Practica: Prototipado Rápido con Protobject Framework

Objetivos del Capítulo:

Al finalizar este capítulo, serás capaz de:

- Describir la filosofía y el enfoque de Protobject Framework para el prototipado de aplicaciones web interactivas, físicas y distribuidas.
- Identificar las capacidades clave y características de Protobject Framework, incluyendo su arquitectura basada en web y el uso de dispositivos cotidianos.
- Explicar cómo Protobject Framework integra componentes interactivos con elementos físicos y hardware como Arduino/Lego Technic.
- Configurar y utilizar Protobject Framework con HTML, CSS y JavaScript, utilizando Glitch como entorno recomendado.
- Analizar ejemplos de aplicación y reconocer las ventajas y limitaciones de este enfoque.

Introducción: Simplificando el Prototipado Interactivo Distribuido

Mientras que el prototipado tradicional de sistemas interactivos ubicuos a menudo requiere hardware especializado (sensores, actuadores, microcontroladores específicos) y software complejo, emergen nuevos enfoques para hacerlo más accesible, rápido y flexible. Este capítulo presenta "Protobject Framework", un sistema diseñado para el prototipado rápido de sistemas de Internet de las Cosas (IoT) y Ciber-Físicos (CPS) utilizando tecnologías web estándar.

La filosofía central de Protobject Framework es aprovechar la tecnología que ya poseemos: utiliza dispositivos comunes como smartphones, tablets y PCs, transformándolos en nodos de una aplicación distribuida. Cada nodo ejecuta una página HTML que contiene componentes interactivos programados con JavaScript. Además, fomenta la integración de estos componentes digitales con materiales físicos simples e inertes (papel, cartón, etc.), permitiendo crear prototipos físico-digitales funcionales sin necesidad de equipamiento costoso o especializado. Al operar completamente dentro de un navegador web y recomendarse el uso de plataformas como Glitch para el desarrollo, Protobject Framework busca reducir significativamente el tiempo y el costo de desarrollo, haciendo el prototipado de IoT más sostenible e inclusivo.

Arquitectura y Capacidades Fundamentales de Protobject Framework

Protobject Framework se distingue por varias características clave:

1. **Arquitectura Basada en Web (HTML/CSS/JS):** Las aplicaciones se construyen como un conjunto de páginas HTML estándar. Cada página define una parte de la interfaz o un "dispositivo" virtual. La interacción y la lógica se programan completamente en JavaScript, y la apariencia se controla con CSS. El framework adopta un enfoque integrado, combinando funcionalidad, estructura y estilo sin separar estrictamente HTML, CSS y JavaScript, aunque se puede manipular el DOM dinámicamente.
2. **Configuración Centralizada (config.js):** Un archivo config.js define la estructura de la aplicación distribuida. Aquí se listan todas las páginas (identificadas por name y page) que componen el sistema, se designa una página como main (principal) y se configuran las opciones de depuración (debug) para cada una mediante la función Protobject.initialize([...]).
3. **Funcionamiento Navegador/Glitch:** No requiere instalación de software. Las páginas HTML se cargan en navegadores web en diferentes dispositivos. Glitch es una plataforma recomendada para alojar y desarrollar proyectos con Protobject Framework, facilitando el prototipado rápido, la colaboración y las actualizaciones en tiempo real. También se puede usar cualquier servicio de hosting web estático.
4. **Conexión entre Dispositivos:** La página main (principal) muestra un botón de conexión (icono + por defecto, personalizable vía CSS). Las páginas secundarias pueden conectarse a la principal escaneando un código QR o usando un enlace compartido, estableciendo la comunicación.
5. **Comunicación mediante Core API:** Los diferentes dispositivos/páginas se comunican enviando y recibiendo mensajes a través de la Protobject Core API.
    - **Envío:** Protobject.Core.send(mensaje).to("pagina_destino.html");
    - **Recepción:** Protobject.Core.onReceived(function(mensaje) { ... });
6. **Componentes Interactivos:** Proporciona una biblioteca de componentes JavaScript predefinidos (Protobject.Button, Protobject.Lamp, Protobject.NoiseSensor, etc.) que encapsulan funcionalidades de UI, sensores (usando APIs del navegador como cámara, micrófono, acelerómetro) y actuadores (pantalla, sonido, vibración). Estos se instancian (new

Protobject.Componente({...})) y configuran directamente en el script de cada página HTML.

7. **Reutilización de Dispositivos y Sensores/Actuadores del Navegador:** Aprovecha las capacidades nativas de los dispositivos (cámara, micrófono, acelerómetro, GPS, altavoz, vibración) accesibles a través de las APIs web estándar (Sensor API, Geolocation API, Vibration API, WebAudio API, Web Speech API, Media Capture and Streams API, etc.).
8. **Integración Física (Opcional):**
    - **Arduino:** Permite la comunicación con placas Arduino (ej. Leonardo) vía WebUSB para controlar hardware externo (servos, LEDs) o leer sensores específicos, usando el componente Protobject.Arduino.
    - **Lego Technic:** Ofrece integración con Lego Technic mediante microservos controlados por Arduino, utilizando una pieza 3D diseñada para conectar ambos.
9. **Modos de Desarrollo y Producción:** Se puede configurar (Protobject.setProduction(true/false)) para facilitar el desarrollo local (una sola instancia) o asegurar identificadores únicos en despliegues públicos (múltiples instancias).
10. **Depuración Distribuida:** Ofrece modos de depuración (local, remote, master) configurables en config.js para gestionar y centralizar los mensajes de consola (logs, errores, warnings) provenientes de los distintos dispositivos conectados, facilitando el troubleshooting en un entorno distribuido.

Catálogo de Componentes Disponibles

Protobject Framework ofrece una variedad de componentes que se instancian y controlan mediante JavaScript en las páginas HTML. Cada uno simula un elemento físico o utiliza una API del navegador:

- **Basados en Cámara:** Protobject.PresenceSensor (detecta cambios vs. imagen base), Protobject.LightSensor (mide brillo), Protobject.CameraMovement (detecta movimiento con flujo óptico), Protobject.ArUco (detecta marcadores ArUco), Protobject.HandSensor (detecta manos y gestos), Protobject.BodySensor (detecta cuerpos), Protobject.FaceSensor (detecta rostros y expresiones).

- **Basados en Micrófono:** Protobject.NoiseSensor (mide intensidad de ruido), Protobject.VoiceRecognition (convierte voz a texto), Protobject.AudioClassifier (clasifica sonidos).
- **Basados en Altavoz:** Protobject.SoundPlayer (reproduce audio), Protobject.NotePlayer (reproduce notas musicales), Protobject.TextToSpeech (convierte texto a voz).
- **Sensores de Movimiento/Orientación:** Protobject.Acceleration (aceleración lineal sin gravedad), Protobject.Inclination (inclinación/ángulo respecto al suelo, con gravedad), Protobject.Orientation (orientación 3D, fusión de sensores).
- **Componentes Interactivos (UI):** Protobject.Lamp (elemento visual de lámpara), Protobject.Switch (interruptor on/off), Protobject.Button (botón táctil), Protobject.Knob (potenciómetro/perilla virtual), Protobject.Text (campo de texto editable o no).
- **Misceláneos:** Protobject.Haptic (control de vibración), Protobject.GPS (datos de geolocalización), Protobject.Arduino (interfaz con Arduino vía WebUSB).

(Nota: La documentación oficial de Protobject Framework contiene detalles y ejemplos de uso para cada componente).

Aplicaciones y Casos de Estudio

Hay varios casos de estudio que ilustran la aplicación de Protobject Framework (implementados usando su metodología basada en HTML/JavaScript) en dos contextos principales:

1. **Sistemas Inteligentes (Smart Systems):**
    - **Dispositivo de Postura:** Un smartphone (simulando un wearable) usa el componente Protobject.Acceleration para detectar mala postura. Envía un mensaje a otro componente (en el mismo o diferente dispositivo) Protobject.Haptic (vibración) o Protobject.Lamp (luz) para alertar al usuario. Una interfaz de control (ej. en otro smartphone/página) con Protobject.Knob y Protobject.Switch permite ajustar la sensibilidad o pausar el sistema.
    - **Indicador de Ruido para Aulas:** Una página HTML en un dispositivo (tablet/PC) usa Protobject.NoiseSensor para monitorear el nivel de ruido. Si supera un umbral, envía un mensaje a otra página (ej. proyectada en pantalla) que usa Protobject.Text y Protobject.Lamp (rojo) para mostrar una alerta visual, incentivando el silencio.

- **Sistema de Alarma:** Una página con Protobject.CameraMovement detecta movimiento. Envía un mensaje para iniciar una cuenta regresiva mostrada con Protobject.Text en otra página. Esta segunda página permite ingresar un código usando Protobject.Text (editable) o Protobject.Button. Si no se ingresa el código a tiempo, se envía un mensaje a una tercera página (o la misma) que activa un sonido de alarma con Protobject.SoundPlayer.
- **Cerradura Inteligente:** Una página actúa como teclado usando componentes Protobject.Button. Al ingresar la combinación correcta, envía un mensaje a una página conectada a un Arduino (usando Protobject.Arduino). El script en esta página controla un servo conectado a un mecanismo de Lego Technic para abrir/cerrar la cerradura. Permite establecer y resetear códigos.
- **Sistema de Iluminación Inteligente:** Páginas con Protobject.PresenceSensor detectan presencia en diferentes zonas. Envían mensajes a una página conectada a un Arduino (Protobject.Arduino) que controla servos y Lego Technic para accionar interruptores físicos de luces, encendiendo las zonas ocupadas y apagando las vacías.

2. **Visualizaciones de Información Físico-Digitales (InfoVis):** Estos prototipos integran visualizaciones creadas con librerías JavaScript como Plotly.js dentro de las páginas HTML, haciéndolas reaccionar a interacciones físicas detectadas por los componentes de Protobject Framework.
    - **Visualización de Consumo Energético:** Una página usa Protobject.FaceSensor para contar personas. Envía el número a otra página que muestra un gráfico Plotly.js, actualizándolo para reflejar el consumo energético estimado según las personas detectadas.
    - **Regulación de Temperatura:** Un dispositivo con Protobject.Orientation detecta el ángulo de apertura de una puerta física. Envía el ángulo a una página con un gráfico Plotly.js que simula y visualiza la caída de temperatura resultante en tiempo real.
    - **Experiencia Sísmica:** Un modelo físico de casa se mueve manualmente. Una página asociada usa Protobject.Acceleration para medir la intensidad del movimiento y la compara con datos históricos para encontrar el sismo equivalente, mostrándolo en un mapa (Plotly.js). Alternativamente, al hacer clic en un sismo en el mapa, se envía la intensidad a una página

con Protobject.Arduino que mueve el modelo físico de la casa (con servo/Lego) simulando el temblor.

- **Filtrado de Datos por Movimiento Corporal:** Una página usa Protobject.BodySensor para detectar la distancia y posición horizontal del usuario respecto a la pantalla. Esta información se usa para controlar dinámicamente el nivel de zoom y el punto temporal de una visualización de datos (Plotly.js) mostrada en la misma página.
- **Predicciones Precio Bitcoin:** Una visualización Plotly.js muestra predicciones (optimista, neutral, pesimista). La predicción mostrada cambia según la interacción detectada por diferentes componentes en páginas separadas: Protobject.FaceSensor (detectando expresión facial), Protobject.HandSensor (interpretando gestos), o Protobject.Inclination (leyendo la inclinación de un objeto físico asociado a cada predicción).
- **Control de Visualización de Exportaciones:** Una página usa Protobject.ArUco para detectar marcadores asociados a productos físicos (judías, té, atún). La posición o presencia/ausencia de estos objetos físicos envía mensajes que actualizan una visualización Plotly.js mostrando los datos de exportación correspondientes.
- **Interacción con Mapa Físico:** Un usuario toca un país en un mapa físico. Una página usa Protobject.HandSensor para detectar la posición del dedo índice. Identifica el país y envía un mensaje a otra página que usa Protobject.TextToSpeech para verbalizar la cantidad de exportación de fruta de ese país.

Ventajas y Limitaciones

Ventajas:

- **Accesibilidad y Bajo Costo:** Utiliza hardware existente y tecnologías web estándar.
- **Rapidez y Colaboración:** El enfoque web y el uso recomendado de Glitch aceleran el prototipado y facilitan el trabajo en equipo.
- **Flexibilidad:** Usa el ecosistema web (HTML/CSS/JS) y permite integrar hardware externo (Arduino/Lego).
- **Sostenibilidad:** Promueve la reutilización de dispositivos, reduciendo el desecho electrónico.

- **Depuración Centralizada:** Facilita el seguimiento de errores en sistemas distribuidos.

Limitaciones:

- **Precisión de Sensores:** Limitada por las capacidades de los sensores del dispositivo y las APIs del navegador (ej. ruido con micrófono, luz con cámara no son muy precisos).
- **Foco en Baja Fidelidad:** Ideal para etapas tempranas de exploración conceptual; no está diseñado para prototipos de alta precisión o fiabilidad final. La reconfigurabilidad es clave.
- **Eficiencia y Miniaturización:** Usar un smartphone como sensor/actuador es menos eficiente y más grande que soluciones embebidas dedicadas.
- **Curva de Aprendizaje Web:** Requiere conocimientos sólidos de desarrollo web (JS, manipulación del DOM, CSS) para personalizaciones avanzadas o integración con librerías complejas, lo cual puede ser un desafío para principiantes.
- **Exclusivo JavaScript:** A diferencia de lo mencionado en el abstract del paper para "Protobject Code", el "Protobject Framework" documentado funciona solo con JavaScript.

Conclusión del Capítulo

Protobject Framework ofrece una vía ágil y accesible para el prototipado de aplicaciones interactivas distribuidas y sistemas ciber-físicos, apoyándose completamente en el ecosistema de tecnologías web (HTML, CSS, JavaScript) y el hardware que ya poseemos. Su estructura basada en páginas conectadas, su API de comunicación simple y la recomendación de plataformas como Glitch lo convierten en una herramienta interesante para explorar rápidamente ideas de interacción multi-dispositivo y físico-digital en etapas tempranas del diseño, especialmente útil en contextos educativos o de innovación rápida donde la accesibilidad y la velocidad son prioritarias. Es crucial, sin embargo, entender sus limitaciones en cuanto a precisión y fidelidad para aplicarlo adecuadamente dentro del ciclo de diseño.