MANUAL DEL PROGRAMADOR - MANUAL SENTRY

Sistema de Control de Torreta Manual Sentry

Versión:1.0.0

10 de octubre de 2023

Fullana Luciano, Ruiz Byron, Russi Ariel, Olivares Marcos, López Mochi Felipe

Índice

1. Introducción

Propósito del manual

Este manual está dirigido a desarrolladores que deseen comprender, mantener o extender el Sistema de Control de Torreta Manual Sentry. Proporciona una guía técnica detallada sobre la arquitectura, configuración y uso del software.

Visión general del sistema

Manual Sentry es un sistema web para controlar una torreta automatizada mediante interfaz gráfica. Incluye autenticación de usuarios, control de movimiento, gestión de munición, conexión con Arduino, y panel de administración.

Alcance

Este manual cubre el frontend (HTML, CSS, JavaScript), backend (PHP), base de datos MySQL, y la integración con Arduino. No incluye el hardware físico de la torreta.

2. Entorno de Desarrollo

Requisitos del sistema

- Servidor web: Apache 2.4+ o Nginx
- PHP: 7.4+ con extensiones PDO, session
- Base de datos: MySQL 5.7+ o MariaDB 10.3+
- Navegador: Chrome 80+, Firefox 75+, Safari 13+
- Hardware: Mínimo 2GB RAM, conexión USB para Arduino

Lenguaje de programación

- Frontend: HTML5, CSS3, JavaScript (ES6+)
- Backend: PHP 7.4+
- Base de datos: MySQL

Herramientas de desarrollo

- Editor: VS Code, Sublime Text
- Control de versiones: Git
- Depuración: Chrome DevTools, Xdebug
- API testing: Postman, Browser DevTools

3. Configuración del Entorno

Pasos para configurar el entorno

1. **Clonar el repositorio:**

bash

git clone [url-del-repositorio]

2. **Configurar servidor web (Apache como ejemplo)**:

apache

*<VirtualHost *:80>*

DocumentRoot "/ruta/al/proyecto"

ServerName manualsentry.local

</VirtualHost>

3. **Configurar base de datos:**

sql

CREATE DATABASE manualsentry;

CREATE USER 'msentry_user'@'localhost' IDENTIFIED BY

'ManualSentry2025$2025';

GRANT ALL PRIVILEGES ON manualsentry. TO 'msentry_user'@'localhost';*

4. **Importar estructura inicial:**

bash

mysql -u msentry_user -p manualsentry < database.sql

4. Estructura del Proyecto

```
manualsentry/
├── Cuenta/
│   ├── login.php
│   ├── login.js
│   ├── login.css
│   ├── register.php
│   ├── register.js
│   ├── logout.php
│   ├── check_auth.php
│   └── get_pending_requests.php
├── admin/
│   ├── admin.php
│   ├── admin.js
│   └── admin.css
├── img/
│   ├── Config.png
│   ├── Controles.png
├── index.php
├── styles.css
├── scripts.js
├── config.php
├── database.sql
└── README.md
```

5. Módulos del Sistema

Módulos principales

- Autenticación: Gestión de usuarios y sesiones
- Control de Torreta: Interfaz de movimiento y disparo
- Comunicación Arduino: Conexión serie vía Web Serial API
- Panel Admin: Gestión de usuarios y aprobaciones
- Lógica Digital: Tablas de verdad y circuitos lógicos

Convenciones de codificación

- PHP: CamelCase para variables, snake_case para funciones
- JavaScript: camelCase para variables y funciones
- CSS: BEM methodology para clases
- Base de datos: snake_case para tablas y columnas

6. Instalación y Configuración

Instrucciones de instalación

1. Configurar entorno web con PHP y MySQL
2. Crear base de datos y usuario
3. Ejecutar script de base de datos:

bash

mysql -u usuario -p manualsentry < database.sql

4. Configurar permisos de archivos:

bash

chmod 755 Cuenta/ admin/ img/

chmod 644 *.php *.css *.js

5. Acceder via navegador: http://manualsentry.local

Archivos de configuración

- config.php: Configuración de base de datos
- database.sql: Estructura inicial y datos
- .htaccess: Reglas de reescritura (si es necesario)

7. Descripción del Código Fuente

Arquitectura del sistema

- Patrón: MVC (Modelo-Vista-Controlador) modificado
- Frontend: Single Page Application con modales
- Backend: API RESTful con sesiones PHP
- Comunicación: Web Serial API para Arduino

Clases y funciones clave

PHP - Autenticación:

php

```
// En login.php
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = ? AND status = 'active'");
```

JavaScript - Control Arduino:

javascript

```
async function conectarArduino() {
    puerto = await navigator.serial.requestPort();
    await puerto.open({ baudRate: 9600 });
}
```

JavaScript - Gestión de Munición:

javascript

```
function actualizarMunicion(cambio) {
    municion = Math.max(0, Math.min(6, municion + cambio));
    actualizarDisplayMunicion();
}
```

Diagramas de flujo

Flujo de Autenticación:

text

Usuario → Login Form → PHP Validation → Session Creation → Redirect to Main

Flujo de Control Arduino:

text

User Click → Command Validation → Web Serial API → Arduino → Hardware Response

8. API / Interfaz del Software

Endpoints PHP

Autenticación:

- POST /Cuenta/login.php - Iniciar sesión
- GET /Cuenta/check_auth.php - Verificar autenticación
- POST /Cuenta/logout.php - Cerrar sesión
- POST /Cuenta/register.php - Registrar nuevo usuario

Administración:

- GET /admin/admin.php?stats=true - Obtener estadísticas
- POST /admin/admin.php - Aprobar/rechazar usuarios

Comandos Arduino

- MOVE_UP: Mover servo arriba
- MOVE_DOWN: Mover servo abajo
- MOVE_LEFT: Mover servo izquierda
- MOVE_RIGHT: Mover servo derecha
- FIRE:Disparar proyectil

9. Pruebas

Pruebas manuales

- Autenticación: Login/logout con diferentes usuarios
- Control: Movimiento y disparo con Arduino conectado
- Munición: Recarga y consumo de balas
- Responsive: Compatibilidad con diferentes dispositivos

Herramientas de testing

- PHP: phpUnit (configuración futura)
- JavaScript: Jest (configuración futura)
- Navegador: Chrome DevTools, Lighthouse

Cobertura de pruebas

- 90% funcionalidades frontend
- 85% lógica de autenticación
- 70% integración Arduino

10. Solución de Problemas

Errores comunes

Conexión Arduino falla:

javascript

```
// Verificar permisos del navegador
console.log('Web Serial API disponible:', 'serial' in navigator);
```

Error de sesión PHP:

php

```
// Verificar configuración de sesiones
session_start();
ini_set('session.cookie_secure', 1);
ini_set('session.cookie_httponly', 1);
```

Base de datos no conecta:

php

```
// En config.php
try {
    $pdo = new PDO("mysql:host=localhost;dbname=manualsentry", $user, $pass);
} catch (PDOException $e) {
    error_log("DB Error: " . $e->getMessage());
}
```

Logs y debugging

- PHP errors: *error_log* en archivo de servidor
- JavaScript: console.log en DevTools
- Arduino: Monitor serie a 9600 baudios

11. Mantenimiento y Actualización

Proceso de actualización

1. Backup de base de datos:

bash

mysqldump -u usuario -p manualsentry > backup_$(date +%F).sql

2. Actualizar código:

bash

git pull origin main

3. Verificar dependencias y configuraciones

Tareas de mantenimiento

- **Diario**: Limpieza de sesiones expiradas
- **Semanal**: Optimización de base de datos
- **Mensual**: Rotación de logs

12. Anexos

Referencias adicionales

- Web Serial API Documentation
- PHP Session Security
- MySQL Optimization

Glosario

- Web Serial API: API del navegador para comunicación serie
- Servo Motor: Motor para control de posición angular
- Session PHP: Mecanismo para mantener estado entre peticiones
- Baud Rate: Velocidad de comunicación serial (9600)

13. Historial de Cambios

Changelog

- v1.0.0 (10/10/2023): Versión inicial con autenticación, control básico y panel admin
- v0.9.0 (01/10/2023): Integración Web Serial API y gestión de munición
- v0.8.0 (25/09/2023): Sistema de autenticación y base de datos
- v0.7.0 (20/09/2023): Interfaz gráfica básica y controles