ERNI Hackathon 2023  
Munich Voltage Vanguards

# Link to Repository

https://dev.azure.com/erniegh/ERNI-SmartFactory/\_git/Munich-Voltage-Vanguards?path=%2F&version=GBmain&\_a=contents

Goal

1. Build and program a robot which is an RC car with a camera that is able to move through an obstacle course as fast as possible.  
2. You are not allowed to control the robot from your office/hackathon location.  
3. Necessary control actions have to be done via the internet by someone else.  
4. Stop the time.  
5. Time starts when first part of the robot crosses start/finish line.  
6. Robot must pass all gates in given order without touching obstacles.  
7. Time ends when last part of the robot crosses start/finish line.  
8. Show your best result on video in your final presentation.

# UI Components

The UI requirements for the web application are as follows:

**Overview:** The goal is to create a web application accessible from both a browser or an app. The UI shall consist of various elements arranged in the upper part of the interface.

**Upper Part UI Elements:**

* **Connection State:** Displays the status of the connection between the application and the RC car.
* **Car State:** Indicates the current state of the RC car.
* **Switch with Label 'Driving Mode':** Allows users to switch between manual and automatic driving modes.
* **Voltage:** Shows the battery voltage (e.g., "X.X V") or displays a greyed-out field if the connection is still being established.
* **Stopwatch:** Shows the elapsed time during operation.
* **Live Video Feed:** Provides a live video stream from the RC car.
* **Start / Stop Button:** Initiates and stops the RC car's movement.
* **Reset Button:** Reset for the current driving mode and reset the stopwatch
* **Checkpoint Counter:** Displays the current checkpoint count (e.g., "0/9") during the automatic mode.

**Connection State:**

* When connecting, the UI should display "Connecting" to indicate that the connection is being established.
* If the connection is lost for more than 15 seconds, a timer should show when the connection was last active.
* When the connection is successfully established, display "Connected."
* If the Connection State is "Connecting," the Car State should show "Car status will be shown when the connection is established."
* When the Connection State is "Connected," the Car State should display "Ready."

**Voltage:**

* When the Connection State is "Connected," display the battery voltage as "X.X V."
* If the Connection State is "Connecting," show the voltage field without values on a grey background.

**Optional - Checkpoint Counter:**

* During automatic mode, the counter should increase after passing each checkpoint until it displays "9/9."
* A reset button allows the user to reset the counter to "0/9."

**Switch for Driving Mode:**

* Users can switch between manual and automatic driving modes using this control.

**Automatic Mode:**

* In automatic mode, the Start button is active when the connection to the car is established, and the car state is "Ready."
* Pressing the Start button initiates the automatic run.
* A Stop button can be pressed to halt the car's movement and abort the automatic run.
* After pressing Stop, the user receives a message to return the car to the start position and must confirm this by pressing the Reset button on the UI (which becomes active only after pressing Stop).
* Once a run is completed, the user is notified, and they are asked to place the car on the starting position and confirm this by pressing the Reset button (similar to stopping a run). The Reset button will also reset the stopwatch and make an reset from the car.

**Manual Mode:**

* In manual mode, controls for sending commands to the RC car (e.g., forward, backward, left, right) are displayed on the right-hand side of the screen. In addition, there is also a Start / Stop and Reset button, to Stop the driving time and reset the general car parameters including stopwach.

**Stopwatch**:

* The stopwatch shows the counted time for the run automatically. With Button “Start” unter the Autonomy driving mode, the time is running and counting. With Button “Stop” the time is stopped.
* Under the Manual Driving mode, the Start and Stop Buttons are also needed but coun

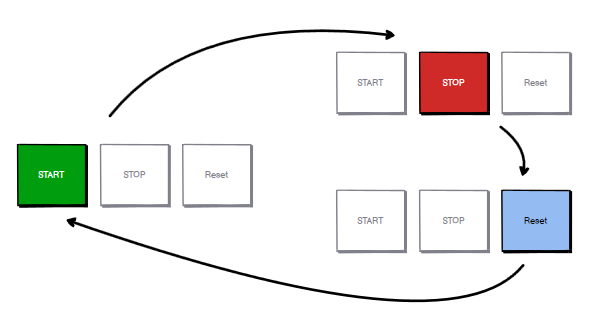
These UI requirements outline the key elements and behaviors of the web application for controlling an RC car.

## Requirements

### Mockup – Autonomy driving mode

Ein Bild, das Screenshot, Diagramm, Rechteck enthält.

Automatisch generierte Beschreibung



### Mockup – Manual driving mode - Default

Ein Bild, das Text, Software, Computersymbol, Zahl enthält.

Automatisch generierte Beschreibung

### **Functional Requirements (FR):**

* **FR1 - Connectivity**: The RC car must be able to connect to the internet using a Wi-Fi connection.
* **FR2 - Camera Integration**: The RC car must be equipped with a camera that can broadcast a live feed.
* **FR3 - Obstacle Course Navigation**: The robot must navigate the obstacle course without touching any obstacles.
* **FR4 - Remote Control**: The robot must be controllable remotely via the internet.
* **FR5 - Timing Mechanism**: A timing mechanism must start when the first part of the robot crosses the starting line and stop when the last part of the robot crosses the finish line.
* **FR6 - Checkpoints**: The robot must pass all checkpoints in the given order.
* **FR7 - Web Application**: The system shall have a web application with a UI that displays the connection state, car state, battery load or voltage, a checkpoint counter, and a stopwatch.
* **FR8 - Video Feed**: The web application must show a live video feed from the RC car's camera.
* **FR9 - User Controls**: The UI shall provide controls for manual movement (forward, backward, left, right) of the RC car.
* **FR10 - Mode Switching**: The system shall allow the user to switch between manual and automatic mode via the UI.
* **FR11 - Auto Mode Commands**: In automatic mode, the system shall allow starting, stopping, and resetting the robot's run. If stopped, the user should be prompted to return the car to the starting position.

### **Non-Functional Requirements (NFR):**

* **NFR1 - User Feedback (Visibility of system status)**: The UI shall provide real-time feedback on the connection and car state, battery load, and checkpoint counter.
* **NFR2 - Conformity with User Expectations**: The UI's interaction principles should match user expectations, ensuring it is consistent with user knowledge and accepted conventions.
* **NFR3 - Error Tolerance**: The system should tolerate minor user errors, guiding them to achieve the desired outcome.
* **NFR4 - Controllability**: Users should have the ability to control the speed and direction of the RC car efficiently via the UI.
* **NFR5 - Suitability for Learning**: The UI should be intuitive, ensuring that users can quickly understand and use the application without prior training.
* **NFR6 - Aesthetic Design**: The design of the UI should be minimalist, ensuring that only essential and relevant information is displayed.
* **NFR7 - Consistency and Standards**: All UI elements and controls must adhere to industry and platform standards to avoid user confusion.
* **NFR8 - Error Prevention and Recovery**: The system should minimize opportunities for errors. If errors occur, it should guide users with clear messages and recovery options.
* **NFR9 - Flexibility**: The UI should offer efficiency, catering to both novice and experienced users.
* **NFR10 - Help and Documentation**: The system should provide accessible help documentation or tooltips for users if needed.
* **NFR11 - Responsiveness**: The live video feed and control responses in the web application should have a latency of no more than 2 seconds.

### **Detailed UI Requirements:**

#### Connection State Block:

* **FR12**: The web application shall display a "Connection state" block in the upper part.
* **FR13**: When the RC car has a working connection to the web application, the "Connection state" block shall display the word “Connected”.
* **FR14**: While the connection to the RC car is being established, the "Connection state" block shall display the word “Connecting”.
* **FR15**: If the connection to the RC car is lost for more than 15 seconds, a timer shall be displayed in the "Connection state" block, indicating the last time the connection was active.

#### Car State Block:

* **FR16**: The web application shall display a "Car state" block adjacent to the "Connection state" block in the upper part.
* **FR17**: When the "Connection state" is “Connected”, the "Car state" block shall display the word “Ready”.
* **FR18**: If the "Connection state" is “Connecting”, the "Car state" block shall display “Car status will be shown when connection is established”.

#### Battery Load/Voltage Block:

* **FR19**: The web application shall display a "Battery load or voltage" block adjacent to the "Car state" block in the upper part.
* **FR20**: When the "Connection state" is “Connected”, the "Battery load or voltage" block shall display the voltage as “X.X V”.
* **FR21**: If the "Connection state" is “Connecting”, the "Battery load or voltage" block shall display a greyed-out voltage value.

#### Checkpoint Counter and Stopwatch Block:

* **FR22**: The web application shall display a "Checkpoint counter" block in the upper part, showing checkpoints passed as a fraction, e.g., “0/9”.
* **FR23**: When the RC car is in "Automatic mode" and passes a checkpoint, the "Checkpoint counter" shall increment by one.
* **FR24**: A reset button shall be provided near the "Checkpoint counter". On pressing this button, the "Checkpoint counter" shall be reset to “0/9”.
* **FR25**: The web application shall display a "Stopwatch" adjacent to the "Checkpoint counter" to indicate the elapsed time since the RC car started the obstacle course.

#### Video Feed:

* **FR26**: The web application shall prominently display a live video feed from the RC car below the aforementioned blocks on the left side of the screen.

#### User Controls:

* **FR27**: The web application shall display user controls on the right-hand side of the screen.
* **FR28**: In "Manual mode", the user controls shall provide buttons to send forward, backward, left, and right commands to the RC car.
* **FR29**: The web application shall provide a switch on the UI for users to change between "Manual" and "Automatic" mode.
* **FR30**: In "Automatic mode", the user controls shall provide "Start", "Stop", and "Reset" buttons.
* **FR31**: The "Start" button shall be active only when the car is connected and ready.
* **FR32**: Once "Start" is pressed, a "Stop" button shall be activated, allowing users to abort the RC car's automatic run.
* **FR33**: On pressing the "Stop" button, a message shall prompt users to return the car to the starting position, and a "Reset" button shall be activated.
* **FR34**: Upon completion of a run, a message shall prompt users to place the car on the starting position, and the "Reset" button shall be activated.

#### Robot Requirements:

* **FR35**: The robot, being an RC car, shall be equipped with a camera capable of streaming live video.
* **FR36**: The robot shall be able to navigate through the obstacle course autonomously at maximum achievable speed without hitting obstacles.
* **FR37**: The robot shall be equipped with sensors to detect the start/finish line and gates for accurate navigation and time calculation.

#### Remote Control:

* **FR38**: The web application shall be accessible from any location via the internet.
* **FR39**: All control actions for the robot, whether manual or automatic, shall be executable remotely over the internet.

#### Timing Mechanism:

* **FR40**: The web application shall have a built-in timer mechanism to track the robot's progress through the obstacle course.
* **FR41**: The timer shall start when the first part of the robot crosses the start line.
* **FR42**: The timer shall stop when the last part of the robot crosses the finish line.
* **FR43**: The elapsed time shall be displayed prominently on the web application, adjacent to the live video feed.

#### Navigation & Obstacle Course:

* **FR44**: The robot shall have a mechanism to detect and navigate through gates in the given order.
* **FR45**: The robot's sensors shall be capable of detecting nearby obstacles, preventing collisions.
* **FR46**: The web application shall provide alerts or notifications when the robot approaches an obstacle too closely, or if there is a potential risk of not passing through a gate correctly.

#### Presentation & Reporting:

* **FR47**: The web application shall allow for recording the robot's run, saving it as a video file for presentation purposes.
* **FR48**: The best run's video, along with the time taken, shall be easily accessible from the application for the final presentation.
* **FR49**: The application shall provide a leaderboard or results panel to showcase the best run times, allowing comparison between different attempts or even different users' controls.

#### Miscellaneous:

* **FR50**: The application shall provide real-time feedback regarding the robot's health, signal strength, and any technical issues.
* **FR51**: The web application should offer a test mode, allowing users to familiarize themselves with controls before attempting the main obstacle course.
* **FR52**: There should be an emergency stop or pause button on the application UI, allowing for immediate halting of the robot in case of unforeseen issues or challenges.

### **Extended Architecture Description**

#### 1. **System Components:**

* **Robot (RC Car)**:
* **Camera Module**: Provides live video streaming capability. Ideally, it will use a low latency system for real-time feedback.
* **Propulsion System**: Motors equipped with encoders to ensure accurate movement and turning, facilitating precise navigation.
* **Onboard Sensors**:
* Ultrasonic or infrared sensors for obstacle detection.
* RFID or magnetic sensors for gate detection.
* **Onboard Computer/Microcontroller**: Handles decision-making and interprets commands from the web server. A system like Raspberry Pi could be suitable for this purpose.
* **Battery Monitor**: Sends battery voltage to the server for UI display.
* **Web Server**:
* **Backend Framework**: A server framework like Node.js or Django to process incoming commands and send data back to the robot.
* **Video Streaming Component**: Utilizing services like WebRTC to relay real-time video feed with low latency.
* **Database**: For storing run logs, timestamps, and video data. Options include databases like MongoDB or PostgreSQL.
* **Web Application (UI)**:
* **Framework**: Built on frameworks like React or Angular for real-time UI updates.
* **Live Video Feed Viewer**: Embedded video player using protocols like HLS or MPEG-DASH.
* **Controls Interface**: Custom-built or library-based UI components for user interactivity.

#### 2. **Communication**:

* **Robot to Web Server**: Bi-directional communication. MQTT could be ideal because of its lightweight nature and real-time capabilities.
* **Web Server to Web Application**: RESTful API endpoints for non-real-time data and WebSockets for real-time data.

#### 3. **Data Storage**:

* Videos are stored in a cloud storage solution (like AWS S3) for scalability and ease of access.
* Metadata about runs is stored in the relational database linked with video URLs.

#### 4. **Security**:

* Secure the communication between components using protocols like TLS/SSL.
* The web application should have user authentication to prevent unauthorized access.

### **Extended User Manual**

#### **1. Setting Up:**

* **1.1**. Ensure the robot is fully charged. An LED indicator will show the charging status.
* **1.2**. Turn on the robot using the main switch located at its base.
* **1.3**. Connect the robot to a stable Wi-Fi connection using its onboard interface or the provided configuration tool.
* **1.4**. Access the web application using a supported web browser (e.g., Chrome, Firefox).

#### **2. Web Application Interface**:

* **2.1 Dashboard**:
* **Connection Info**: Provides details about robot-server connectivity, including latency.
* **Video Feed**: Ensure you have a stable internet connection for seamless streaming.
* **2.2 Control Panel**:
* Mode selectors, movement controls, and real-time data like battery status.
* **2.3 System Alerts**:
* Any critical alerts, such as battery low or lost connection, will be displayed prominently.

#### **3. Navigation & Controls**:

* **3.1 Manual Mode**:
* Use arrow icons or W (forward), A (left), S (backward), D (right) on the keyboard for movement.
* Movement is stopped by releasing the key or button.
* **3.2 Automatic Mode**:
* One-click "Start" initiates the automated obstacle course run.
* "Stop" will halt the robot immediately, triggering a prompt for the user.
* “Reset”

#### **4. Review & Analysis**:

* **4.1 Results Section**:
* All completed runs, with their timestamps, are listed.
* Click on a specific run to view detailed metrics and video replay.
* **4.2 Best Runs Gallery**:
* Highlighted section with top-performing runs. Videos can be played directly from this section.

#### **5. Troubleshooting & Support**:

* **5.1 Connectivity**:
* If you lose the connection to the robot, check its Wi-Fi status. If online, refresh the web application.
* **5.2 Movement Issues**:
* In case of erratic movement, recalibrate the robot using the "Calibration" option in the settings.
* **5.3 Support**:
* Contact technical support via the "Help" section for assistance or to report bugs.

## EN ISO 9241-110:2020 interaction dialog principles

**1. Suitability for the task**

The dialog principles are relevant to the task when it assists the user in performing the task efficiently and effectively.

**2. Suitability for learning**

The dialogue principles are self-describing when each step of the dialogue is immediately understandable through system feedback or explained to the user on demand.

**3. Suitability for individualization**

The dialogue principles are controllable as the user can initiate and control the direction and speed of the ergonomics of human system interaction part to the point where the goal is reached.

**4. Conformity with user expectations**

The dialogue principles or interaction principles match user expectations when it is consistent and consistent with user characteristics, such as knowledge of the task, education, experience, and generally accepted conventions. receive.

**5. Self descriptiveness**

The dialog principles or interaction principles are fault tolerant if, despite an obvious input error, it is possible to achieve the expected result without the user needing or taking very little ergonomics of human system interaction part.

**6. Controllability (if applicable, i.e. in desktop or web applications)**

The dialogue principles can be personalized and the user interfaces software can be modified to suit the needs of the interactive system task, personal preferences, and skills of the user.

**7.Error tolerance (if applicable, i.e. in systems accommodating free form user input)**

The dialogue principles are suitable for learning as they support and guide the user and interactive system.

## 10 usability heuristics by Jakob Nielsen

1. **Visibility of system status**

The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.

1. **Match between system and the real world**

The design should speak the users' language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.

1. **Use control and freedom**

Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process.

1. **Consistency and standards**

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions.

1. **Error prevention**

Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions, or check for them and present users with a confirmation option before they commit to the action.

1. **Recognition rather than recall**

Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design (e.g. field labels or menu items) should be visible or easily retrievable when needed.

1. **Flexibility and efficiency of use**

Shortcuts — hidden from novice users — may speed up the interaction for the expert user so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

1. **Aesthetic and minimalist design**

Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.  
  
This heuristic doesn't mean you have to use a flat design — it's about making sure you're keeping the content and visual design focused on the essentials. Ensure that the visual elements of the interface support the user's primary goals.

1. **Help users recognize, diagnose, and recover from errors**

Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.

1. **Help and documentation**

It’s best if the system doesn’t need any additional explanation. However, it may be necessary to provide documentation to help users understand how to complete their tasks.