**.
   * For precise and efficient edge detection, we adopted **libgpiod** for **blocking GPIO event handling**, avoiding CPU waste from polling.
3. **Real-Time Design Compliance**
   * The system now reacts to GPIO interrupts instantly, reducing latency and improving real-time behavior.
   * No active loops are used for sensor checking, aligning with event-based programming principles.
4. **Modular Build System**
   * Introduced a **standard Makefile**, which automatically detects and compiles all source files across modules.
   * The project directory follows a clean structure: src/, include/, build/, Makefile.

**Design Philosophy**

* **Modularity**: Each component is logically separated and reusable.
* **Scalability**: Easy to add or replace modules with minimal changes.
* **Readability**: Clear class interfaces and well-documented headers.
* **Compliance**: Fully meets embedded systems course requirements for encapsulation, real-time behavior, and build system support.

**Project Log – Day 4**

**Date**: 21 June 2025  
**Team**: Team 5  
**Project**: Bluetooth Controlled Obstacle Avoidance Car

**Hardware Debugging & Refinement**

**Motor Orientation Correction**

During obstacle avoidance testing, we observed irregular forward and turning behavior. Investigation revealed that some DC motors had reversed polarity due to incorrect wiring.  
**Action Taken**: Rewired affected motors to ensure all motors rotate in the intended direction for forward and backward movement.

**Directional Inaccuracy Issue**

**Problem Description**

The car showed imprecise turning behavior:

* Turning radius was inconsistent.
* Occasionally, the car failed to complete a turn or slipped in place.

**Root Causes**

* **Uneven Wheel Friction**: Variation in surface contact or tire wear caused inconsistent drag.
* **Power Imbalance Between Motors**: TT motors, although identical in specification, can differ slightly in performance.
* **Voltage Drop Across L298N Modules**: L298N introduces voltage loss (~1.5–2V), reducing effective motor torque.
* **Uneven Weight Distribution**: Battery and component placement caused off-center mass, increasing resistance on one side.

|  |  |
| --- | --- |
| 地上穿着球鞋的一双脚  描述已自动生成 | 人在系鞋带  描述已自动生成 |
| **Obstacle detected** | **Turn completed** |

**Resolution Strategy (Based on TT Motors + L298N Drivers)**

* **PWM-Based Differential Adjustment**  
  Dynamically adjust left/right motor speeds using PWM to compensate for imbalance.
* left\_motor.setSpeed(512);//0-1024
* right\_motor.setSpeed(512); //0-1024
* **Motor Calibration Routine**  
  Develop a simple speed test to run each motor independently and record RPMs, allowing for fine-tuning of individual PWM outputs.

**Project Log – Day 5**  
**Date:** 23 June 2025  
**Team:** Team 5  
**Project:** Bluetooth Controlled Obstacle Avoidance Car

**GUI-Based System Upgrade**

**Major Update:**  
Today, we upgraded the project from a command-line interface to a fully Qt-based graphical user interface (GUI), significantly improving user interaction and system visualization.

**Software Enhancements**

**Control Panel Implementation**  
A new ControlPanel class was developed with the following features:

* Directional control of the car via on-screen buttons.
* Real-time motor speed adjustment through sliders.
* Live video feed display from the onboard camera.
* Real-time distance readings from three ultrasonic sensors (front, left, right).

**Robot Class Refactoring**

* Added speed parameter support for finer movement control.
* Encapsulated sensor reading interfaces for modular use.

**Ultrasonic Class Improvements**

* Replaced traditional timing with std::chrono for high-precision distance measurement.

**Camera Class Optimization**

* Removed the original display window logic.
* Now provides a frame-read interface for integration with the Qt GUI.

**Build System Update**

* Updated the Makefile to support both the Qt GUI version and the command-line testing version.
* Ensured modular, maintainable, and extensible builds aligned with embedded system design principles.

**Summary**

This update marks a significant shift from basic control to a comprehensive system-level application. The project now features a robust GUI, modular architecture, real-time responsiveness, and full compliance with the requirements of the embedded systems course.

**Project Log – Day 6**  
**Date:** 24 June 2025  
**Team:** Team 5  
**Project:** Bluetooth Controlled Obstacle Avoidance Car

**Real-Time Responsiveness Optimization**

**Timer-Based Scheduling**

To ensure real-time responsiveness, we replaced continuous polling mechanisms with QTimer-based scheduling. This approach provides predictable and bounded timing behavior without relying on hardware interrupts.

* **Camera Frame Updates:** Set to a fixed interval of **50 ms** (20 FPS).
* **Ultrasonic Sensor Triggers:** Executed every **300 ms**, which remains below the threshold of user-perceived delay.

**Measured System Latency**

Using std::chrono high-resolution timestamps, we measured the actual delay from ultrasonic trigger to motion command. The observed reaction time is approximately **10–15 ms**, ensuring tight control loop timing.

**Real-Time Control Suitability**

Though our current system is **timer-driven** rather than **interrupt-based**, it still achieves **bounded and predictable delays** suitable for robotic control loops operating in the **10–100 Hz** range. This meets the real-time constraints of our project while retaining software simplicity and stability within the Qt framework.