ROBOTSCHOOL · LÍNEA ARES

Documento Maestro Compilado

El concepto completo en un solo documento · 18 documentos · Junio 2026

> Este documento reúne **los 18 documentos del marco** de la Línea Ares en un solo archivo, ordenados por fases, para revisión y edición. Los **15 libros** (uno por grado + banda micro:bit) y sus PDF son archivos aparte; al final se listan.

Contenido

Línea Ares — Hilo Conceptual Transversal

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Este documento define la **columna vertebral** de la línea Ares: lo que permanece constante mientras cambian la placa, el lenguaje y la edad del estudiante. Todo el mapa curricular y el análisis de proyectos por nivel se cuelgan de aquí. Es la fuente única; los formatos Word, PPTX y la landing se generan a partir de este texto.

---

1. Propósito y principio rector

Ares es la línea editorial **abierta e interdisciplinar** de ROBOTSchool (complementaria a la línea Ecua, cerrada y guiada). Su promesa pedagógica es:

> **El estudiante no cambia de tema cada tres años; cambia de herramienta. La lógica, el pensamiento computacional y los hábitos de diseño se acumulan; la placa solo aporta más potencia.**

Este principio resuelve el riesgo de fragmentación que amenaza a cualquier currículo que rota de plataforma (Makey Makey → Arduino → Raspberry Pi Pico → ESP32). La continuidad no la da el hardware: la dan **el pensamiento computacional, las estructuras de programación y los tres ejes transversales**, que son los tres pilares de este documento.

---

2. Los cuatro niveles de un vistazo

Nivel	Nombre	Grados	Tecnología base	Lenguaje / entorno	Relación del estudiante con la tecnología
1	**Exploradores**	Transición · 1° · 2°	Makey Makey + mecánica y mecanismos	Bloques (Scratch / mBlock)	La descubre: causa–efecto y secuencias
2	**Constructores**	3° · 4° · 5°	Arduino + mecánica	Bloques (mBlock) → asomada a C++	La construye: arma mecanismos y los programa
3	**Inventores**	6° · 7° · 8°	Raspberry Pi Pico	MicroPython	La hace autónoma: sensores, datos y decisiones
4	**Innovadores**	9° · 10° · 11°	ESP32	Python / C++ + stack web (HTML, CSS, JS, Bootstrap, SQL)	La conecta y la crea: domótica, IoT y sistemas

El nombre de cada nivel **describe el verbo dominante** del estudiante en esa etapa: explorar → construir → inventar → innovar. Esa escalada de autonomía es, en sí misma, el relato de la línea.

---

3. Pilar A — Progresión del pensamiento computacional

El pensamiento computacional (PC) es la habilidad de fondo. Usamos las cuatro competencias clásicas más la depuración, y mostramos cómo **maduran** —no aparecen y desaparecen— a lo largo de los cuatro niveles.

3.1 Las competencias

- **Descomposición:** partir un problema grande en partes manejables.
- **Reconocimiento de patrones:** detectar regularidades y repeticiones.
- **Abstracción:** quedarse con lo esencial e ignorar el detalle irrelevante.
- **Diseño de algoritmos:** definir pasos ordenados que resuelven el problema.

- **Evaluación y depuración:** probar, encontrar el error y corregirlo (eje metacognitivo que cruza a todas las demás).

3.2 Cómo escala cada competencia por nivel

Competencia	Exploradores	Constructores	Inventores	Innovadores
Descomposición	Separar una acción en pasos simples (“para que suene, toco aquí”)	Dividir un mecanismo en partes y un programa en bloques	Separar un sistema en módulos (sensar / decidir / actuar)	Arquitectura de un sistema: dispositivo, red, datos, interfaz
Patrones	Reconocer que una acción repetida da el mismo efecto	Identificar repeticiones → usar bucles	Reutilizar funciones y librerías	Reconocer patrones de diseño y de datos en sistemas conectados
Abstracción	Símbolos = acciones	Variables como “cajas” que guardan valores	Funciones con parámetros; modelos de datos simples	Clases/objetos, APIs, esquemas de base de datos
Algoritmos	Secuencias y eventos (“cuando…”)	Bucles y condicionales	Algoritmos con variables, lógica y sensores	Algoritmos concurrentes, manejo de eventos en red, consultas
Depuración	“No pasó lo que esperaba” → reintentar	Probar y ajustar por bloques	Aislar el módulo que falla	Diagnóstico de sistema: hardware, código, red y datos

> **Regla de oro para los autores de proyecto:** un proyecto de un nivel superior nunca puede *omitir* una competencia que ya se trabajó abajo; debe ejercerla a mayor profundidad. Así se garantiza la acumulación.

4. Pilar B — Estructuras de programación que persisten

Este es el corazón del argumento “se cambia la placa, no la lógica”. Definimos la **secuencia canónica de estructuras** y la mapeamos a cada entorno. El estudiante reconoce la misma estructura aunque la sintaxis cambie.

4.1 Secuencia canónica (orden de introducción)

1. **Secuencia** (instrucciones en orden)
2. **Eventos** (“cuando ocurra X, haz Y”)
3. **Bucles** (repetición: definida e indefinida)
4. **Condicionales** (decisiones: si / si no)
5. **Variables y operadores** (guardar y operar datos)
6. **Funciones** / **abstracción** (empaquetar y reutilizar lógica)
7. **Estructuras de datos** (listas, diccionarios)
8. **Concurrencia y conectividad** (varias cosas a la vez; comunicación entre dispositivos)
9. **Manejo de datos y persistencia** (bases de datos, dashboards)

4.2 Mapeo estructura ↔ entorno por nivel

Estructura	Exploradores (bloques)	Constructores (bloques → C++)	Inventores (MicroPython)	Innovadores (Python/C++ + web)
Secuencia	Base			
Eventos	(“al presionar”)		(interrupciones simples)	(eventos de red)
Bucles	Introducción visual	`for` / `while`		
Condicionales	Introducción visual	`if/else`		
Variables/ operadores	Concepto de “caja”	tipado básico en C++		
Funciones	—	Asomada	con parámetros	+ módulos/clases
Estructuras de datos	—	—	listas/ diccionarios	+ modelos y SQL
Concurrencia/ conectividad	—	—	Introducción (sensores múltiples)	BLE/WiFi, IoT
Datos/ persistencia	—	—	Registro simple de datos	SQL + dashboard

> **Nota de honestidad pedagógica:** C++ en Constructores (3°–5°, 8–10 años) debe entrar como **“asomada”**, no como objetivo de dominio. La meta del nivel es consolidar bucles, condicionales y variables *en bloques*; ver C++ sirve para que el salto a texto en Inventores (MicroPython) no sea abrupto. Forzar sintaxis de C++ a esa edad frustra más de lo que enseña.

> **El puente bloques → texto (BIPES):** el paso de mBlock (bloques) a MicroPython (texto) en 6º se hace con **BIPES**, una plataforma web y gratuita que programa MicroPython **por bloques y muestra el código Python real** que generan. El estudiante sigue armando en bloques mientras *lee* el texto equivalente; cuando está listo, escribe directamente en

> Thonny. Así la continuidad del hilo no se rompe en el cambio de lenguaje. (Secuencia de adopción en `03…§6`.)

---

5. Pilar C — Los tres ejes transversales

Estos tres ejes cruzan **los cuatro niveles** y son la marca diferencial de Ares frente a un currículo de "solo robótica". Cada proyecto debe declarar explícitamente cómo aborda los tres.

5.1 IA bien usada

No es "usar IA por usarla", sino formar criterio. Progresión:

- **Exploradores:** la IA como "ayudante" que reconoce (voz, imágenes) — uso guiado y conversación sobre qué puede y qué no puede hacer una máquina.
- **Constructores:** clasificadores simples (p. ej. reconocimiento de imágenes/sonidos con herramientas no-code) integrados a un mecanismo; introducción a la idea de "entrenar con ejemplos".
- **Inventores:** uso de modelos para sensar/clasificar datos del proyecto; conversación sobre sesgos y datos; uso responsable de asistentes de código.
- **Innovadores:** integración de servicios/modelos de IA en proyectos IoT; ética de datos, privacidad y criterio sobre cuándo *no* automatizar.

> **Criterio transversal de "IA bien usada":** en todos los niveles se evalúa que el estudiante distinga entre *delegar el pensamiento* (malo) y *usar la IA como herramienta para amplificar el suyo* (bueno).

5.2 Diseño

El diseño da forma, función y comunicación al proyecto. Progresión:

- **Exploradores:** diseño físico y estético del montaje; bocetos.
- **Constructores:** diseño de mecanismos (relación forma-función); fabricación con corte láser/impresión 3D guiada.
- **Inventores:** diseño centrado en el usuario; prototipado iterativo; modelado 3D propio.
- **Innovadores:** diseño de interfaz (UI/UX) para dashboards y apps; diseño de sistema completo.

5.3 Robotización

La robótica como integración de mecánica, electrónica y programación. Progresión:

- **Exploradores:** mecanismos básicos y circuitos simples (motores, cables, portapilas).
- **Constructores:** automatización de mecanismos con Arduino (entradas → lógica → salidas).
- **Inventores:** sistemas autónomos que sensan, deciden y actúan (Pico + MicroPython).
- **Innovadores:** robots/dispositivos conectados; domótica e IoT (ESP32 + red).

---

6. Marco de extracción STEAM e interdisciplinariedad

Ares **no fuerza** que cada proyecto toque todas las asignaturas. Parte de un hecho: un proyecto tecnológico ya contiene de forma natural matemáticas, física y biología, y con frecuencia otras áreas. El trabajo editorial es **analizar cada proyecto y extraer** esas conexiones, documentándolas con rigor.

6.1 Plantilla de análisis por proyecto

Cada proyecto del mapa curricular se documenta con esta ficha:

1. **Nombre y nivel del proyecto.**
2. **Reto** / **pregunta guía** (anclada a un ODS).
3. **Estructuras de programación** que se ejercen (ver §4).
4. **Competencias de pensamiento computacional** en juego (ver §3).
5. **Ejes transversales:** cómo aparece IA, diseño y robotización (ver §5).
6. **Conexiones STEAM auténticas:** los conceptos de Ciencia, Tecnología, Ingeniería, Arte y Matemáticas que el proyecto *realmente* usa (no de relleno).
7. **Otras asignaturas detectadas** (lengua, sociales, ética…) — solo si aparecen de forma genuina.
8. **Estándar ISTE** principal y secundarios.
9. **Producto** / **evidencia** de aprendizaje y criterio de evaluación.

> **Principio anti-relleno:** una conexión solo se documenta si el estudiante *necesita* ese concepto para avanzar en el proyecto. Si hay que inventar una excusa para “meter” una asignatura, no va. La amplitud se cubre **a lo largo del nivel**, no dentro de cada proyecto.

6.2 Anclajes de marco

- **ODS (Objetivos de Desarrollo Sostenible):** dan el paraguas temático y la pertinencia social de cada reto. Permiten que las conexiones a ciencias sociales, ética y ciudadanía surjan de forma natural.
- **Estándares ISTE para Estudiantes** (siete roles): Aprendiz Empoderado, Ciudadano Digital, Constructor de Conocimiento, Diseñador Innovador, Pensador Computacional, Comunicador Creativo y Colaborador Global. Ares trabaja sobre todo **Pensador Computacional** (1.5), **Diseñador Innovador** (1.4) y **Comunicador Creativo** (1.6), con **Ciudadano Digital** (1.2) como hogar natural del eje “IA bien usada”.
- **Aprendizaje Basado en Proyectos (ABP):** estructura metodológica de cada unidad (reto → investigación → prototipo → producto → socialización).
- **Enfoque STEAM:** la integración disciplinar, con la “A” (Arte/Diseño) como eje propio en Ares.

---

7. Decisiones abiertas (para resolver en el mapa curricular)

Estas preguntas quedan registradas para no perderlas al detallar cada nivel:

1. **Evaluación:** ¿rúbricas por competencia de PC, por proyecto, o ambas? Recomendación: una rúbrica transversal de PC + criterios específicos por proyecto.
2. **Formación docente:** ¿qué ruta de capacitación necesita un docente para pasar de Arduino (lo que hoy domina el equipo) a MicroPython/ESP32? Es el cuello de botella real de Innovadores.
3. **Costo y logística de kits** por nivel y su relación con la capacidad de fabricación propia (corte láser, 3D, importación de electrónicos).
4. **Profundidad vs. amplitud en Innovadores:** definir el núcleo obligatorio (IoT con stack mínimo) y lo electivo, para no sobrecargar la banda superior.

Fin del documento maestro v0.1. Próximo hito: mapa curricular detallado por nivel, construido sobre este hilo.

Línea Ares — Hilos de Robotización y Diseño + Modelo de Convergencia

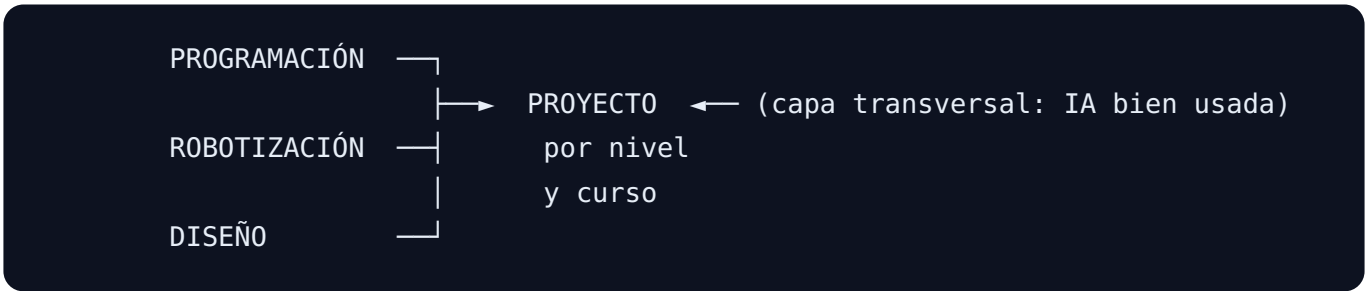
ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Documento complementario a `01_hilo_conceptual_ares.md`. Allí quedó definido el **hilo de programación** (pensamiento computacional + estructuras). Aquí se definen los otros dos hilos —**robotización** y **diseño**— con la misma lógica de maduración, y se establece el **modelo de convergencia**: la forma en que los tres hilos se trenzan dentro de cada proyecto, por nivel y por curso. El **buen uso de la IA** se trata como **capa transversal** que se monta sobre los tres hilos (ver §3 y `01…§5.1`).

---

0. El tejido de Ares en una imagen

Ares no enseña tres materias en paralelo. Enseña **un proyecto** en el que tres hilos se trenzan:



Cada hilo tiene su propia progresión (madura de Exploradores a Innovadores). Pero el aprendizaje **ocurre en el punto donde se encuentran**: la tarea del proyecto que sería imposible sin los tres a la vez. Definir ese **punto de convergencia** en cada proyecto es el verdadero trabajo editorial (ver §4 y §5).

---

1. Hilo conceptual de Robotización

La robotización es la integración de **energía, mecánica, electrónica y control**. Definimos seis **sub-hilos** que persisten en los cuatro niveles; lo que cambia es la profundidad, no el concepto. (Funcionan igual que los sub-hilos del diseño: cada uno madura por separado, pero se ejercen juntos en el proyecto.)

1.1 Sub-hilos persistentes

- **Energía y circuitos:** cómo se alimenta y circula la corriente.
- **Estructuras y mecanismos:** cómo se sostiene y se mueve.
- **Sensado (percepción):** cómo el sistema “siente” el entorno.
- **Actuación:** cómo el sistema “actúa” sobre el entorno.
- **Control y automatización:** el lazo que une percepción, decisión y acción.
- **Conectividad:** cómo el sistema se comunica con otros.

1.2 Maduración por nivel

Subconcepto	Exploradores	Constructores	Inventores	Innovadores
Energía y circuitos	Circuito simple: pila, cable, motor; encender/apagar	Protoboard, polaridad, resistencias; alimentar Arduino	Gestión de energía y consumo; baterías recargables	Eficiencia energética; alimentación de sistemas conectados y autonomía
Estructuras y mecanismos	Palancas, ruedas y ejes; uniones	Engranajes, poleas, transmisión; chasis	Mecanismos compuestos; grados de libertad	Mecatrónica integrada; diseño estructural de dispositivos
Sensado	El cuerpo y el tacto como interruptor (Makey Makey)	Sensores básicos: botón, LDR, ultrasónico	Múltiples sensores; calibración y lectura de datos	Fusión de sensores; sensores en red y telemetría
Actuación	Un motor gira, una luz enciende	LED, zumbador, servo, motor DC controlados	Actuadores con lógica; PWM y control de posición	Actuadores dentro de sistemas; control remoto/ automático
Control y automatización	Causa–efecto directo (toco → suena)	Lazo abierto: entrada → lógica → salida	Lazo cerrado: sensa → decide → actúa (autonomía)	Control distribuido; automatización inteligente
Conectividad	—	—	Introducción (Pico W, opcional)	BLE / WiFi / MQTT; IoT y dispositivos que dialogan

> **Nota de continuidad:** el subconcepto de **control y automatización** es la bisagra con el hilo de programación. “Sensa → decide → actúa” es, a la vez, un concepto de robótica y un algoritmo. Por eso la convergencia entre ambos hilos es natural y debe hacerse explícita.

2. Hilo conceptual de Diseño

Diseño en Ares no es “decorar”. Es un hilo tan importante como el de programación y el de robotización, y se organiza en **tres sub-hilos**, cada uno con su propia progresión y sus **herramientas gratuitas** que evolucionan nivel a nivel:

- **A · Diseño de prototipo y modelado** — la *forma* (de bocetos en Scratch a CAD 3D).
- **B · Diseño esquemático** — las *conexiones* (cómo van cableados los circuitos).
- **C · Diseño artístico** — el *acabado* y la identidad del prototipo.

Tres prácticas atraviesan los tres sub-hilos en todos los niveles: **diseño centrado en el usuario** (¿para quién y para qué?), **prototipado iterativo** (probar, medir, mejorar) y, en los niveles altos, **diseño de sistema** (cómo encajan las partes en un todo).

> **Nota:** todas las herramientas listadas son gratuitas. Matices: *BlocksCAD* tiene edición gratuita (la Education es de pago); *Fritzing* es libre y de código abierto con donación opcional; *Onshape* es gratis para uso educativo/personal (los planes comerciales son de pago); *Figma* tiene plan gratuito.

2.A · Diseño de prototipo y modelado (la forma)

Nivel	Qué se diseña	Herramientas gratuitas
Exploradores	Boceto del prototipo y *mockup* digital del proyecto (idea, escenario, personajes)	Papel y lápiz · **Scratch**
Constructores	Primeras piezas 3D para imprimir o cortar; modelado guiado	**Tinkercad** (3D) · **BlocksCAD** (modelado por bloques, enlaza con programación)
Inventores	Modelado 3D propio del prototipo (carcasas, soportes, piezas)	**Tinkercad** · **SketchUp Free** · inicio en **FreeCAD**
Innovadores	Diseño de producto y piezas para manufactura; modelado paramétrico	**FreeCAD** · **Onshape** (edu/personal) · **SelfCAD**

2.B · Diseño esquemático (las conexiones)

Nivel	Qué se diseña	Herramientas gratuitas
Exploradores	El circuito simple representado con símbolos básicos (pila–cable–motor)	Papel y símbolos impresos
Constructores	Diagrama y simulación del circuito Arduino + protoboard	**Tinkercad Circuits** (simula) · **Fritzing** (vista breadboard)
Inventores	Esquemático del proyecto (Pico, varios sensores); simular antes de armar	**Fritzing** · **Wokwi** (simulador online) · **EasyEDA**
Innovadores	Esquemático formal y diagrama de arquitectura del sistema IoT (dispositivo–red–datos)	**EasyEDA** · **KiCad**

2.C · Diseño artístico (el acabado y la identidad)

Nivel	Qué se diseña	Herramientas gratuitas
Exploradores	Decoración, color y materiales del montaje	Materiales físicos · **Scratch** (arte digital)
Constructores	Estética funcional; acabado de piezas (pintura, vinilo, grabado láser)	Corte láser · **Canva** (etiquetas, carteles)
Inventores	Identidad visual del prototipo (logo, etiqueta, carcasa coherente)	**Inkscape** · **GIMP** / **Krita** · **Canva**
Innovadores	Branding del producto y **UI/UX** del dashboard/app	**Figma** · **Inkscape** · **Canva**

2.D · Resumen de herramientas por nivel

Sub-hilo	Exploradores	Constructores	Inventores	Innovadores
A · Modelado/ forma	Scratch, papel	Tinkercad, BlocksCAD	Tinkercad, SketchUp Free, FreeCAD	FreeCAD, Onshape, SelfCAD
B · Esquemático	Papel, símbolos	Tinkercad Circuits, Fritzing	Fritzing, Wokwi, EasyEDA	EasyEDA, KiCad
C · Artístico	Materiales, Scratch	Canva, corte láser	Inkscape, GIMP/ Krita, Canva	Figma, Inkscape, Canva

> **Notas de continuidad (cómo el diseño se enlaza con los otros hilos):** - El sub-hilo **B (esquemático)** es la bisagra directa con **robotización**: el circuito que se diagrama es el que luego se monta. Diseñar el esquemático *antes* de cablear reduce errores y enseña a leer un sistema. - El sub-hilo **C (artístico)** en Innovadores se convierte en **UI/UX** y se enlaza con el **front-end del hilo de programación** (HTML/CSS/JS/Bootstrap). El diseño de la pantalla OLED en Inventores es el ensayo de esa experiencia. - El sub-hilo **A (modelado)** alimenta la **fabricación propia** de ROBOTSchool (corte láser e impresión 3D), una ventaja competitiva real de la empresa.

---

3. La IA como capa transversal (overlay)

A diferencia de los tres hilos, el **buen uso de la IA no tiene una progresión de hardware ni de técnica manual**: es una **competencia de criterio** que se aplica sobre cualquiera de los tres hilos. Por eso no es un cuarto hilo paralelo, sino una **capa que atraviesa el tejido**.

Se manifiesta de tres formas, en cualquier nivel: - **IA como herramienta del proyecto** (clasificar, predecir, reconocer) — se monta sobre robotización/datos. - **IA como asistente del estudiante** (apoyo para programar o diseñar) — se monta sobre programación/diseño. - **IA como objeto de reflexión ética** (sesgos, privacidad, cuándo NO automatizar) — se monta sobre todo el proyecto.

El criterio transversal, en todos los niveles, es el mismo: **distinguir entre delegar el pensamiento (mal uso) y amplificar el propio (buen uso)**. La progresión está en la complejidad del juicio, no en una técnica. (Detalle por nivel en `01…§5.1`.)

4. Modelo de convergencia: la anatomía de un proyecto

Cada proyecto Ares se diseña como el **trenzado de una hebra de cada hilo**, más la capa de IA. La regla de oro:

> **Todo proyecto debe tener un "punto de convergencia": una tarea que sería imposible de resolver activando solo uno o dos hilos.** Si un proyecto se puede completar sin diseño, o sin programación, o sin robotización, entonces no es un proyecto Ares: es una actividad de una sola materia disfrazada.

4.1 Matriz de convergencia por nivel (proyectos ancla)

Nivel 1 · Exploradores — "El carrito que avanza"

Hilo	Hebra activada
Programación	Secuencia y evento (encender el motor al activar)
Robotización	Energía+circuito (pila–motor), mecanismo (ruedas/ejes), actuación (gira)
Diseño	Forma del carrito, decoración, "¿para quién/para qué?"
IA (overlay)	Conversación: "¿podría un carrito moverse solo? ¿qué necesitaría sentir?"
Punto de convergencia	**Que el carrito construido (robot) avance al activarse (programación) y sirva para la misión que el equipo le diseñó (diseño).**

Nivel 2 · Constructores — "Riego automático"

Hilo	Hebra activada
Programación	Condicional (umbral de humedad), variables
Robotización	Sensado (humedad), actuación (bomba/servo), control entrada→lógica→salida
Diseño	Forma del dispositivo, fabricación guiada (soporte en 3D/láser), usuario (quién cuida la planta)
IA (overlay)	Clasificar con ejemplos: “¿planta sana o seca?”
Punto de convergencia	**Que el dispositivo decida regar solo cuando el suelo lo necesita: imposible sin sensor (robot) + condicional (programación) + un diseño que el usuario pueda usar.**

Nivel 3 · Inventores — “Invernadero autónomo”

Hilo	Hebra activada
Programación	Funciones, listas, registro de datos
Robotización	Múltiples sensores, lazo cerrado (autonomía), actuación con lógica
Diseño	Modelado 3D propio, DCU, prototipado iterativo, interfaz OLED
IA (overlay)	Usar los datos para clasificar/decidir; conversación sobre sesgos en los datos
Punto de convergencia	**Un sistema que mantiene el cultivo por sí mismo y muestra su estado: exige modularizar (programación), integrar varios sensores/actuadores (robot) y una interfaz que comunique (diseño).**

Nivel 4 · Innovadores — “Monitoreo ambiental IoT”

Hilo	Hebra activada
Programación	Clases, concurrencia, SQL, dashboard
Robotización	Sensores en red, telemetría, conectividad (WiFi/MQTT)
Diseño	UI/UX del dashboard, diseño de sistema end-to-end, branding
IA (overlay)	Predicción/alertas con modelos; ética de datos y privacidad
Punto de convergencia	**Un sistema conectado que mide el ambiente y lo presenta a una comunidad: imposible sin el dispositivo conectado (robot), el back/front y la base de datos (programación) y una experiencia de usuario clara (diseño).**

4.2 Cómo madura la convergencia

La convergencia también progresa: en Exploradores los tres hilos se **tocan** (un montaje simple); en Innovadores se **integran en un sistema** donde ninguna hebra es separable. La ambición del punto de convergencia sube con el nivel.

---

5. Plantilla operativa de convergencia (para autores)

Cada proyecto del mapa curricular se documenta —además de la ficha STEAM de `01…§6.1`— con este trenzado explícito:

1. **Reto y ODS.**
2. **Hebra de programación:** estructura(s) y competencia(s) de PC que se ejercen.
3. **Hebra de robotización:** sub-hilos activados (de §1.1) y las herramientas/componentes.
4. **Hebra de diseño:** sub-hilos activados — **A** modelado/forma, **B** esquemático, **C** artístico (de §2) y las herramientas gratuitas del nivel.
5. **Capa de IA:** qué forma toma (herramienta / asistente / reflexión ética) y qué criterio se evalúa.
6. **Punto de convergencia:** la tarea integradora que exige los tres hilos a la vez (redactada como un enunciado verificable).
7. **Evidencia y evaluación:** por hebra **y** una evaluación integrada del punto de convergencia.

> **Advertencia de mentor (el error más común):** es fácil enseñar los tres hilos en silos durante el bimestre y “juntarlos” solo de nombre al final. Eso produce proyectos donde el

> diseño es decoración y la robótica es un adorno del código. El antídoto es redactar **primero** el punto de convergencia (paso 6) y derivar las hebras desde ahí, no al revés.

---

6. Secuencia de adopción de herramientas por grado

Cada nivel abarca **tres grados**. El error a evitar es introducir todas las herramientas del nivel el primer año: satura al docente y al estudiante. Reglas:

- **Máximo 1–2 herramientas nuevas por grado**, y por sub-hilo.
- **Consolidar antes de sumar:** una herramienta no se "ve y se abandona"; se usa hasta dominarla.
- **BIPES como puente:** en 6º, **BIPES** (programación por bloques para MicroPython, web y gratuita) permite pasar de los bloques de mBlock a MicroPython **sin romper la continuidad**: el estudiante arma en bloques y *ve el código Python real* que generan. El salto a texto puro (Thonny) deja de ser abrupto.

6.1 Nivel 1 · Exploradores

Grado	Programación	Robotización	Diseño	Foco del grado
Transición	Scratch Jr / Scratch (guiado)	Makey Makey	A: papel / Scratch	Causa–efecto
1°	Scratch (secuencia, eventos)	Makey Makey + materiales conductores	C: decoración	Secuencias
2°	Scratch (intro a bucles)	Circuito simple (motor, pila), mecánica	A: boceto del prototipo	Primer mecanismo

6.2 Nivel 2 · Constructores

Grado	Programación	Robotización	Diseño	Foco del grado
3°	mBlock (bloques) + Arduino	Sensores básicos, protoboard	B: **Tinkercad Circuits** (simular antes de armar)	Entrada → salida
4°	mBlock (bucles, condicionales)	Más sensores y actuadores	A: **Tinkercad 3D** · B: **Fritzing**	Mecanismo automatizado
5°	mBlock + *asomada* a C++	Proyecto integrador del nivel	C: **Canva** (acabado)	Automatización + acabado

6.3 Nivel 3 · Inventores

Grado	Programación	Robotización	Diseño	Foco del grado
6°	**BIPES** (bloques → MicroPython) + leer el texto en **Thonny**	Raspberry Pi Pico + sensores	B: **Fritzing**	**Puente bloques → texto**
7°	MicroPython en texto (BIPES como apoyo)	Múltiples sensores, lazo cerrado	A: **FreeCAD** / **SketchUp** · B: **Wokwi** (simular)	Autonomía + modelado propio
8°	MicroPython + estructuras de datos	Registro y manejo de datos	B: **EasyEDA** · C: **Inkscape** (identidad)	Datos + esquemático formal

6.4 Nivel 4 · Innovadores

Grado	Programación	Robotización	Diseño	Foco del grado
9°	MicroPython/Python en ESP32 (consolidar) — BIPES como rampa rápida	Conectividad básica (WiFi)	B: **EasyEDA**	Embebido + conectividad
10°	C++ donde convenga + front-end (HTML/ CSS/JS, Bootstrap)	MQTT / IoT	C: **Figma** (UI/UX) · B: **KiCad** (opcional)	Front-end + protocolos
11°	**SQL** + dashboard	Sistema IoT completo	A: **FreeCAD** / **Onshape** (producto)	Datos + sistema + proyecto integrador

> **Regla de oro de adopción:** si al final de un grado el estudiante no usa con soltura las herramientas que se introdujeron, **no se suma una nueva** en el siguiente: se consolida. La secuencia es una guía, no una obligación de calendario.

---

7. Puntos de convergencia (estado y siguiente paso)

Los **puntos de convergencia de los proyectos ancla** ya están definidos en §4.1 (uno por nivel). El siguiente hito es redactarlos para **todos** los proyectos del mapa curricular y crear la **rúbrica integrada** que evalúe el trenzado, no cada hilo por separado.

Formato de cada punto de convergencia (enunciado verificable): *“El proyecto exige <tarea> que es imposible sin <hebra de programación> + <hebra de robotización> + <hebra de diseño>.”*

8. Pendientes para los siguientes hitos

1. Redactar el **punto de convergencia** para cada proyecto del mapa curricular (no solo los ancla).
2. **Rúbrica integrada** que evalúe el trenzado, no solo cada hilo por separado.
3. Diagrama visual del tejido por nivel para la guía docente (ya reflejado en la landing).

Fin del documento v0.1. Construido sobre `01_hilo_conceptual_ares.md` y `02_mapa_curricular_ares.md`.

Línea Ares — Hilo de Programación en Profundidad

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Profundiza el **hilo de programación** definido en `01` (competencias de pensamiento computacional + estructuras). Aquí se añaden tres capas que se trabajan **antes y alrededor** del código del robot, para desarrollar de verdad el **pensamiento computacional**: (A) **representar el algoritmo** (diagramas de flujo y pseudocódigo), (B) **fundamentos de lógica** (booleana y compuertas), y (C) **entornos de práctica** (Karel / Reeborg's World, Blockly y actividades desenchufadas). Todo mapeado por banda y conectado a los proyectos y libros.

---

0. La idea

Programar no empieza en el teclado: empieza **pensando el algoritmo**. Antes de cablear un sensor o escribir código, el estudiante **diseña su solución** (diagrama de flujo / pseudocódigo) y **practica la lógica** en un entorno seguro (Karel/Reeborg). Así el código del robot es la *consecuencia* de un razonamiento, no un copiar-pegar.

> **Regla:** en cada proyecto, **primero el algoritmo** (representación), **luego el código**. El robot ejecuta lo que el estudiante ya pensó.

---

1. Representar el algoritmo (antes de codear)

1.1 Diagramas de flujo

Una forma **visual** de mostrar los pasos y las decisiones. Símbolos básicos:

- **Óvalo** — inicio / fin.
- **Rectángulo** — una acción ("encender LED").
- **Rombo** — una decisión ("¿humedad < umbral?") con salidas Sí/No.
- **Flechas** — el orden (incluye los bucles, que "regresan").

1.2 Pseudocódigo

Escribir el algoritmo en **lenguaje casi natural**, estructurado, sin la sintaxis exacta de ningún lenguaje. Es el puente entre el diagrama y el código real:

```
INICIO
  REPETIR por siempre
    leer humedad del sensor
    SI humedad < umbral ENTONCES
      encender la bomba
    SI NO
      apagar la bomba
  FIN REPETIR
FIN
```

1.3 Progresión por banda

Banda	Cómo representa su algoritmo
Exploradores	Secuencia con **dibujos/íconos** (“primero… luego…”); tarjetas de pasos en orden
Constructores	**Diagrama de flujo** simple (acción + decisión + bucle) antes de armar en bloques
Inventores	**Pseudocódigo** + diagrama; de ahí pasa a MicroPython
Innovadores	Diagrama de flujo y pseudocódigo como **diseño previo** del sistema (varios módulos)

---

2. Fundamentos de lógica

2.1 Lógica booleana y condiciones

Todo condicional se apoya en **verdadero/falso**. Las condiciones se combinan con **Y (AND)**, **O (OR)** y **NO (NOT)**: - *“SI hace calor **Y** la tierra está seca, entonces…”* - *“SI **NO** hay nadie, apaga la luz.”*

2.2 Compuertas lógicas (AND, OR, NOT) — dónde y cómo

Las compuertas son la versión “física/formal” de esa lógica. **Recomendación de dónde ponerlas** (no son para todas las edades):

Banda	Tratamiento de las compuertas
Exploradores	**No** (aún) — se trabaja “y/o/no” solo en lenguaje cotidiano
Constructores	**Desenchufado** / **juego**: compuertas con interruptores y un LED (AND = dos interruptores en serie; OR = en paralelo). Conecta con circuitos
Inventores	Formalizar: tablas de verdad simples y condiciones compuestas en el código (`and` , `or` , `not`)
Innovadores	Lógica en decisiones de sistema y en electrónica (sensores múltiples, alarmas)

> **Nota de mentor:** las compuertas como **circuito con interruptores y LED** (Constructores) son potentísimas porque unen los tres hilos: lógica (programación), circuito (robotización) y un montaje (diseño). Evitar enseñarlas como álgebra de Boole abstracta antes de tiempo.

3. Entornos de práctica (gamificados y visuales)

Espacios para **practicar la lógica** sin depender del hardware. Todos **gratuitos**.

Banda	Entornos recomendados	Para qué
Exploradores	**ScratchJr**, **Code.org** (cursos express), **Lightbot**, **Blockly Games: Laberinto**	Secuencias y eventos; “dar órdenes” en orden
Constructores	**Karel** / **Reeborg’s World** (bloques), **Blockly Games**, **Scratch**, **Code.org**	Bucles y condicionales con un robot virtual; del bloque a la lógica
Inventores	**Reeborg’s World en Python**, **Blockly→Python**	Puente bloques→texto: el mismo robot, ahora en MicroPython/Python
Innovadores	**Python** (problemas y retos), estructuras de datos	Lógica de programación “real” antes de los sistemas

> **Karel y Reeborg's World** (gratis, web, sin instalar): un robot virtual recibe **tareas** y el estudiante escribe el programa para cumplirlas. Reeborg permite **bloques, Python y JavaScript** en el mismo entorno — por eso es el **puente ideal** Blockly→MicroPython que necesita Inventores.

---

4. Pensamiento computacional desenchufado (sin computador)

Actividades para desarrollar PC sin pantalla — ideales para empezar cualquier tema: - **El robot humano:** un estudiante da "instrucciones" paso a paso a otro que actúa como robot (secuencia, depuración). - **Tarjetas de algoritmo:** ordenar tarjetas de pasos para lograr una meta. - **Compuertas con el cuerpo / interruptores:** juego de AND/OR/NOT. - **Laberintos en papel:** escribir las instrucciones para salir.

---

5. Matriz del hilo de programación profundo

Banda	Representación	Lógica	Entorno de práctica	Desenchufado
Exploradores	Dibujos/secuencia	y/o/no cotidiano	ScratchJr, Lightbot, Blockly	Robot humano, tarjetas
Constructores	Diagrama de flujo	Compuertas como circuito	Karel/Reeborg (bloques), Scratch	Compuertas con interruptores
Inventores	Pseudocódigo + diagrama	Tablas de verdad, condiciones compuestas	Reeborg en Python	Laberintos, depuración
Innovadores	Diseño previo del sistema	Lógica de sistema y electrónica	Python (retos)	Diseño de algoritmos en equipo

---

6. Cómo se integra a los proyectos y libros

Esto **enriquece los libros** (es lo que pediste para completarlos con contenido):

1. **Nuevo paso en cada microproyecto: "Diseña tu algoritmo".** Antes de la sección de programación, el estudiante dibuja el **diagrama de flujo** o escribe el **pseudocódigo** de lo que va a hacer.
2. **Práctica previa en Karel/Reeborg.** Un reto en el entorno virtual con la **misma lógica** del proyecto (p. ej., "el robot decide girar si hay un muro" antes de "el riego decide regar si hay sequía").
3. **Actividad desenchufada de apertura** por libro, para introducir el concepto sin pantalla.
4. En el **comodín** `COM-BS-PROG` se incluyen las cartillas de diagramas de flujo y pseudocódigo (digitales imprimibles).

> Con esto, cada libro tiene la cadena completa: **desenchufado → representar (diagrama/pseudocódigo) → practicar (Karel/Reeborg) → programar el robot.**

---

7. Pendiente

1. Crear las **plantillas de diagrama de flujo y pseudocódigo** (imprimibles) como parte del comodín de programación.
2. Diseñar **2–3 retos de Karel/Reeborg por banda** alineados a los proyectos.
3. Banco de **actividades desenchufadas** por concepto.
4. Insertar el paso **"Diseña tu algoritmo"** en los microproyectos de los libros (regenerar con el motor).

Fin del documento v0.1. Profundiza `01` *(hilo de programación) y se conecta con los libros (* `11` *,* `12` *) y comodines (* `13` *).*

Fuentes: Reeborg's World · GitHub — Reeborg (clon mejorado de Karel).

Línea Ares — Hilo de Robotización en Profundidad

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Profundiza el **hilo de robotización** de `03 §1` (sub-hilos: energía/circuitos, mecanismos, sensado, actuación, control, conectividad). Aquí se detallan los **conceptos** detrás de cada sub-hilo, **dónde** se trabajan por banda, las **herramientas de simulación** y cómo se integran a proyectos y libros. Regla general: **simular antes de construir**, y **seguridad primero**.

---

1. Energía y circuitos

Conceptos: voltaje (empuje), corriente (flujo), resistencia (freno); polaridad; circuito en **serie** y en **paralelo**; cortocircuito y por qué evitarlo.

Banda	Profundidad
Exploradores	Pila → cable → motor/LED; “el circuito debe estar cerrado para que funcione”
Constructores	Protoboard, polaridad, **resistencia para el LED**, serie vs paralelo
Inventores	Alimentación de la placa, consumo, fuentes; cuidar 3.3V vs 5V
Innovadores	Eficiencia energética, autonomía, alimentación de sistemas conectados

> Para los grados altos se introduce la **Ley de Ohm** ($V = I \times R$) de forma aplicada (elegir la resistencia de un LED), nunca como fórmula vacía.

---

2. Estructuras y mecanismos

Conceptos: palanca, rueda-eje, **engranajes** (relación de transmisión), polea/correa, leva, biela-manivela; grados de libertad; transmisión de movimiento y fuerza.

Banda	Profundidad
Exploradores	Palancas, ruedas y ejes; “qué se mueve y cómo”
Constructores	Engranajes y poleas; **relación de transmisión** (más vueltas vs más fuerza)
Inventores	Mecanismos compuestos; grados de libertad (un brazo, una compuerta)
Innovadores	Mecatrónica: integrar mecanismo + electrónica + control

3. Sensado (percepción)

Conceptos: señal **digital** (sí/no) vs **analógica** (un rango); lectura (0–1023 / 0–4095); **calibración** (qué número = qué realidad); muestreo (cada cuánto leer); ruido.

Banda	Profundidad
Exploradores	El tacto como interruptor (Makey)
Constructores	Botón (digital), LDR/humedad (analógico); leer un valor
Inventores	Varios sensores, **calibrar** umbrales, promediar para reducir ruido
Innovadores	Fusión de sensores, telemetría, sensores en red

4. Actuación

Conceptos: salida digital (encender/apagar); **PWM** (controlar intensidad/posición/velocidad); **torque**; cuándo el actuador necesita **más potencia** que la que da la placa (→ relé/driver).

Banda	Profundidad
Exploradores	El motor gira, el LED enciende
Constructores	LED, zumbador, **servo (PWM, ángulo)**, motor DC; **relé** para la bomba
Inventores	Control de posición/velocidad; varios actuadores coordinados
Innovadores	Actuadores dentro de un sistema; control remoto/automático

5. Control y automatización

Conceptos: lazo abierto (hace siempre lo mismo) vs **lazo cerrado** (usa el sensor para decidir: realimentación); control **on/off** (umbral) y, en avanzado, idea de control **proporcional**.

Banda	Profundidad
Exploradores	Causa–efecto directo
Constructores	Entrada → lógica → salida (lazo abierto y primer on/off)
Inventores	**Lazo cerrado**: sensa → decide → actúa, con realimentación
Innovadores	Control distribuido; automatización inteligente; múltiples lazos

6. Conectividad

Conceptos: comunicación entre dispositivos; protocolos: **I2C** (sensores/pantallas), **WiFi/BLE** (a la red/celular), **MQTT/HTTP** (a internet); cliente/servidor; publicar/suscribir.

Banda	Profundidad
Exploradores / Constructores	— (cableado local)
Inventores	I2C (OLED); primera conectividad opcional (Pico W)
Innovadores	WiFi, BLE, **MQTT**, HTTP; IoT y dispositivos que dialogan

7. Herramientas de simulación (simular antes de construir)

Todas **gratuitas**: - **Tinkercad Circuits** — simular el circuito Arduino + el código, sin quemar nada. - **Wokwi** — simular Arduino/ESP32/Pico con sensores y ver el comportamiento. - **Falstad** / **CircuitJS** — entender el flujo de corriente en un circuito. - **Fritzing** — dibujar el esquemático y la vista de protoboard.

> **Regla de oro:** el estudiante **arma y prueba en el simulador** antes de tocar el hardware. Ahorra componentes, evita errores de conexión y permite que todos practiquen aunque no tengan el kit en la mano.

---

8. Seguridad (transversal y obligatoria)

- Manejo de herramientas, **soldadura** (supervisada), **corte láser** e impresión 3D.
- Cuidado con polaridad, cortos y voltajes (3.3V vs 5V).
- Agua + electrónica = separar (proyectos de riego/invernadero).
- Va en el comodín/curso de **seguridad y fabricación** (`10`).

---

9. Matriz del hilo de robotización profundo

Banda	Energía	Mecanismo	Sensado	Actuación	Control	Conectividad
Exploradores	Circuito cerrado	Palanca, rueda	Tacto	Motor/LED	Causa-efecto	—
Constructores	Serie/ paralelo, resistencia	Engranajes, transmisión	Digital/ analógico	Servo, relé	Lazo abierto, on/off	—
Inventores	Consumo, 3.3V/5V	Grados de libertad	Calibrar, promediar	Posición/ velocidad	**Lazo cerrado**	I2C, Pico W
Innovadores	Eficiencia, autonomía	Mecatrónica	Fusión, telemetría	En sistema	Distribuido	WiFi/BLE/ MQTT

---

10. Cómo se integra a los proyectos y libros

1. Nuevo paso **“Diseña tu circuito”**: el estudiante dibuja el **esquemático** (Fritzing) y lo **simula** (Tinkercad/Wokwi) antes de cablear.
2. Para los proyectos con mecanismo, paso **“Prueba el mecanismo”** (relación de transmisión, grados de libertad).
3. El concepto del proyecto se ancla a su sub-hilo dominante (p. ej., riego = sensado + control + actuación).
4. Los comodines de cada elemento (`13`) traen la conexión y el dato técnico; este documento da el **porqué**.

11. Pendiente

1. Retos de simulación (Tinkercad/Wokwi) por proyecto.
2. Cartilla de seguridad ilustrada (parte del comodín de seguridad).
3. Insertar el paso “Diseña/simula tu circuito” en los microproyectos (regenerar libros).

Fin del documento v0.1. Profundiza `03 §1` *y se conecta con comodines (* `13` *) y libros (* `11` *,* `12` *).*

Línea Ares — Hilo de Diseño en Profundidad

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Profundiza el **hilo de diseño** de `03 §2` (sub-hilos: A modelado/forma, B esquemático, C artístico/experiencia). Aquí se detallan el **proceso de diseño**, los conceptos de cada sub-hilo, el **diseño para fabricación** (clave porque ROBOTSchool corta en láser e imprime en 3D), las herramientas y la integración a proyectos y libros.

---

1. El proceso de diseño (transversal a todo)

Una versión simple de *design thinking*, igual en todas las bandas (cambia la profundidad):

1. **Empatizar** — ¿para quién es? ¿qué necesita?
2. **Definir** — el problema en una frase.
3. **Idear** — bocetar varias soluciones.
4. **Prototipar** — construir una primera versión (¡en papel!).
5. **Probar** — usarla, ver qué falla, mejorar.

> El **prototipo en papel** del libro y el **anexo recortable** son, literalmente, los pasos 4 y 5. El MDF llega cuando el diseño ya se probó.

---

2. Sub-hilo A · Modelado / forma

Conceptos: boceto, vistas, proporción; del 2D al **3D**; en CAD: **medidas, tolerancias y ensamble** (que las piezas encajen).

Banda	Profundidad	Herramientas
Exploradores	Bocetar; forma y color	Papel, Scratch
Constructores	Piezas 3D simples; pensar el ensamble	Tinkercad, BlocksCAD
Inventores	Modelado propio con medidas y tolerancias	FreeCAD, SketchUp
Innovadores	Diseño de producto, paramétrico	FreeCAD, Onshape

3. Diseño para fabricación (DFM) — láser y 3D

Lo que hace única a ROBOTSchool: diseñar **pensando en cómo se va a cortar/imprimir**.

- **Corte láser (MDF):** encajes tipo “dedos” (caja), espesor del material, **kerf** (lo que “come” el láser), líneas de corte vs grabado.
- **Impresión 3D:** orientación de la pieza, soportes, paredes mínimas, voladizos.
- **Del papel al MDF:** el prototipo de papel valida el ensamble antes de gastar material.

Banda	DFM
Exploradores	Piezas pre-cortadas; arma y entiende el encaje
Constructores	Diseña piezas simples para cortar; respeta el espesor
Inventores	Diseña encajes y soportes propios
Innovadores	Diseño para manufactura; iteración rápida

4. Sub-hilo B · Esquemático

Conceptos: símbolos, leer y dibujar un circuito, del **esquema** al **montaje** real; por qué diseñar el esquema *antes* de cablear.

Banda	Profundidad	Herramientas
Exploradores	Símbolos básicos en papel	Papel
Constructores	Esquema + vista protoboard	Tinkercad Circuits, Fritzing
Inventores	Esquema del proyecto; simular	Fritzing, Wokwi, EasyEDA
Innovadores	Esquema formal + diagrama de sistema	EasyEDA, KiCad

5. Sub-hilo C · Artístico y experiencia

Conceptos: color, contraste, **tipografía**, jerarquía visual, identidad/branding; y para pantallas: **UI/UX** (qué ve primero el usuario, *affordances*, claridad de una alerta).

Banda	Profundidad	Herramientas
Exploradores	Color y decoración	Materiales, Scratch
Constructores	Estética funcional, acabado	Canva
Inventores	Identidad del prototipo; primera interfaz (OLED)	Inkscape, Canva
Innovadores	UI/UX de dashboards y apps; branding del sistema	Figma, Inkscape

6. Diseño centrado en el usuario y prototipado iterativo

- **DCU:** decisiones a partir del usuario real (¿quién usa esto?, ¿le sirve?).
- **Iterar:** prototipo → prueba → ajuste; no enamorarse de la primera idea.
- **Probar con usuarios:** mostrar el prototipo a un compañero/familiar y observar qué no entiende.

Banda	DCU / iteración
Exploradores	“¿Para quién es?” (intuitivo)
Constructores	Pensar en el usuario del mecanismo; varias versiones
Inventores	Investigación de usuario; ciclos de prototipo
Innovadores	DCU formal; pruebas de usabilidad; MVP

7. Matriz del hilo de diseño profundo

Banda	Modelado (A)	Fabricación (DFM)	Esquemático (B)	Artístico/UX (C)	Usuario
Exploradores	Boceto	Piezas pre-cortadas	Símbolos en papel	Color, decorar	¿para quién?
Constructores	Tinkercad	Diseñar para láser	Tinkercad Circuits/Fritzing	Acabado, Canva	usuario del mecanismo
Inventores	FreeCAD/ SketchUp	Encajes y soportes	Fritzing/ EasyEDA	Identidad, OLED	investigación de usuario
Innovadores	Onshape, paramétrico	Diseño para manufactura	EasyEDA/KiCad	UI/UX, branding	usabilidad, MVP

8. Cómo se integra a los proyectos y libros

1. Cada microproyecto abre con un mini **proceso de diseño** (empatizar→definir→idear) antes de construir.
2. El **anexo recortable** se presenta como un ejercicio de **DFM** (¿por qué esta pieza encaja así?).
3. Paso **“Diseña la forma / la interfaz”** según el sub-hilo dominante del proyecto.
4. En grados altos, una **prueba de usabilidad** corta como parte de la socialización.

9. Pendiente

1. Plantillas de boceto y de “ficha de usuario” (imprimibles).
2. Guía de DFM ilustrada (reglas de láser y 3D) como comodín de diseño.
3. Insertar el paso de diseño en los microproyectos (regenerar libros).

Fin del documento v0.1. Profundiza `03 §2` y se conecta con comodines (`13`) y libros (`11` , `12`).

Línea Ares — Mapa Curricular por Nivel

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Este documento detalla los cuatro niveles de Ares. Se construye **sobre el hilo conceptual** (ver `01_hilo_conceptual_ares.md`): cada nivel ejerce las mismas competencias de pensamiento computacional y las mismas estructuras de programación, a mayor profundidad. Los proyectos son **ejemplos ancla**, no una lista cerrada; sirven de plantilla para que el equipo genere los demás.

Cómo leer cada nivel

Cada nivel se describe con la misma estructura, para que la progresión sea evidente:

1. **Identidad y meta del nivel** — el verbo dominante y a qué aspira el estudiante.
2. **Tecnología y entorno** — placa, componentes y lenguaje.
3. **Pensamiento computacional** — qué competencias se profundizan (§3 del hilo).
4. **Estructuras de programación** — qué se introduce y consolida (§4 del hilo).
5. **Ejes transversales** — cómo aparecen IA, diseño y robotización (§5 del hilo).
6. **Proyectos ancla** — con ODS, conexiones STEAM y producto.
7. **Estándares ISTE** y **evaluación**.

Nivel 1 · Exploradores

Transición · 1° · 2° — Makey Makey + mecánica y mecanismos

Identidad y meta

El estudiante **descubre** que puede hacer que las cosas reaccionen. Aprende que toda acción tecnológica responde a una causa, y que las acciones se ordenan en secuencias. La meta no es programar “bien”, sino construir intuición de causa–efecto y de “primero esto, luego aquello”.

Tecnología y entorno

- **Makey Makey** y materiales conductores (frutas, plastilina, papel aluminio, grafito).
- Mecánica básica: palancas, ruedas y ejes, engranajes simples.

- Circuitos elementales: motores DC, cables, portapilas, interruptores.
- Entorno de programación: **bloques** (Scratch / mBlock), con apoyo de pictogramas para pre-lectores.

Pensamiento computacional (lo que se siembra)

- **Descomposición:** separar una acción en pasos simples.
- **Patrones:** notar que repetir una acción produce el mismo efecto.
- **Algoritmos:** secuencias y eventos (“cuando toco… suena”).
- **Depuración:** “no pasó lo que esperaba” → volver a intentar.

Estructuras de programación

Secuencia y eventos (introducción visual). No se espera dominio de bucles ni condicionales: se *exponen*, no se exigen.

Ejes transversales

- **IA bien usada:** conversación guiada “¿qué puede y qué no puede hacer una máquina?”; los asistentes de voz como ejemplo cotidiano.
- **Diseño:** decoración y forma física del montaje; bocetos antes de construir.
- **Robotización:** mecanismos que se mueven (el carrito, la palanca) y circuitos que encienden.

Proyectos ancla

Proyecto	Reto / ODS	Estructuras	Conexiones STEAM auténticas
Piano de frutas	"¿Cómo hacemos música tocando comida?" · ODS 4	Eventos	Ciencia (conductividad), Arte (música y ritmo), Mate (secuencia de notas)
Cuenta-cuentos interactivo	"Hagamos que nuestro cuento suene y responda" · ODS 4	Secuencia, eventos	Lengua (narrativa), Arte (ilustración), Tecnología (interacción)
El carrito que avanza	"Construyamos algo que se mueva solo" · ODS 11	Secuencia (encender/apagar)	Física (movimiento, fuerza), Mate (medir distancias), Ingeniería (mecanismo)

Estándares ISTE y evaluación

- **ISTE:** Pensador Computacional (1.5), Comunicador Creativo (1.6).
- **Evaluación:** observación directa y lista de cotejo (¿logra una secuencia?, ¿identifica la causa de un fallo?, ¿describe lo que hizo?). Evidencia: el montaje funcionando + relato del estudiante.

Nivel 2 · Constructores

3° · 4° · 5° — Arduino + mecánica

Identidad y meta

El estudiante **construye**: arma un mecanismo y le da comportamiento con código. Pasa de "que reaccione" a "que decida y repita". Consolida las tres estructuras base —bucles, condicionales y variables— en un entorno de bloques, y se *asoma* al código de texto.

Tecnología y entorno

- **Arduino UNO** y protoboard.
- Sensores: pulsadores, LDR (luz), ultrasónico (distancia), potenciómetro.

- Actuadores: LED, zumbador, servomotor, motor DC.
- Mecánica: estructuras móviles, transmisión por engranajes y poleas.
- Entorno: **mBlock (bloques)** como base; **asomada a C++** (ver el mismo programa en bloques y en texto).

Pensamiento computacional (lo que se profundiza)

- **Descomposición:** dividir el mecanismo en partes y el programa en bloques funcionales.
- **Patrones:** identificar repeticiones → introducir **bucles**.
- **Abstracción:** las **variables** como “cajas” que guardan un valor.
- **Algoritmos:** decisiones con **condicionales** (si la distancia es corta, frena).
- **Depuración:** probar y ajustar bloque por bloque.

Estructuras de programación

Bucles (`for` / `while`), condicionales (`if/else`), variables y operadores. C++ entra como **asomada**: se muestra la equivalencia, no se exige dominio de sintaxis (decisión de §4.2 del hilo conceptual).

Ejes transversales

- **IA bien usada:** clasificadores no-code (reconocer imágenes o sonidos con extensiones de ML de mBlock); idea de “entrenar con ejemplos”.
- **Diseño:** relación forma–función del mecanismo; fabricación guiada con corte láser e impresión 3D.
- **Robotización:** automatización real (entrada → lógica → salida).

Proyectos ancla

Proyecto	Reto / ODS	Estructuras	Conexiones STEAM auténticas
Semáforo inteligente	"¿Cómo ordena el tráfico una ciudad?" · ODS 11	Bucles, condicionales, variables (tiempos)	Física (luz/color), Mate (intervalos), Sociales (movilidad y ciudadanía vial)
Brazo o mano robótica	"Imitemos cómo agarra la mano humana" · ODS 9	Condicionales, variables (ángulos)	Biología (anatomía de la mano), Física (palancas, torque), Diseño (ergonomía)
Riego automático	"Cuidemos una planta sin desperdiciar agua" · ODS 6 / ODS 12	Condicionales (umbral de humedad), bucles	Biología (necesidades de la planta), Química (humedad del suelo), Mate (umbrales)

Estándares ISTE y evaluación

- **ISTE:** Pensador Computacional (1.5), Diseñador Innovador (1.4).
- **Evaluación:** rúbrica de dos ejes — (a) mecanismo funcional y bien construido; (b) lógica correcta (uso adecuado de bucle/condicional/variable). Evidencia: prototipo + explicación del algoritmo.

Nivel 3 · Inventores

6° · 7° · 8° — Raspberry Pi Pico + MicroPython

Identidad y meta

El estudiante **inventa**: ya no solo construye, le da **autonomía** a su creación. El sistema sensa el entorno, decide y actúa por su cuenta, y empieza a **registrar y usar datos**. Es el salto del bloque al texto real (MicroPython), apoyado en la "asomada" a C++ del nivel anterior.

Tecnología y entorno

- **Raspberry Pi Pico** (y Pico W para una primera conectividad opcional).
- Sensores: DHT (temperatura/humedad), ultrasónico, PIR (movimiento), LDR, sensores analógicos.

- Salidas: pantalla OLED, servos, motores, LEDs direccionables.
- Lenguaje: **MicroPython** (entorno Thonny).
- Fabricación: modelado 3D **propio** del estudiante (no solo guiado).

Pensamiento computacional (lo que se profundiza)

- **Descomposición:** separar el sistema en módulos — sensar / decidir / actuar.
- **Patrones:** reutilizar lógica mediante **funciones**.
- **Abstracción:** funciones con parámetros; primeros **modelos de datos**.
- **Algoritmos:** lógica con variables, sensores y **estructuras de datos** (listas).
- **Depuración:** aislar el módulo que falla (¿es el sensor, la lógica o el actuador?).

Estructuras de programación

Funciones con parámetros, listas y diccionarios, y **registro simple de datos** (guardar lecturas, calcular promedios). Primera **conectividad** opcional con Pico W.

Ejes transversales

- **IA bien usada:** clasificar los datos del propio proyecto; uso responsable de asistentes de código; conversación sobre **sesgos y calidad de los datos**.
- **Diseño: diseño centrado en el usuario** y prototipado iterativo; modelado 3D propio.
- **Robotización:** sistemas autónomos completos (sensar → decidir → actuar).

Proyectos ancla

Proyecto	Reto / ODS	Estructuras	Conexiones STEAM auténticas
Estación meteorológica	"Midamos y entendamos nuestro clima" · ODS 13	Funciones, listas, registro de datos	Física (temperatura/presión), Mate (promedios, gráficas), Geografía (clima local)
Invernadero autónomo	"Cultivemos alimento con menos recursos" · ODS 2 / ODS 12	Funciones, condicionales, múltiples sensores	Biología (fotosíntesis, riego), Química (suelo/pH), Mate (control por umbrales)
Contador de aforo / acceso	"¿Cuántas personas usan un espacio?" · ODS 11	Funciones, listas, conteo	Mate (estadística básica), Ética (privacidad de datos), Sociales (uso de espacios)

Estándares ISTE y evaluación

- **ISTE:** Pensador Computacional (1.5), Constructor de Conocimiento (1.3, por el trabajo con datos).
- **Evaluación:** rúbrica de tres ejes — PC (modularización y funciones), funcionamiento autónomo, y manejo/lectura de datos. Evidencia: sistema funcionando + bitácora de datos + análisis breve.

Nivel 4 · Innovadores

9° · 10° · 11° — ESP32 + stack web (HTML, CSS, JS, Bootstrap, SQL)

Identidad y meta

El estudiante **innova**: genera tecnología nueva y la **conecta al mundo**. Diseña sistemas completos —dispositivo, red, datos e interfaz— en torno a domótica e IoT. Es el nivel de mayor ambición y, por eso, el que exige **priorizar núcleo vs. electivo** (ver §7 del hilo conceptual).

Tecnología y entorno

- **ESP32** con conectividad **BLE y WiFi**.
- Protocolos: HTTP y **MQTT** para IoT.

- Frontend: **HTML, CSS, JavaScript, Bootstrap**.
- Datos: **SQL** y **dashboard** de visualización.
- Lenguajes: **Python** / **C++** según el subsistema.

Núcleo obligatorio vs. electivo

> **Decisión de diseño anti-sobrecarga:** el **núcleo** que todo estudiante debe lograr es un proyecto IoT con stack mínimo: *ESP32 → envía datos → se almacenan → se visualizan en un dashboard*. Todo lo demás (apps avanzadas, integración de IA, múltiples dispositivos, BD compleja) es **electivo** / **de profundización**. Así se garantiza que todos terminen un sistema completo en lugar de muchos a medias.

Pensamiento computacional (lo que se profundiza)

- **Descomposición:** arquitectura de sistema (dispositivo / red / datos / interfaz).
- **Patrones:** patrones de diseño y de datos en sistemas conectados.
- **Abstracción:** clases/objetos, APIs, esquema de base de datos.
- **Algoritmos:** eventos de red, concurrencia, consultas SQL.
- **Depuración:** diagnóstico integral (hardware, código, red y datos).

Estructuras de programación

Funciones y módulos/clases, estructuras de datos, **concurrencia y conectividad** (BLE/WiFi, MQTT), y **manejo de datos con persistencia** (SQL + dashboard).

Ejes transversales

- **IA bien usada:** integrar servicios o modelos de IA (visión, predicción) en proyectos IoT; **ética de datos, privacidad** y criterio sobre cuándo *no* automatizar.
- **Diseño: UI/UX** de dashboards y apps; diseño del sistema completo.
- **Robotización:** dispositivos conectados; domótica e IoT.

Proyectos ancla

Proyecto	Reto / ODS	Estructuras	Conexiones STEAM auténticas
Hogar inteligente (domótica)	"Hagamos una vivienda que ahorre energía" · ODS 7 / ODS 11	Clases, WiFi, eventos	Física (energía eléctrica), Mate (consumo), Economía (ahorro), Diseño (UI de control)
Monitoreo ambiental IoT	"Vigilemos el aire/agua de nuestra comunidad" · ODS 13 / ODS 11	Concurrencia, MQTT, SQL, dashboard	Ciencias (variables ambientales), Mate (estadística y datos), Informática (BD)
Sistema de alerta conectado	"Avisemos a tiempo ante un riesgo" · ODS 11	Eventos de red, notificaciones	Ética (privacidad/vigilancia), Sociales (gestión de riesgo), Diseño (interfaz)

Estándares ISTE y evaluación

- **ISTE:** Pensador Computacional (1.5), Diseñador Innovador (1.4), Comunicador Creativo (1.6), Ciudadano Digital (1.2).
- **Evaluación:** proyecto integrador con **defensa oral**; rúbrica de sistema completo (funcionamiento end-to-end, calidad de datos, diseño de interfaz, criterio ético). Evidencia: sistema IoT funcionando + dashboard + documentación técnica.

Síntesis de la progresión

	Exploradores	Constructores	Inventores	Innovadores
Verbo	Descubrir	Construir	Inventar	Innovar
Tecnología	Makey Makey	Arduino	Raspberry Pi Pico	ESP32
Lenguaje	Bloques	Bloques → C++	MicroPython	Python/C++ + web
Hito de PC	Secuencia y causa-efecto	Bucles, condicionales, variables	Funciones, datos, autonomía	Sistemas, concurrencia, datos
IA	¿Qué hace una máquina?	Clasificar con ejemplos	Datos y sesgos	Integración y ética IoT
Producto típico	Montaje interactivo	Mecanismo automatizado	Sistema autónomo con datos	Sistema IoT con dashboard

---

Pendientes para los siguientes hitos

1. **Fichas de proyecto** detalladas (una por proyecto, con la plantilla de §6.1 del hilo).
2. **Rúbricas** transversales de pensamiento computacional por nivel.
3. **Ruta de capacitación docente** (Arduino → MicroPython → ESP32).
4. **Especificación de kits** y su producción (corte láser, 3D, electrónicos).

Fin del mapa curricular v0.1.

Línea Ares — Repositorio de Proyectos (por ODS)

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Catálogo de **36 proyectos** (9 por nivel) anclados a los **Objetivos de Desarrollo Sostenible**, pensado para dar **variedad de oferta**: el colegio elige según su contexto, sus ODS prioritarios y sus recursos. Todos respetan el tejido de los tres hilos y traen su **punto de convergencia** en una frase. Los proyectos marcados con ★ son los **ancla** ya desarrollados a fondo (ver `06` para Constructores).
>
> **Formato de cada ficha breve:** *Reto · Prog / Robot / Diseño · Convergencia.* La tecnología y las herramientas son las del nivel (`02` , `03…§6`). Casi todos son **fabricables** con corte láser e impresión 3D de ROBOTSchool (se detallará en la fase de kits).

Nivel 1 · Exploradores (Transición – 2°)

Makey Makey + mecánica · Scratch (secuencia, eventos)

1. Piano de frutas ★ · **ODS 4** Reto: hacer música tocando alimentos. · Prog: eventos “al tocar” / Robot: conductividad, circuito Makey / Diseño: disposición y decoración del teclado. · **Convergencia:** suena la nota correcta solo si hay contacto (robot) + el evento programado (prog) + un teclado entendible (diseño).

2. Cuenta-cuentos interactivo ★ · **ODS 4** Reto: un cuento que suena y responde al tocarlo. · Prog: secuencia de escenas y sonidos / Robot: sensores táctiles Makey / Diseño: ilustración y narrativa. · **Convergencia:** el cuento avanza al tocar la parte correcta solo con los tres.

3. El carrito que avanza ★ · **ODS 11** Reto: algo que se mueva solo. · Prog: encender/apagar / Robot: motor, pila, ruedas y ejes / Diseño: forma del carrito y misión. · **Convergencia:** el carrito cumple su misión solo si avanza (robot), se activa (prog) y su forma sirve (diseño).

4. Clasificador de residuos (juego) · **ODS 12** Reto: aprender a separar la basura jugando. · Prog: acierto/error con sonido / Robot: tablero conductor Makey / Diseño: íconos de cada caneca. · **Convergencia:** el juego enseña a separar solo si reconoce el toque (robot) + responde bien/mal (prog) + las canecas se entienden (diseño).

5. Molino de viento con motor · **ODS 7** Reto: representar la energía del viento. · Prog: encender el motor / Robot: motor, portapilas, aspas que giran / Diseño: estructura del molino. ·

Convergencia: el molino gira y “genera” solo con motor (robot), activación (prog) y aspas bien diseñadas (diseño).

6. Mapa sonoro de animales de mi región · **ODS 15** Reto: conocer la fauna local. · Prog: cada zona reproduce un sonido / Robot: puntos táctiles Makey / Diseño: mapa ilustrado. · **Convergencia:** el mapa “habla” solo si detecta el toque (robot) + dispara el sonido (prog) + el mapa es claro (diseño).

7. Juego de hábitos saludables (quiz) · **ODS 3** Reto: reforzar higiene y alimentación. · Prog: preguntas y respuestas / Robot: botones conductores / Diseño: tablero del quiz. · **Convergencia:** el quiz funciona solo con los tres hilos juntos.

8. Gotero ahorrador (mecanismo) · **ODS 6** Reto: regar sin desperdiciar agua. · Prog: (conteo/temporización simple en Scratch) / Robot: mecanismo dosificador / Diseño: forma del gotero. · **Convergencia:** dosifica el agua justa solo con el mecanismo (robot), la guía (prog/diseño) y el soporte (diseño).

9. Alarma para mi cuarto · **ODS 11** Reto: avisar cuando se abre la puerta. · Prog: evento de activación / Robot: circuito con interruptor / Diseño: caja y aviso. · **Convergencia:** la alarma avisa solo si el circuito cierra (robot), el aviso se dispara (prog) y se instala bien (diseño).

Nivel 2 · Constructores (3° – 5°)

Arduino + mBlock (bucles, condicionales, variables) · Tinkercad, Fritzing, Canva

1. Semáforo inteligente ★ · **ODS 11** — *(detalle completo en `06`)* Reto: ordenar un cruce con peatón. · Prog: bucle + condicional + variables (tiempos) / Robot: LEDs + pulsador / Diseño: maqueta y señalización.

2. Brazo / mano robótica ★ · **ODS 9** — *(detalle en `06`)* Reto: imitar el agarre de la mano. · Prog: secuencia de ángulos / Robot: servos + mecanismo / Diseño: piezas 3D forma-función.

3. Riego automático ★ · **ODS 6 / 12** — *(detalle en `06`)* Reto: regar solo cuando hace falta. · Prog: condicional con umbral / Robot: sensor humedad + bomba / Diseño: carcasa.

4. Luz automática de pasillo · **ODS 7** Reto: ahorrar energía encendiendo solo cuando se necesita. · Prog: condicional con LDR / Robot: LDR + LED/relé / Diseño: lámpara. · **Convergencia:** la luz ahorra solo si mide la luz (robot) + decide (prog) + se instala útil (diseño).

5. Asistente de parqueo · **ODS 11** Reto: ayudar a estacionar sin golpes. · Prog: rangos con condicional / Robot: ultrasónico + zumbador / Diseño: soporte y guía visual. · **Convergencia:** avisa la distancia solo con sensor (robot), lógica de rangos (prog) y montaje (diseño).

6. Dispensador de comida para mascota · **ODS 15** Reto: alimentar a una mascota a horas fijas. · Prog: temporización + condicional / Robot: servo dosificador / Diseño: tolva y base (3D/ láser). · **Convergencia:** dosifica la ración solo con servo (robot), lógica (prog) y tolva bien diseñada (diseño).

7. Caja fuerte con clave · **ODS 16** Reto: proteger algo con una clave. · Prog: comparación de clave (condicional) / Robot: teclado + servo cerrojo / Diseño: caja. · **Convergencia:** abre solo con clave correcta (prog) + cerrojo (robot) + caja resistente (diseño).

8. Lavamanos automático · **ODS 3** / **6** Reto: higiene sin desperdiciar agua. · Prog: condicional con sensor / Robot: ultrasónico + bomba/servo / Diseño: grifo y base. · **Convergencia:** sale agua solo al detectar manos (robot+prog) en un grifo usable (diseño).

9. Mini-invernadero con ventilación · **ODS 13** Reto: mantener fresco un cultivo. · Prog: condicional con temperatura / Robot: sensor temp + ventilador / Diseño: estructura del invernadero. · **Convergencia:** ventila al subir la temperatura solo con sensor (robot), lógica (prog) y estructura (diseño).

Nivel 3 · Inventores (6° – 8°)

Raspberry Pi Pico + MicroPython · funciones, listas, datos · BIPES (puente), Thonny, Fritzing, FreeCAD

1. Estación meteorológica ★ · **ODS 13** Reto: medir y entender el clima local. · Prog: funciones + registro de datos / Robot: sensores temp/humedad + OLED / Diseño: carcasa intemperie. · **Convergencia:** muestra y registra el clima solo con sensores (robot), funciones (prog) y carcasa (diseño).

2. Invernadero autónomo ★ · **ODS 2** / **12** Reto: cultivar con menos recursos. · Prog: funciones + condicionales / Robot: múltiples sensores + actuadores / Diseño: modelado 3D + interfaz OLED.

3. Contador de aforo ★ · **ODS 11** Reto: saber cuántas personas usan un espacio. · Prog: funciones + listas (conteo) / Robot: sensor de paso / Diseño: caja y ubicación.

4. Monitor de calidad del aire del salón · **ODS 3** Reto: saber si el aire del aula es sano. · Prog: lectura + umbrales + alerta / Robot: sensor de aire/polvo + OLED / Diseño: estación de

escritorio. · **Convergencia:** alerta el aire malo solo con sensor (robot), lógica de umbral (prog) y una estación legible (diseño).

5. Medidor de consumo de agua · **ODS 6** Reto: visualizar cuánta agua usamos. · Prog: funciones + acumulado de litros / Robot: sensor de flujo + OLED / Diseño: módulo para el grifo. · **Convergencia:** muestra el consumo solo con sensor de flujo (robot), cálculo (prog) y montaje (diseño).

6. Alerta temprana de inundación · **ODS 11** Reto: avisar cuando sube el nivel del agua. · Prog: condicional + registro / Robot: sensor de nivel + alarma / Diseño: boya/soporte. · **Convergencia:** alerta a tiempo solo con sensor de nivel (robot), lógica (prog) y soporte resistente (diseño).

7. Seguidor solar · **ODS 7** Reto: captar mejor la energía del sol. · Prog: comparar luz y mover (funciones) / Robot: LDRs + servo / Diseño: estructura del panel. · **Convergencia:** sigue al sol solo con sensores (robot), lógica de comparación (prog) y estructura móvil (diseño).

8. Clasificador de residuos con sensores (+ML) · **ODS 12** Reto: separar residuos automáticamente. · Prog: lógica + (ML opcional) / Robot: sensores + servo compuerta / Diseño: contenedor. · **Convergencia:** clasifica solo con detección (robot), decisión (prog/ML) y compuerta diseñada (diseño).

9. Recordatorio de hábitos saludables · **ODS 3** Reto: recordar tomar agua, moverse, postura. · Prog: temporización + listas de mensajes / Robot: OLED + zumbador / Diseño: objeto de escritorio. · **Convergencia:** recuerda a tiempo solo con la lógica (prog), la salida (robot) y un objeto que invite a usarlo (diseño).

Nivel 4 · Innovadores (9° – 11°)

ESP32 + WiFi/BLE/MQTT · Python/C++ · web (HTML/CSS/JS/Bootstrap) + SQL + dashboard

1. Hogar inteligente (domótica) ★ · **ODS 7** / **11** Reto: una vivienda que ahorra energía. · Prog: clases + WiFi + control / Robot: relés/sensores conectados / Diseño: UI de control (app/ dashboard).

2. Monitoreo ambiental IoT con dashboard ★ · **ODS 13** / **11** Reto: vigilar el ambiente de la comunidad. · Prog: MQTT + SQL + dashboard / Robot: sensores en red / Diseño: UI/UX del tablero.

3. Sistema de alerta conectado ★ · **ODS 11** Reto: avisar a tiempo ante un riesgo. · Prog: eventos de red + notificaciones / Robot: sensores + conectividad / Diseño: interfaz de alertas.

4. Red comunitaria de calidad del aire · **ODS 3** / **11** Reto: mapear el aire de varios puntos. · Prog: varios nodos → MQTT → dashboard / Robot: nodos ESP32 con sensores / Diseño: visualización de mapa y datos. · **Convergencia:** el mapa de aire existe solo con nodos en red (robot), backend/datos (prog) y un dashboard claro (diseño).

5. Gestión inteligente de agua (finca/huerta) · **ODS 6** / **2** Reto: regar una zona grande con datos. · Prog: control + base de datos / Robot: válvulas y sensores conectados / Diseño: panel de control. · **Convergencia:** la zona se riega óptimamente solo con actuadores en red (robot), lógica con datos (prog) y un panel usable (diseño).

6. Medidor de energía del hogar + ahorro · **ODS 7** / **12** Reto: ver y bajar el consumo eléctrico. · Prog: medición + dashboard + recomendaciones / Robot: sensor de corriente ESP32 / Diseño: dashboard de ahorro. · **Convergencia:** se ahorra energía solo si se mide (robot), se procesa y recomienda (prog) y el dashboard comunica (diseño).

7. Control de acceso con base de datos · **ODS 4** / **16** Reto: registrar entradas/salidas escolares. · Prog: RFID + SQL + reportes / Robot: lector + cerrojo / Diseño: interfaz de administración. · **Convergencia:** el registro confiable existe solo con lector/cerrojo (robot), base de datos (prog) y una interfaz administrable (diseño).

8. Alerta climática temprana (+ML) · **ODS 13** Reto: anticipar un evento climático. · Prog: datos + modelo de predicción (ML) / Robot: estación multisensor conectada / Diseño: tablero de alertas. · **Convergencia:** la alerta anticipada existe solo con la estación (robot), el modelo con datos (prog/ML) y un tablero que avisa (diseño).

9. Plataforma de huerta urbana conectada · **ODS 2** / **11** Reto: una huerta comunitaria monitoreada. · Prog: sensores → app/web + base de datos / Robot: nodos de sensado y riego / Diseño: app comunitaria y branding. · **Convergencia:** la huerta conectada funciona solo con los nodos (robot), la plataforma (prog) y una experiencia que la comunidad use (diseño).

Cómo se usa este repositorio

- **Diversidad de oferta:** el colegio elige sus 1–3 proyectos por nivel según sus ODS prioritarios y su contexto.
- **Misma vara:** todo proyecto elegido se documenta con la ficha de convergencia (`06`) y se evalúa con la rúbrica (`04`).
- **Fabricación propia:** la mayoría son fabricables con corte láser e impresión 3D de ROBOTSchool — clave para los kits (fase siguiente).

Pendiente

1. Desarrollar la **ficha de convergencia completa** de cada proyecto seleccionado (como en `06`).
2. Asociar a cada proyecto su **lista de materiales/kit** y costo (fase de kits y libros).

Fin del documento v0.1.

Línea Ares — Esquema Maestro de los Libros (el continuo del currículo)

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> El esquema de los **12 libros** (uno por grado, de Transición a 11°). Puesto en fila, hace **visible el continuo**: la programación nunca se reinicia, los elementos (comodines) se **acumulan**, la placa cambia **solo tres veces** —siempre con un puente— y el diseño y la IA escalan. Cada libro está anclado a un **ODS** y termina en un **prototipo de solución**.

1. Los 12 libros de un vistazo

Grado	Banda	Libro (título · ODS)	Placa	Proyecto final
Transición	Exploradores	*El mundo que responde* · ODS 4	Makey Makey	Piano de frutas
1°	Exploradores	*Mi planeta sin basura* · ODS 12	Makey Makey	Juego clasificador de residuos
2°	Exploradores	*La fuerza del viento* · ODS 7	Makey + circuito	Molino con motor
3°	Constructores	*Ciudad que ordena* · ODS 11	**Arduino**	Semáforo inteligente
4°	Constructores	*Ni una gota de más* · ODS 6	Arduino	Riego automático
5°	Constructores	*Manos que ayudan* · ODS 9	Arduino	Brazo / mano robótica
6°	Inventores	*Leer el cielo* · ODS 13	**Raspberry Pi Pico**	Estación meteorológica
7°	Inventores	*El huerto que se cuida solo* · ODS 2	Pico	Invernadero autónomo
8°	Inventores	*Datos que clasifican* · ODS 12	Pico	Clasificador con sensores (+ML)
9°	Innovadores	*Casa que ahorra* · ODS 7	**ESP32**	Hogar inteligente (domótica)
10°	Innovadores	*El aire de mi barrio* · ODS 11	ESP32	Monitoreo ambiental IoT + dashboard
11°	Innovadores	*Comunidad conectada* · ODS 2 / 11	ESP32	Huerta urbana conectada (proyecto integrador)

2. Matriz del continuo (qué es **nuevo** en cada grado)

Lo nuevo se aprende **completo**; lo de grados anteriores se **repasa** (currículo en espiral). Por eso cada fila solo suma.

Grado	Programación (lo nuevo)	Elementos nuevos (comodines)	Diseño (énfasis)	IA
Trans.	Secuencia, eventos	`COM-PL-MK`, contacto/ tacto	Boceto y decoración	¿Qué hace una máquina?
1°	Comparar (acierto/ error)	(lógica en Scratch)	Íconos y categorías	La máquina “reconoce”
2°	Secuencia con salida	`COM-BS-CIR`, `COM-AC-MDC` (motor), `COM-BS-MEC`	Estructura y mecanismo	—
3°	**Bucles, condicionales**	`COM-PL-ARD`, `COM-AC-LED`, `COM-SE-PUL`, `COM-BS-PROG`	Maqueta + **esquemático** (Fritzing)	Tiempos “inteligentes” (charla)
4°	**Variables, umbral**	`COM-SE-HUM` (analógico), `COM-AC-BMB`, `COM-MO-REL`	Carcasa **3D** (Tinkercad)	Clasificar con ejemplos
5°	**Funciones** (asomada), C++ asomada	`COM-AC-SRV` (servo/ PWM)	Piezas 3D **forma-función**	Prótesis y sensores (charla)
6°	**MicroPython** (puente **BIPES**), lectura periódica	`COM-PL-PIC`, `COM-SE-DHT`, `COM-MO-OLED`	Carcasa intemperie	Datos como “historia”
7°	Lógica multi-sensor, **lazo cerrado**	(reúso de sensores) + actuadores	**Modelado 3D propio**	Decidir con datos
8°	**Listas, registro de datos**	`COM-SE-ULT` / `COM-SE-PIR`	Interfaz OLED + datos	**ML no-code** (Teachable Machine)
9°	**Clases**, WiFi, control	`COM-PL-ESP`, `COM-MO-WIFI` (+relé reúso)	**UI** de app de control	Recomendaciones con datos
10°	**Concurrencia, MQTT, SQL**, web	`COM-SE-GAS`, stack web (HTML/CSS/JS)	**UI/UX** de dashboard	Alertas/predicción
11°	Sistema completo, integración	Red multi-nodo, **ML** (Edge Impulse)	**Diseño de sistema** + branding	ML + ética de datos

3. La lectura del continuo (qué se hace **notorio**)

Al leer la matriz de arriba hacia abajo, se ven **siete hilos que nunca se cortan**:

1. **La programación no se reinicia: se acumula.** Secuencia y eventos (Exploradores) → bucles, condicionales y variables (Constructores) → funciones, listas y datos (Inventores) → clases, concurrencia y SQL (Innovadores). Lo de un grado sigue vivo en el siguiente.
2. **Los elementos se acumulan, no se desechan.** El LED de 3° reaparece como señal en casi todos los proyectos; el relé de 4° vuelve en 9° (domótica); el sensor de humedad de 4° regresa en el invernadero de 7°. Cada comodín se reutiliza.
3. **La placa cambia solo tres veces** —Transición→3° (Makey→Arduino), 5°→6° (Arduino→Pico), 8°→9° (Pico→ESP32)— y **siempre con un puente**: C++ “asomada” antes de Pico, BIPES (bloques→MicroPython) en 6°. El lenguaje cambia; la lógica, no.
4. **El diseño escala:** decorar → dibujar el esquemático → modelar en 3D → diseñar la UI/UX de un sistema.
5. **La IA madura:** de “¿qué hace una máquina?” a entrenar modelos (ML) y discutir la ética de los datos.
6. **Los ODS diversifican**, pero el patrón es constante: el proyecto final **siempre** es un prototipo de solución a una pregunta real.
7. **La convergencia de los tres hilos está en los 12 libros**; lo que cambia es la **ambición** del punto de convergencia, no el método.

> **Conclusión:** un estudiante que recorre los 12 libros no estudia 12 temas sueltos; recorre **un solo hilo** que se engrosa cada año. Esa es la promesa de Ares hecha currículo.

---

4. Cómo se relaciona con el resto

- Cada libro detalla sus microproyectos según el **modelo editorial** (`11`) y los cuatro **libros modelo** (`12`).
- Cada libro **llama los comodines** de los elementos que usa (`13`).
- Cada proyecto final se evalúa con la **rúbrica de convergencia** (`04`) y su **punto de convergencia** (`06` , `09`).
- Los proyectos salen del **repositorio** (`07`); aquí se eligió **uno por grado** como libro oficial (los demás quedan como alternativas).

5. Pendiente

1. Confirmar la **asignación ODS↔grado** con el equipo (es la propuesta base; se puede ajustar por contexto del colegio).
2. Desarrollar los **12 libros** con la estructura del libro de ejemplo.
3. Producir los **comodines** de todos los elementos listados y los **kits** por grado.

Fin del documento v0.1.

Línea Ares — Rúbrica Integrada de Convergencia

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Esta rúbrica evalúa lo que hace única a Ares: **que los tres hilos (programación, robotización y diseño) funcionen juntos**, no por separado. Por eso, además de calificar cada hilo, tiene un criterio dedicado —y de mayor peso— a la **convergencia**. Se usa igual en los cuatro niveles; lo único que cambia por nivel es *qué se espera* de cada hilo (eso lo dan las tablas de maduración de `01…§3-4` y `03…§1-2`).

---

1. La idea en una frase

> No basta con que el código funcione, el robot se mueva y el prototipo se vea bonito **por separado**. La pregunta clave es: **¿se necesitan los tres para lograr la tarea del proyecto?** Si quitas un hilo y el proyecto sigue funcionando igual, ese hilo estaba "pegado", no integrado.

A esa pregunta la llamamos la **prueba de "quita un hilo"**, y es el corazón de la evaluación.

---

2. Escala de desempeño

Cuatro niveles, iguales para todos los criterios:

Nivel	Nombre	Valor
1	Inicial	Aún no se logra
2	En desarrollo	Se logra con ayuda o a medias
3	Logrado	Se logra de forma autónoma
4	Destacado	Se logra y se explica/supera

---

3. Criterios y pesos

#	Criterio	Qué mira	Peso
1	**Hebra de programación**	La lógica funciona y usa las estructuras esperadas del nivel	15%
2	**Hebra de robotización**	El sistema físico sensa y actúa de forma confiable	15%
3	**Hebra de diseño**	Forma, esquemático y acabado pertinentes y pensados para el usuario	15%
4	**Convergencia (integración)**	Los tres hilos funcionan **juntos** para cumplir la tarea integradora	**30%**
5	**Buen uso de la IA**	Criterio al usar IA (cuando el proyecto la incluye)	10%
6	**Proceso y comunicación**	Documenta, itera y explica/ defiende su proyecto	15%

> La **convergencia pesa el doble** que cualquier hilo individual: es lo que distingue a Ares. Los pesos son ajustables por el equipo, pero la convergencia debe seguir siendo el criterio dominante.

---

4. Descriptores por criterio

4.1 Convergencia (el criterio central)

1 · Inicial	2 · En desarrollo	3 · Logrado	4 · Destacado
Las tres partes existen pero no se conectan; el proyecto no cumple la tarea integradora	Dos hilos se integran; el tercero está “pegado” o es decorativo (p. ej., el diseño solo adorna)	Los tres hilos funcionan juntos y el proyecto cumple la tarea integradora	La integración es fluida y el estudiante **demuestra que cada hilo es necesario** (pasa la prueba de “quita un hilo”) y lo explica

4.2 Hebra de programación

1 · Inicial	2 · En desarrollo	3 · Logrado	4 · Destacado
La lógica no funciona o no usa las estructuras del nivel	Funciona a medias o requiere ayuda constante	Funciona y usa correctamente las estructuras esperadas del nivel	Lógica eficiente; el estudiante la explica y depura solo

4.3 Hebra de robotización

1 · Inicial	2 · En desarrollo	3 · Logrado	4 · Destacado
El sistema no sensa/ actúa de forma confiable	Funciona de forma intermitente o mal conectado	Sensa y actúa de forma confiable según el diseño	Sistema robusto, bien cableado y calibrado; diagnostica sus fallos

4.4 Hebra de diseño

1 · Inicial	2 · En desarrollo	3 · Logrado	4 · Destacado
Sin diseño intencional (forma/esquemático/ acabado ausentes)	Diseño parcial; descuida al usuario o la fabricación	Diseño adecuado en los sub-hilos pertinentes y pensado para el usuario	Diseño cuidado, fabricación propia limpia, identidad/UX clara e iterada

4.5 Buen uso de la IA *(si el proyecto la incluye; si no, no se califica)*

1 · Inicial	2 · En desarrollo	3 · Logrado	4 · Destacado
Usa la IA sin criterio o para que “haga la tarea por él”	La usa como herramienta pero sin entender ni verificar	La usa para amplificar su trabajo y verifica los resultados	La usa con criterio y reflexiona sobre datos, sesgos y cuándo *no* usarla

4.6 Proceso y comunicación

1 · Inicial	2 · En desarrollo	3 · Logrado	4 · Destacado
Sin evidencia de proceso; no explica	Proceso parcial; explicación confusa	Documenta, itera y explica su proyecto	Bitácora clara, varias iteraciones justificadas y defensa convincente

5. Cómo se adapta por nivel

La rúbrica es **la misma** en los cuatro niveles. Lo que cambia es la *vara* de los criterios 1–3, que se toma de las tablas de maduración:

- En **Constructores**, “estructuras esperadas” = bucles, condicionales y variables (en bloques).
- En **Inventores**, “estructuras esperadas” = funciones, listas y registro de datos (MicroPython).
- En **Innovadores**, = clases, concurrencia y SQL.

Los criterios 4 (convergencia), 5 (IA) y 6 (proceso) **no cambian** de un nivel a otro: lo que sube es la ambición del proyecto, no la rúbrica.

6. Ejemplo aplicado — “Riego automático” (Nivel 2 · Constructores)

Tarea integradora (punto de convergencia): *que la maceta se riegue sola justo cuando le hace falta.*

Criterio	Evidencia esperada	Desempeño (ejemplo)
Programación	Condicional con umbral de humedad + variable	3 · Logrado
Robotización	Sensor de humedad + bomba/servo; cableado confiable	3 · Logrado
Diseño	Carcasa que protege el circuito del agua; usable	3 · Logrado
Convergencia	Riega solo cuando hace falta; **quita el sensor y no sabe cuándo; quita el condicional y no decide; quita la carcasa y el agua daña el circuito**	4 · Destacado
IA	Clasificó “planta sana/seca” con ejemplos y verificó	3 · Logrado
Proceso	Bitácora con 2 iteraciones (ajuste de umbral) y explicación	3 · Logrado

Cómo se lee: el proyecto logra los tres hilos (3) y destaca en convergencia (4) porque el equipo puede mostrar que **ninguna de las tres partes sobra**. Ese es el resultado que Ares busca.

---

7. Cómo usarla en el aula

- **Autoevaluación y coevaluación:** el estudiante (o el equipo) se califica antes que el docente; luego se contrasta.
- **La prueba de "quita un hilo"** se hace en voz alta en la socialización: *"¿qué pasaría si tu proyecto no tuviera diseño / programación / robótica?"*. Si el estudiante no sabe responder, la convergencia aún no está lograda.
- **Evidencia:** el proyecto funcionando + la bitácora + la defensa oral.

---

8. Pendiente

1. Redactar el **punto de convergencia** (la tarea integradora) de cada proyecto del mapa, empezando por un nivel completo para ajustar el formato.
2. Versión imprimible de la rúbrica (1 página) para el docente.

Fin del documento v0.1. Se aplica sobre `02_mapa_curricular_ares.md` *y* `03…§4-5`*.*

Línea Ares — Puntos de Convergencia · Nivel Constructores (3°–5°)

ROBOTSchool · Documento maestro (fuente única) — NIVEL MODELO Versión 0.1 — Junio 2026

> Este documento desarrolla a fondo los **tres proyectos ancla de Constructores** como **modelo a ajustar**. Una vez validado el formato, se replica en los otros tres niveles. Cada ficha sigue la plantilla de `03…§5` y se evalúa con la rúbrica de `04`. El elemento central de cada ficha es el **punto de convergencia**: una frase verificable que prueba que el proyecto necesita los tres hilos.
>
> **Estructuras esperadas del nivel** (la "vara" de la rúbrica): bucles, condicionales y variables, en bloques (mBlock), con C++ como *asomada*. **Herramientas del nivel:** mBlock · Tinkercad 3D · Tinkercad Circuits / Fritzing · Canva.

Cómo leer una ficha

Cada proyecto trae: **reto y ODS**, las **tres hebras** (programación / robotización / diseño) y la **capa de IA**, el **punto de convergencia** (la frase verificable), la **prueba de "quita un hilo"** y la **evidencia** de evaluación.

Proyecto 1 · Semáforo inteligente

Reto: ¿Cómo ordena el tráfico una ciudad y cómo lo hacemos más seguro para el peatón? · **ODS 11 — Ciudades sostenibles**

Hilo	Hebra activada en este proyecto
Programación	**Bucle** que repite el ciclo del semáforo + **condicional** (si el peatón pulsa, cambia) + **variables** (tiempos de cada luz)
Robotización	**Actuación** (3 LEDs por semáforo) · **sensado** (pulsador peatonal) · **control** (entrada → lógica → salida)
Diseño	**A** maqueta de la intersección (cartón o piezas 3D) · **B** esquemático del circuito de LEDs y botón (Tinkercad Circuits / Fritzing) · **C** señalización y acabado de las calles (Canva)
IA (opcional)	Conversación: ¿cómo decidiría sus tiempos un semáforo *de verdad* inteligente según el tráfico? (se discute, no se implementa)

Punto de convergencia (verificable): > *El semáforo regula un cruce con peatón **solo si**: la lógica controla la secuencia y atiende la petición del botón (programación) **+** los LEDs y el pulsador funcionan y están bien cableados (robotización) **+** la maqueta representa una intersección comprensible y segura (diseño).*

Prueba de "quita un hilo": - Sin **programación**: las luces no siguen secuencia ni responden al peatón. - Sin **robotización**: no hay semáforo físico que encienda ni botón que registre. - Sin **diseño**: sin la maqueta y la señalización no se entiende qué cruce es ni para quién.

Evidencia de evaluación: maqueta funcionando + esquemático del circuito + explicación del ciclo y de los tiempos elegidos.

Proyecto 2 · Brazo / mano robótica

Reto: Imitar cómo agarra la mano humana para mover un objeto. · **ODS 9 — Industria, innovación e infraestructura**

Hilo	Hebra activada en este proyecto
Programación	**Condicionales** + **variables** (ángulos del servo) + **bucle** para la secuencia de movimiento (agarrar → mover → soltar)
Robotización	**Actuación** (servomotores) · **mecanismo** (dedos/articulaciones, transmisión) · **control** de posición
Diseño	**A** modelado de las piezas del brazo y los dedos para imprimir/cortar (Tinkercad 3D), con **forma–función** (que de verdad agarre) · **B** esquemático servo–Arduino · **C** acabado
IA (opcional)	Conversación sobre prótesis: ¿cómo un sensor podría hacer que la mano reaccione sola?

Punto de convergencia (verificable): > *El brazo toma y suelta un objeto a la orden **solo si**: la lógica comanda los ángulos en la secuencia correcta (programación) **+** los servos y la estructura articulada mueven y sostienen (robotización) **+** las piezas están diseñadas con la forma que permite agarrar (diseño).*

Prueba de "quita un hilo": - Sin **programación**: no hay secuencia de agarre; el servo no sabe cuándo ni cuánto girar. - Sin **robotización**: sin servos ni estructura articulada no hay movimiento ni sujeción. - Sin **diseño**: si la forma de los dedos/superficie es incorrecta, el objeto resbala y no se agarra.

Evidencia de evaluación: el brazo agarrando un objeto + las piezas diseñadas (archivo 3D) + esquemático del conexionado.

Proyecto 3 · Riego automático

Reto: Cuidar una planta sin desperdiciar agua. · **ODS 6 — Agua limpia** / **ODS 12 — Producción y consumo responsables**

Hilo	Hebra activada en este proyecto
Programación	**Condicional** con umbral de humedad + **variable** que guarda la lectura
Robotización	**Sensado** (sensor de humedad) · **actuación** (bomba o servo) · **control** (entrada → lógica → salida)
Diseño	**A** soporte/carcasa para la electrónica (Tinkercad 3D) · **B** esquemático sensor–bomba (Fritzing) · **C** acabado usable
IA (opcional · extensión ML)	Entrenar un clasificador “planta sana / seca” con ejemplos (Teachable Machine) y verificar su acierto

Punto de convergencia (verificable): > *La maceta se riega sola justo cuando le hace falta* ***solo si****: el condicional decide según la humedad (programación)* ***+*** *el sensor mide y la bomba riega (robotización)* ***+*** *la carcasa protege el circuito del agua y es usable (diseño).*

Prueba de “quita un hilo”: - Sin **programación**: no decide cuándo regar. - Sin **robotización**: no mide la humedad ni echa el agua. - Sin **diseño**: el agua daña el circuito y el usuario no puede manipularlo con seguridad.

Evidencia de evaluación: sistema regando según humedad + esquemático + bitácora con el ajuste del umbral (al menos 2 iteraciones).

Nota de ajuste (para tu revisión)

Antes de replicar este formato en Exploradores, Inventores e Innovadores, conviene que valides: 1. **¿La frase del punto de convergencia es clara y verificable** para un docente que la lee por primera vez? 2. **¿La “prueba de quita un hilo” funciona** como pregunta de socialización con los estudiantes? 3. **¿El nivel de detalle es el correcto**, o prefieres más/menos (p. ej. añadir tiempos, materiales o criterios de logro por hebra)?

Con tu visto bueno o tus ajustes, replico el mismo esquema en los otros tres niveles (9 proyectos restantes).

Fin del documento v0.1. Nivel modelo — se aplica sobre `02`, `03…§4-5` *y* `04`.

Línea Ares — Puntos de Convergencia · Exploradores, Inventores e Innovadores

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Completa las fichas de convergencia de los **proyectos ancla** de los tres niveles restantes, con el mismo formato del nivel modelo (`06_convergencia_constructores.md`). Con esto quedan documentados los **12 proyectos ancla** (3 por nivel). Cada ficha trae la **tarea integradora verificable**, las tres hebras + IA, la **prueba de "quita un hilo"** y la evidencia. Se evalúan con la rúbrica de `04`. El resto del repositorio (`07`) se desarrolla bajo demanda.

Nivel 1 · Exploradores (Transición – 2°)

Makey Makey + mecánica · Scratch (secuencia, eventos). Diseño del nivel: boceto/Scratch, circuito simple, decoración.

Proyecto 1 · Piano de frutas

Reto: ¿Cómo hacemos música tocando alimentos? · **ODS 4 — Educación de calidad**

Hilo	Hebra activada
Programación	**Eventos**: “al tocar esta tecla, suena esta nota”; secuencia de sonidos
Robotización	**Energía/circuito** (Makey Makey cierra el circuito por el cuerpo) · **sensado** por tacto · **conductividad**
Diseño	**C** disposición y decoración del teclado (qué fruta es qué nota, claro y atractivo) · **A** boceto previo
IA (opcional)	Conversación: ¿cómo “escucha” un asistente de voz? (la máquina detecta señales)

Punto de convergencia: *El piano suena la nota correcta al tocar una fruta **solo si**: el evento programado asigna la nota (programación) **+** el contacto cierra el circuito conductor (robotización) **+** el teclado está dispuesto de forma comprensible y atractiva (diseño).*

Prueba de “quita un hilo”: sin **programación**, tocar no produce la nota asignada; sin **robotización**, no se detecta el toque; sin **diseño**, no se sabe qué fruta es qué nota ni invita a jugar.

Evidencia: el piano funcionando + el niño explica qué fruta es qué nota.

Proyecto 2 · Cuenta-cuentos interactivo

Reto: un cuento que suena y responde cuando lo tocamos. · **ODS 4 — Educación de calidad**

Hilo	Hebra activada
Programación	**Secuencia** de escenas + **eventos** al tocar zonas
Robotización	**Sensado** táctil (Makey) · salida de **sonido**
Diseño	**C** ilustración y narrativa del cuento · **A** boceto de páginas/escenas
IA (opcional)	Conversación sobre cuentos que “se leen solos”

Punto de convergencia: *El cuento avanza y suena al tocar la parte correcta **solo si**: la secuencia programada responde al toque (programación) **+** el contacto activa la zona (robotización) **+** las ilustraciones guían qué tocar y cuentan la historia (diseño).*

Prueba de “quita un hilo”: sin **programación**, no hay respuesta al toque; sin **robotización**, no se detecta el contacto; sin **diseño**, no se entiende la historia ni dónde tocar.

Evidencia: el cuento interactivo + el niño narra y muestra las interacciones.

Proyecto 3 · El carrito que avanza

Reto: construir algo que se mueva solo para una misión (llevar un mensaje de un punto a otro). · **ODS 11 — Ciudades y comunidades sostenibles**

Hilo	Hebra activada
Programación	Encender/apagar el motor (**secuencia** simple)
Robotización	**Energía/circuito** (pila–portapilas–motor) · **mecanismo** (ruedas, ejes) · **actuación** (gira)
Diseño	**A** forma del carrito (que cargue el objeto) · **C** decoración y la misión (para qué / para quién)
IA (opcional)	¿Podría moverse sin empujarlo? ¿Qué necesitaría “sentir”?

Punto de convergencia: *El carrito lleva un objeto de un punto a otro **solo si**: se activa el motor (programación) **+** el circuito y las ruedas lo hacen avanzar (robotización) **+** su forma carga el objeto y su misión se entiende (diseño).*

Prueba de “quita un hilo”: sin **programación**, no arranca; sin **robotización**, no se mueve; sin **diseño**, no carga el objeto ni tiene propósito.

Evidencia: el carrito avanzando con su carga + explicación de la misión.

Nivel 3 · Inventores (6° – 8°)

Raspberry Pi Pico + MicroPython (funciones, listas, datos) · BIPES (puente), Thonny, Fritzing, FreeCAD.

Proyecto 1 · Estación meteorológica

Reto: medir y entender el clima de nuestro entorno. · **ODS 13 — Acción por el clima**

Hilo	Hebra activada
Programación	**Funciones** (leer, mostrar, registrar) · **listas**/registro de datos · promedios
Robotización	**Sensado** (temperatura, humedad, presión) · salida **OLED** · lectura periódica
Diseño	**A** carcasa para intemperie (3D) · **B** esquemático Pico–sensores–OLED (Fritzing) · **C** lectura clara
IA (opcional · datos)	Detectar tendencia o clasificar "¿lloverá?" con los datos

Punto de convergencia: *La estación mide, muestra y registra el clima de forma autónoma **solo si**: las funciones leen y guardan los datos (programación) **+** los sensores y la pantalla funcionan (robotización) **+** la carcasa resiste la intemperie y la lectura es clara (diseño).*

Prueba de "quita un hilo": sin **programación**, no hay lectura ni registro; sin **robotización**, no mide ni muestra; sin **diseño**, el equipo no sobrevive afuera ni se entiende.

Evidencia: estación registrando datos varios días + bitácora + lectura de tendencias.

Proyecto 2 · Invernadero autónomo

Reto: cultivar algo manteniéndolo solo, con menos recursos. · **ODS 2 — Hambre cero** / **ODS 12 — Consumo responsable**

Hilo	Hebra activada
Programación	**Funciones** + **condicionales** con varios sensores · registro de datos
Robotización	**Múltiples sensores** (humedad, temperatura, luz) · **actuadores** (bomba, ventilador) · **lazo cerrado** (autonomía)
Diseño	**A** modelado 3D de la estructura · **B** esquemático multisensor · **C** interfaz OLED e identidad
IA (opcional)	Clasificar el estado de la planta o decidir con datos

Punto de convergencia: *El invernadero mantiene el cultivo por sí mismo **solo si**: la lógica modular decide según varios sensores (programación) **+** sensores y actuadores perciben y actúan (robotización) **+** la estructura y la interfaz hacen el sistema viable y legible (diseño).*

Prueba de "quita un hilo": sin **programación**, no decide; sin **robotización**, no percibe ni actúa; sin **diseño**, no es un invernadero usable.

Evidencia: el sistema manteniendo condiciones + datos + decisiones tomadas.

Proyecto 3 · Contador de aforo

Reto: saber cuántas personas usan un espacio para gestionarlo mejor. · **ODS 11 — Ciudades y comunidades sostenibles**

Hilo	Hebra activada
Programación	**Funciones** + **listas**, conteo (incrementar/decrementar), mostrar
Robotización	**Sensor de paso** (ultrasónico/PIR) · salida **OLED**
Diseño	**A** caja y ubicación en el acceso · **B** esquemático · **C** indicación clara del conteo
IA (opcional · datos)	Analizar el aforo por hora/día

Punto de convergencia: *El sistema cuenta confiablemente el aforo **solo si**: la lógica de conteo registra entradas y salidas (programación) **+** el sensor detecta el paso (robotización) **+** la caja se instala bien en el acceso y muestra el dato (diseño).*

Prueba de "quita un hilo": sin **programación**, no cuenta; sin **robotización**, no detecta el paso; sin **diseño**, no funciona en la puerta ni comunica el dato.

Evidencia: conteo correcto en una prueba real + registro por intervalos.

Nivel 4 · Innovadores (9° – 11°)

ESP32 + WiFi/BLE/MQTT · Python/C++ · web (HTML/CSS/JS) + SQL + dashboard.

Proyecto 1 · Hogar inteligente (domótica)

Reto: una vivienda (maqueta) que ahorra energía y se controla a distancia. · **ODS 7 — Energía** / **ODS 11 — Comunidades**

Hilo	Hebra activada
Programación	**Clases/módulos** · **WiFi** y control · (consumo → dashboard, con anexo `08`)
Robotización	**Actuadores** (luces, clima) · **sensores** · **conectividad**
Diseño	**C** UI/UX de la app/dashboard de control · **A** maqueta de la casa · **B** esquemático
IA (opcional)	Recomendaciones de ahorro a partir de los datos

Punto de convergencia: *La vivienda se controla a distancia y ahorra energía **solo si**: el firmware conectado comanda y reporta (programación) **+** los actuadores y sensores ejecutan y miden (robotización) **+** una interfaz clara permite controlarla y entender el consumo (diseño).*

Prueba de "quita un hilo": sin **programación**, no hay control remoto; sin **robotización**, no hay casa que responda; sin **diseño**, nadie la controla ni ve el ahorro.

Evidencia: maqueta controlada por app vía WiFi + medición de consumo.

Proyecto 2 · Monitoreo ambiental IoT con dashboard

Reto: vigilar una variable ambiental de la comunidad y mostrarla a todos. · **ODS 13 — Clima** / **ODS 11 — Comunidades**

Hilo	Hebra activada
Programación	**MQTT/HTTP** · base **SQL** · **dashboard** web (HTML/CSS/JS)
Robotización	**Sensores** ambientales + ESP32 · **conectividad** (telemetría)
Diseño	**C** UI/UX del dashboard · **diseño de sistema** end-to-end
IA (opcional · datos/ML)	Predicción o alertas a partir de los datos

Punto de convergencia: *La comunidad ve el estado ambiental en tiempo real **solo si**: el dispositivo envía datos a una base y un tablero (programación) **+** los sensores conectados miden y transmiten (robotización) **+** el dashboard comunica de forma clara y útil (diseño).*

Prueba de "quita un hilo": sin **programación**, no hay base ni tablero; sin **robotización**, no hay datos que mostrar; sin **diseño**, el dato no comunica ni sirve para decidir.

Evidencia: sistema midiendo y publicando en un dashboard accesible + datos históricos.

Proyecto 3 · Sistema de alerta conectado

Reto: avisar a tiempo y a distancia ante un riesgo (humo, nivel de agua, intrusión). · **ODS 11 — Ciudades y comunidades sostenibles**

Hilo	Hebra activada
Programación	**Eventos de red** · notificaciones · lógica de **umbral**
Robotización	**Sensores** + ESP32 · **conectividad** (WiFi/BLE)
Diseño	**C** interfaz de alertas (app/mensaje) · **B** esquemático · **A** carcasa
IA (opcional)	Reducir falsas alarmas con datos

Punto de convergencia: *El sistema avisa a tiempo y a distancia ante un riesgo **solo si**: la lógica detecta el umbral y envía la alerta (programación) **+** el sensor conectado percibe el evento (robotización) **+** la alerta llega clara y oportuna al usuario (diseño).*

Prueba de "quita un hilo": sin **programación**, no hay aviso remoto; sin **robotización**, no se detecta el riesgo; sin **diseño**, la alerta no se entiende ni llega bien.

Evidencia: alerta real disparada y recibida a distancia + registro de eventos.

Cierre

Con `06` (Constructores) y este documento quedan los **12 proyectos ancla** con su punto de convergencia. Próximo paso sugerido: desarrollar fichas para los proyectos no-ancla del repositorio (`07`) **bajo demanda**, según los que el colegio elija, y luego la **lista de materiales**/**kit** de cada uno (fase de libros y kits).

Fin del documento v0.1.

Línea Ares — Extensiones Opcionales: Machine Learning y RA/RV

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Estas son **extensiones opcionales**, no parte del núcleo obligatorio. El núcleo de Ares son los **tres hilos** (programación, robotización y diseño) y su **convergencia**. Las extensiones se montan *encima* de ese núcleo en los niveles altos, y cada colegio las activa **según su infraestructura y la preparación de sus docentes**. Nunca reemplazan a los hilos ni se exigen para "completar" un nivel.

---

1. Principio: opcional y segmentado por madurez

ROBOTSchool ya ofrece de forma segmentada por tecnología; estas extensiones siguen la misma lógica, ahora por **madurez institucional**:

- Un colegio que inicia en robótica **no necesita** ML ni RA/RV: con los tres hilos tiene una propuesta completa.
- Un colegio con infraestructura (tablets, cámaras, buena conexión) y docentes preparados puede **activar** una o ambas extensiones como diferenciador.

> **Regla anti-adorno (mentor):** una extensión solo entra si pasa **la misma prueba de convergencia** que el núcleo — debe *servir a la tarea del proyecto*, no ser un demo aislado. Vender "IA y metaverso" como eslogan, sin propósito pedagógico, traiciona el modelo. Si el proyecto funciona igual sin la extensión, la extensión sobra.

---

2. Extensión A · Machine Learning

Qué es en Ares: que el estudiante **entrene** un modelo con los datos o ejemplos de su propio proyecto —y entienda qué hace—, en vez de usar la IA como una caja negra. Extiende la **capa transversal de IA** (`01…§5.1`) y se apoya en el hilo de programación/datos.

Progresión sugerida

Nivel	Qué hace el estudiante	Herramientas gratuitas
Inventores (6°–8°)	Clasificación sin código: enseñar a la máquina a reconocer imágenes, sonidos o datos del proyecto con ejemplos	**Teachable Machine** · **Machine Learning for Kids** (basado en Scratch)
Innovadores (9°–11°)	Integrar un modelo en el proyecto IoT (clasificar/predecir); intro a *edge ML* y visión; reflexión sobre datos y sesgos	**Edge Impulse** (plan gratuito de desarrollador) · extensiones de ML en Python

Requisitos

Cámara o micrófono (o un conjunto de datos), conexión a internet y criterio docente para conversar sobre **datos, sesgos y privacidad**.

Punto de convergencia con ML (ejemplo)

En el *invernadero autónomo* (Inventores), el modelo entrenado decide si la planta está sana **mirando** la cámara, y esa decisión dispara el riego. El ML no es un demo aparte: alimenta la tarea integradora del proyecto.

---

3. Extensión B · Realidad Aumentada y Virtual (RA/RV)

Qué es en Ares: usar RA/RV **solo cuando aporta valor real** — para *visualizar lo invisible* (cómo circula la corriente, cómo se mueve un mecanismo), *simular lo costoso o peligroso*, o *comunicar* el proyecto de forma inmersiva. Extiende el hilo de **diseño**, en su sub-hilo de **experiencia**.

Progresión sugerida

Nivel	Qué hace el estudiante	Herramientas gratuitas
Constructores / **Inventores**	Explorar y **crear escenas simples** para visualizar el mecanismo o el sistema	**CoSpaces Edu** (plan básico gratuito) · **Assemblr EDU** (plan básico) · **Merge Cube** (cubo físico + apps)
Innovadores	Crear **experiencias propias en web** (RA del producto o del dashboard, recorridos del sistema)	**A-Frame** / **WebXR** (libre, código abierto, en el navegador)

Requisitos

Tablets o celulares para RA; opcional visor/cardboard para RV; el **Merge Cube** es un accesorio físico de bajo costo. A-Frame/WebXR solo requiere navegador, así que es la vía de **menor barrera** para crear (no solo consumir).

Filtro de valor

Antes de usar RA/RV, responder: *"¿qué entiende mejor el estudiante por verlo así, que no entendería de otro modo?"*. Si no hay respuesta clara, no se usa.

3.5 Extensión C · Análisis y visualización de datos

Qué es en Ares: cerrar el ciclo del IoT — analizar, visualizar y **decidir** con los datos que genera el propio proyecto. Solo para **Innovadores**. Incluye hoja de cálculo avanzada y automatizada (Sheets/Excel, Power Query, Apps Script), dashboards (Looker Studio gratis, Power BI Desktop) y **análisis asistido por IA** (consultas en lenguaje natural). Extiende el manejo de datos del hilo de programación y se monta sobre la capa de IA.

> Detalle completo, herramientas, costos y evaluación en el anexo `08_anexo_datos_innovadores.md`. Regla de oro: el análisis cuenta como convergencia solo si lleva a una **decisión o mejora del proyecto**.

---

4. Cómo se ofrecen (segmentación por madurez institucional)

Madurez del colegio	Núcleo	Extensiones recomendadas
Inicial (primera vez en robótica)	Tres hilos + convergencia	Ninguna (propuesta ya completa)
Intermedia (infraestructura básica: tablets/cámaras)	Tres hilos + convergencia	ML sin código · RA con celular/tablet · datos con Sheets + Looker Studio
Avanzada (laboratorio, docentes preparados)	Tres hilos + convergencia	Edge ML · WebXR propio · Merge Cube · Power BI + análisis asistido por IA

---

5. Relación con el resto de la línea

- **ML** profundiza la capa de IA (`01…§5`) y el manejo de datos del hilo de programación (`01…§4`).
- **RA**/**RV** profundiza el sub-hilo **C (artístico/experiencia)** del diseño (`03…§2.C`).
- Ambas se evalúan con la **misma rúbrica de convergencia** (`04`): si la extensión no se integra a la tarea del proyecto, baja el criterio de convergencia en vez de subirlo.

6. Pendiente

1. Definir el **kit/infraestructura mínima** por extensión y su costo (enlaza con el hito de kits y logística).
2. Microcredencial o ruta de **capacitación docente** específica para cada extensión.

Fin del documento v0.1.

Línea Ares — Anexo opcional: Análisis y Visualización de Datos

ROBOTSchool · Documento maestro (fuente única) — ANEXO Versión 0.1 — Junio 2026

> Anexo **opcional**, solo para **Innovadores (9°–11°)**. Cierra el ciclo del IoT: los datos que generan los proyectos (estación meteorológica, medidor de energía, monitoreo ambiental) por fin se **analizan, se visualizan y se convierten en decisiones**. Sigue la misma lógica de las extensiones de `05`: opcional, segmentado por madurez y **anclado a los datos del propio proyecto** del estudiante — nunca a datasets abstractos. Se evalúa con la rúbrica de `04`.

---

1. Por qué y para quién

- **Cierra el ciclo de datos:** sensor → datos → SQL → dashboard → **análisis** → **decisión**. Le da el "¿y para qué?" a todo el IoT.
- **Solo Innovadores**, porque requiere madurez para razonar con datos; no baja a grados menores.
- **Habilidad empleable** y diferenciador comercial; conecta con los **ODS** (datos para decidir) y con el **buen uso de la IA** (análisis asistido).

> **Regla de oro:** se analizan **los datos reales del proyecto del estudiante**. Si el análisis no lleva a una **decisión o mejora del proyecto**, es un adorno y no cuenta como convergencia.

---

2. Los tres componentes del anexo

A · Hoja de cálculo avanzada y automatizada

De ordenar datos a **automatizar** su tratamiento. - Fórmulas avanzadas, tablas dinámicas, limpieza de datos. - **Automatización:** *Power Query* en Excel; **macros** / **Google Apps Script** para que las tablas se actualicen y se rehagan solas a partir del registro del proyecto. - Herramientas: **Google Sheets + Apps Script** (gratis) · **Excel** (licencia M365) con Power Query.

B · Dashboards y visualización (BI)

Conectar los datos del proyecto (CSV del *datalogger* o base SQL) y construir tableros que comuniquen. - **Looker Studio** — gratis, web, se comparte con un enlace. La vía de **menor barrera**. - **Power BI Desktop** — gratis para construir; *compartir en la nube exige Power BI Pro (US$14/usuario/mes)*. Conviene saberlo antes de prometerlo.

C · Análisis asistido por IA

- Consultas en **lenguaje natural** ya integradas en Looker Studio y Power BI (Copilot): *"¿qué día hubo más consumo?"*, *"muéstrame la tendencia de temperatura"*.
- **Criterio (clave):** el estudiante debe **verificar** lo que responde la IA y entender el dato, no delegar el juicio. Liga directo al eje de "IA bien usada" (`01…§5.1`).

---

3. Anclaje al proyecto (cómo se convierte en convergencia)

El anexo extiende el **punto de convergencia** del proyecto hacia la decisión:

Proyecto (Innovadores)	Dato que produce	Decisión basada en datos
Medidor de energía + ahorro	Consumo por hora/día	"A qué hora conviene apagar/encender para ahorrar"
Estación / monitoreo ambiental	Temperatura, aire, humedad	"Cuándo ventilar; alertar si supera un umbral"
Gestión inteligente de agua	Litros por zona y tiempo	"Ajustar el riego para gastar menos sin secar el cultivo"

> El análisis es exitoso cuando el estudiante **cambia algo de su proyecto** por lo que vio en los datos. Eso es "del dato a la decisión".

---

4. Herramientas y costos (claros)

Herramienta	¿Gratis?	Nota
Google Sheets + Apps Script	Sí	Automatización sin costo; ideal punto de partida
Looker Studio	Sí	Dashboards web, se comparten con enlace
Power BI Desktop	Sí (construir)	Compartir en la nube = **Pro, US$14/usuario/mes**
Excel + Power Query	No	Requiere licencia Microsoft 365
Python + pandas (Colab)	Sí	Vía de programación, opcional, para quien quiera automatizar con código

5. Segmentación por madurez (igual que `05`)

Madurez del colegio	Anexo de datos recomendado
Intermedia	Google Sheets + **Looker Studio** (todo gratis y web)
Avanzada	Power BI, automatización (Apps Script / Power Query) y análisis asistido por IA

6. Cómo se evalúa

- **Misma rúbrica (`04`):** el anexo **sube el criterio de convergencia** solo si el análisis sirve al proyecto (lleva a una decisión). Un dashboard bonito sin decisión **no** suma.

- **Foco propio del anexo:** *“del dato a la decisión”* — ¿el estudiante puede explicar qué cambió en su proyecto gracias a los datos?

> **Advertencia de mentor:** el riesgo es enseñar la *herramienta* (Power BI, fórmulas) en vez del *razonamiento* (leer datos y decidir). Se evalúa la decisión basada en datos, no lo vistoso del tablero.

---

7. Pendiente

1. **Plantilla “informe de datos del proyecto”** (1 página) para el estudiante.
2. **Conjuntos de datos de ejemplo** de cada proyecto IoT para practicar antes de tener los reales.
3. Enlazar con la **capacitación docente** (un docente que guíe BI) y con los **kits** (qué proyectos producen datos suficientes).

Fin del anexo v0.1. Complementa `05` *(extensiones opcionales) y se evalúa con* `04` *.*

Línea Ares — Plan de Capacitación Docente (modelo blended + Academy)

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> La capacitación docente es el **factor crítico** de Ares: la línea es abierta y por proyectos, no un libreto a seguir (como Ecua), así que exige docentes que **faciliten la convergencia**, no solo que sepan la herramienta. Este plan es **blended** (Academy + práctica presencial + acompañamiento en aula), **por bandas de nivel** y **just-in-time**. Academy (el LMS de ROBOTSchool) soporta videos, quizzes, **H5P**, cohortes y foros, certificados/insignias y **evaluación práctica**: el plan se diseña sobre esas capacidades.

---

1. Diagnóstico y principios

Punto de partida real: los docentes aliados dominan **Arduino y mBlock**. Ares les pide subir a MicroPython, ESP32, IoT, web/SQL, **además** de la pedagogía de convergencia y las extensiones opcionales. El salto es grande; por eso:

1. **Blended:** lo conceptual y la programación van async en Academy; el **armado, cableado, calibración y la facilitación** se entrenan presencialmente y en el aula. *Un LMS solo no forma para robótica.*
2. **Por banda de nivel:** se certifica por banda (Exploradores / Constructores / Inventores / Innovadores), no las cuatro de golpe.
3. **Just-in-time:** el docente se forma en la banda que va a dictar **este** año y suma la siguiente después.
4. **Pedagogía + técnica:** se evalúa que el docente sepa *facilitar* (ABP, convergencia, rúbrica), no solo que arme el circuito.
5. **Train-the-trainer:** formadores internos para escalar a muchos colegios.
6. **Seguridad primero:** manejo de herramientas, soldadura, corte láser e impresión 3D como módulo transversal obligatorio.

---

2. Perfil del docente Ares (competencias)

Dimensión	Qué debe poder hacer el docente
Técnica — programación	Programar al nivel de su banda (Scratch → mBlock → MicroPython/BIPES → Python/C++) y depurar con el estudiante
Técnica — robotización	Armar y diagnosticar circuitos, sensores y actuadores de su banda; leer un esquemático
Técnica — diseño	Usar las herramientas del nivel (Tinkercad, Fritzing, FreeCAD, Canva…) y orientar los tres sub-hilos de diseño
Pedagógica	Facilitar **ABP**; redactar/usar el **punto de convergencia**; aplicar la **rúbrica** y la prueba de “quita un hilo”
IA y datos	Modelar el **buen uso de la IA**; orientar las extensiones (ML, RA/RV, datos) cuando apliquen
Gestión y seguridad	Manejo seguro de herramientas y kits; gestión del aula-taller y del trabajo en equipo

3. Rutas de certificación (insignias en Academy)

Microcredenciales acumulables. Cada banda exige el **núcleo pedagógico** y la **banda anterior** (o equivalencia) como prerrequisito.

Insignia	Contenido técnico	Prerrequisito
Núcleo · Pedagogía de convergencia	ABP, los tres hilos, punto de convergencia, rúbrica, “quita un hilo”, uso de ODS/ISTE	— (base para todas)
Seguridad y fabricación	Herramientas, soldadura, corte láser, impresión 3D	— (transversal)
Docente Ares · Exploradores	Makey Makey, Scratch, mecánica y circuitos simples	Núcleo
Docente Ares · Constructores	Arduino/mBlock, sensores/actuadores, Tinkercad, Fritzing	Núcleo + Exploradores
Docente Ares · Inventores	Pico, MicroPython, **BIPES**, datos, FreeCAD	Núcleo + Constructores
Docente Ares · Innovadores	ESP32, IoT (WiFi/MQTT), web/SQL, dashboards	Núcleo + Inventores
Extensiones (opcionales)	ML · RA/RV · Análisis de datos (una insignia c/u)	La banda donde aplica
Formador Ares (train-the-trainer)	Formar y acompañar a otros docentes	2+ bandas + experiencia de aula

> **Por qué con prerrequisito de banda anterior:** un docente de Inventores debe entender de dónde viene el estudiante (qué ya sabe de Constructores) para no romper la continuidad del hilo. La progresión del docente refleja la del alumno.

---

4. Estructura blended de cada ruta

Cada banda combina tres momentos (proporción indicativa, ajustable):

Momento	Dónde	~Peso	Qué se hace
Async	Academy	40%	Videos, lecturas, **H5P** interactivos, quizzes, simuladores (Tinkercad/ Wokwi)
Taller práctico	Presencial / sincrónico	40%	Armar, cablear, calibrar y **construir el proyecto ancla** de la banda
Acompañamiento en aula	En el colegio	20%	Implementar con estudiantes, observación de un formador y retroalimentación

Evaluación para certificar (no es un quiz): el docente entrega en Academy una **evidencia práctica** — video del proyecto ancla funcionando + su **planeación** del proyecto + la **rúbrica de convergencia** aplicada a un caso. Se aprueba con una rúbrica de desempeño docente.

---

5. Mapa del curso en Academy (banda Constructores como modelo)

Así se monta **una** ruta en Academy; las demás siguen el mismo patrón.

Curso: Docente Ares · Constructores *(prerrequisito: Núcleo de convergencia + Seguridad)*

Módulo	Tipo	Recursos en Academy
0. Bienvenida y mapa del nivel	Async	Video intro + lectura del mapa curricular (`02`)
1. Electrónica y circuitos de la banda	Async + Taller	Video + **H5P** (clasificar componentes) + quiz de seguridad + taller de protoboard
2. Programación en mBlock (bucles, condicionales, variables)	Async + Taller	Video + **H5P** (arrastrar bloques / ordenar lógica) + reto en **Tinkercad Circuits**
3. Diseño: esquemático (Fritzing) y pieza 3D (Tinkercad)	Async + Taller	Tutoriales + entrega de un esquemático y una pieza
4. Pedagogía: el punto de convergencia del proyecto	Async	**H5P branching scenario** de una clase + foro de cohorte
5. Proyecto ancla integrador (semáforo / brazo / riego)	Taller	Construcción guiada del proyecto completo
6. Implementación en aula	Acompañamiento	Llevar el proyecto con estudiantes + observación
7. Evaluación práctica y certificación	Evaluación	**Subir evidencia** (video + planeación + rúbrica aplicada) → **insignia**

H5P sugeridos (aprovechando que Academy ya lo soporta): *drag-and-drop* de estructuras de programación, *image hotspots* sobre un circuito real, *branching scenarios* de decisiones de aula, *interactive video* de un montaje paso a paso.

---

6. Secuencia de adopción del docente (just-in-time)

1. **Onboarding:** Núcleo de convergencia + Seguridad y fabricación (todos los docentes).

2. **Banda asignada:** certifica la banda que dictará este año.
3. **Crecimiento:** suma la siguiente banda antes del año en que la necesite.
4. **Extensiones:** solo si su colegio activa ML, RA/RV o datos.

> No se exige que un docente domine las cuatro bandas. Un colegio puede tener un docente “Exploradores–Constructores” y otro “Inventores–Innovadores”. La especialización es válida y baja la barrera.

---

7. Train-the-trainer y comunidad de práctica

- **Formadores Ares** (certificados en 2+ bandas) imparten talleres y hacen el acompañamiento en aula.
- **Comunidad de práctica** en los foros de Academy por cohorte: dudas, evidencias, mejoras de proyectos.
- **Repositorio de evidencias**: los mejores montajes y planeaciones alimentan el banco de ejemplos (y la landing/marketing).

---

8. Mantenimiento de la certificación

- **Actualización** cuando cambia una herramienta o entra una extensión nueva (microcrédito corto en Academy).
- **Métricas** del programa: docentes certificados por banda, proyectos implementados por colegio, evidencias entregadas, retención.

---

9. Pendiente

1. **Cronograma y horas exactas** por banda (definir calendario real).
2. **Producción de contenidos** por módulo (guiones de video, H5P, quizzes, rúbrica docente).
3. **Catálogo de insignias** y reglas de emisión en Academy.
4. Enlace con **kits**: cada taller práctico necesita su kit de formación docente.

Fin del documento v0.1. Se apoya en `02`, `03`, `04` *y* `06`/`09`.

Línea Ares — Modelo Editorial de los Libros

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Define **cómo es un libro Ares**: su anatomía obligatoria, sus dos versiones (inglés y tropicalizada), el concepto **papel → MDF**, la plantilla de cada **microproyecto** y la **rúbrica con puntuación** que el docente llena dentro del libro. Es la plantilla común: todos los libros se construyen igual; cambian el ODS, el grado y el proyecto. **Un libro = un grado**, anclado a un ODS y a un proyecto final.

---

1. Las dos versiones de cada libro

Versión	Para quién	Idioma	Contexto del ODS
Internacional (EN)	Colegios bilingües / mercado internacional	Inglés	Datos y ejemplos globales/neutros
Tropicalizada	Colombia (base) — con opción para **Ecuador, Perú, Costa Rica y México**	Español	Datos, ejemplos, nombres y moneda **locales**

1.1 La "capa tropicalizable" (clave para producir barato)

Para que sacar cada país no sea rehacer el libro, separamos dos capas:

- **Núcleo fijo (idéntico en todas las versiones):** objetivos, microproyectos, guías de construcción/programación/diseño, plantillas recortables, rúbricas. *La técnica no cambia entre países.*
- **Capa de contexto (lo único que se cambia):** la explicación del ODS con **datos del país**, los ejemplos, los nombres propios, la moneda, y el **idioma** (en la versión EN).

> Producir una nueva tropicalización = traducir/ajustar **solo la capa de contexto**, no el libro entero. Editorialmente, esto debe estar marcado en el archivo fuente (p. ej., bloques etiquetados `[CONTEXTO-PAÍS]`).

---

2. Anatomía obligatoria del libro (en este orden)

1. **Portada** — título **ligado al ODS** del grado (evocador, no técnico).
2. **Identificación** — nombre del estudiante, grado, colegio, docente (para la evaluación).
3. **Índice claro** — con números de página.
4. **"El mundo y este ODS"** — explicación a fondo del ODS, con datos (capa tropicalizable).
5. **La pregunta que detona** — una pregunta abierta que conecta el ODS con la vida del estudiante.
6. **El reto: tu prototipo de solución** — qué construirá al final y **por qué responde** a la pregunta.
7. **Mapa del libro** — el **orden de ejecución de los tres ejes + la IA** (ver §3).
8. **Antes de empezar: tus comodines** — el **llamado** a las cartillas/recursos de conocimiento base que el grado necesita (ver `13`). El libro **no repite la teoría**: la referencia. Aquí se indica qué comodines usar y en qué profundidad (completo o repaso).
9. **Microproyectos** — varios, cada uno con su estructura (ver §4) y su rúbrica.
10. **Del papel al MDF** — construcción del **prototipo final** en MDF (fabricación ROBOTSchool).
11. **Socialización y reflexión** — vuelta a la pregunta detonante: ¿la resolvimos?
12. **Rúbrica final del proyecto** — de convergencia, con puntuación y nota del docente.
13. **Anexo recortable** — plantillas troqueladas (líneas de corte, doblez y pegado).
14. **Glosario y créditos.**

> **Comodines, no repetición:** la teoría base (qué es Arduino, sensores, actuadores…) **no se escribe en el libro**: se **llama** como comodín (`13`), disponible en cartilla impresa y recurso digital. El libro lleva los **microproyectos**; los comodines llevan el **conocimiento**.

> **Regla de oro de maquetación:** lo que el estudiante **recorta** vive en el **Anexo recortable (§13)**, separado de las guías, preguntas y rúbricas, que se **conservan**. Nunca poner una plantilla de recorte al respaldo de una página evaluable.

---

3. Orden de ejecución de los ejes (el flujo del libro)

Cada libro y cada microproyecto siguen la misma secuencia, para que el estudiante interiorice el método:

1. **Comprender** — el ODS y la pregunta.
2. **Diseñar** — bocetar la forma (eje Diseño · sub-hilo A).
3. **Construir** — armar la estructura y el circuito (eje Robotización + Diseño · sub-hilo B esquemático).
4. **Programar** — darle comportamiento (eje Programación).
5. **IA y datos (transversal)** — usar/criticar la IA cuando aporte; leer los datos.
6. **Integrar** — unir los tres hilos en el prototipo de **papel**.
7. **Fabricar** — pasar al prototipo final en **MDF**.
8. **Socializar y evaluar** — mostrar, responder la pregunta, calificar.

> El **acabado artístico** (Diseño · sub-hilo C) acompaña todo el proceso y se evalúa en la integración.

---

4. Plantilla de un microproyecto (estructura fija)

Cada microproyecto del libro tiene **siempre** estas secciones:

1. **Título y propósito** — qué se logra y cómo aporta al proyecto final.
2. **Objetivos** — 2 a 4, observables.
3. **Materiales** — divididos en:
   - *Del anexo recortable:* “Recorta la plantilla **A-x**”.
   - *Componentes:* LEDs, servos, sensores, módulos, cables, etc.
4. **Guía de construcción (papel)** — paso a paso: recortar por la línea, doblar, pegar, y **fijar el componente** en el punto marcado (con íconos de corte ✂, doblez ┄ y pegado).
5. **Guía de programación** — el código/bloques del nivel, comentado.
6. **Guía de diseño** — qué sub-hilos se trabajan (A modelado, B esquemático, C acabado) y cómo.
7. **Preguntas para responder en el libro** — espacios en blanco; mezcla de comprensión, predicción y reflexión.

8. **Evaluación del microproyecto** — **rúbrica con puntuación** que el docente llena (ver §5), con espacio para nota y firma.

> El microproyecto en papel es **prototipado barato**: el estudiante prueba y se equivoca en papel **antes** de gastar MDF y componentes en el prototipo final. Todos construyen, no solo el que tiene el kit.

---

5. Las rúbricas dentro del libro (con puntuación)

Hay **dos** niveles de evaluación impresos en el libro:

5.1 Rúbrica de microproyecto (formativa, rápida)

Tabla que el docente marca y suma. Escala 1–4 por criterio (de `04`), convertida a puntos:

Criterio	1	2	3	4	Puntos
Construcción (papel + componentes)					___ /4
Programación					___ /4
Diseño (forma/esquema/acabado)					___ /4
Preguntas del libro					___ /4
Trabajo y proceso					___ /4
				Total	**___ /20**

5.2 Rúbrica final del proyecto (de convergencia)

La rúbrica de `04` (la convergencia pesa el doble) con puntuación, aplicada al **prototipo final en MDF**. Incluye la **prueba de "quita un hilo"** y la **nota y firma del docente**.

> Así el libro **es** el registro de evaluación: el docente califica dentro de él, microproyecto por microproyecto, y al final el proyecto completo.

---

6. El concepto papel → MDF (y la fabricación propia)

- **Fase papel (en el libro):** plantillas troqueladas que el estudiante recorta, dobla, pega y equipa con componentes. Sirve para **entender la estructura** y probar el circuito/programa sin costo.
- **Fase MDF (kit ROBOTSchool):** el mismo prototipo, ahora cortado en **MDF con láser** (y piezas en 3D si aplica), robusto y definitivo. ROBOTSchool fabrica estas piezas; el libro y el kit están **atados al mismo proyecto**.
- Ventaja: el libro reduce el riesgo y el desperdicio antes de la fabricación, y conecta directo con la venta del kit.

---

7. Especificaciones de producción (para el equipo editorial)

- **Formato y troquelado:** definir tamaño, gramaje del anexo recortable (más grueso para que arme bien) y troquel.
- **Marcado del archivo fuente:** etiquetar bloques `[CONTEXTO-PAÍS]` vs núcleo fijo.
- **Códigos de plantilla:** numerar cada plantilla del anexo (A-1, A-2…) y referenciarla desde el microproyecto.
- **Versión EN:** derivar del núcleo + contexto neutro.

8. Pendiente

1. Definir formato físico, gramajes y troquel con producción.
2. Asociar a cada libro su **kit** (lista de materiales y costo) — fase de kits.
3. Plantilla maestra de maquetación (InDesign/editorial) a partir de esta anatomía.

Fin del documento v0.1. Los cuatro libros modelo (uno por nivel) están en `12`.

Línea Ares — Cuatro Libros Modelo (uno por nivel)

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Un **libro modelo por nivel**, siguiendo la anatomía de `11`. Cada uno corresponde a **un grado** (un libro = un grado), está anclado a un **ODS** y culmina en un **prototipo final en MDF**. Para cada libro se muestra: título, pregunta detonante, índice, orden de ejes, lista de microproyectos y el proyecto final; y se **desarrolla a fondo un microproyecto** con su plantilla y evaluación. Los demás grados de cada banda siguen el mismo molde con su propio ODS.
>
> *(Versión mostrada: tropicalizada Colombia. La versión EN y las de Ecuador/Perú/Costa Rica/México cambian solo la capa de contexto — ver `11…§1`.)*

Libro · Exploradores · 2° — ODS 12

Título: *"Mi planeta sin basura"*

ODS 12 · Producción y consumo responsables. Pregunta que detona: *¿Qué pasa con la basura que botamos, y cómo podemos ayudar a separarla?*

El reto (prototipo de solución): construir un **juego clasificador** que diga "¡sí!" cuando pones un residuo en su lugar correcto. Primero en papel, luego en madera (MDF).

Índice: 1) Nuestro planeta y la basura · 2) La pregunta · 3) Lo que vamos a construir · 4) Mapa del libro · 5) Microproyecto 1: El tablero que responde · 6) Microproyecto 2: Las canecas de colores · 7) Microproyecto 3: ¡El juego completo! · 8) Del papel a la madera · 9) Lo mostramos · 10) Mi nota · 11) Anexo recortable · 12) Palabras nuevas.

Orden de ejes: Comprender → Diseñar (decorar canecas) → Construir (circuito Makey) → Programar (Scratch: sonido al acertar) → Integrar (papel) → Madera → Mostrar.

Microproyectos: - **MP1 · El tablero que responde** — un toque correcto enciende/suena. *(desarrollado abajo)* - **MP2 · Las canecas de colores** — clasificar en orgánico, reciclable y no aprovechable. - **MP3 · ¡El juego completo!** — unir todo en el tablero de papel.

Proyecto final (MDF): tablero clasificador de residuos en madera, con Makey Makey y Scratch.

▶ Microproyecto 1 desarrollado · "El tablero que responde"

- **Objetivos:** (1) entender que un toque puede activar un sonido; (2) construir un punto conductor; (3) reconocer una secuencia "si toco → suena".
- **Materiales:** *Del anexo:* plantilla **A-1** (tablero base). *Componentes:* Makey Makey, papel aluminio, cinta, cocodrilos.
- **Construcción (papel):** recorta el tablero por la línea ✂; pega un trozo de aluminio en el círculo marcado; conecta el cocodrilo del Makey al aluminio.
- **Programación (Scratch):** al recibir el toque (tecla "espacio") → reproducir sonido "¡bien!".
- **Diseño:** decora tu tablero (sub-hilo C) y dibuja qué residuo va en el punto.
- **Preguntas en el libro:** ¿Qué pasó cuando tocaste el aluminio? ¿Por qué crees que suena? Dibuja otra cosa que te gustaría que respondiera al tocarla.
- **Evaluación:** rúbrica de microproyecto (`11…§5.1`), total /20, con nota y firma del docente.

Libro · Constructores · 4° — ODS 6

Título: *“Ni una gota de más”*

ODS 6 · Agua limpia y saneamiento. Pregunta que detona: *¿Cómo cuidamos una planta dándole exactamente el agua que necesita, sin desperdiciar?*

El reto (prototipo de solución): una **estación de riego** que mide la humedad de la tierra y riega **solo cuando hace falta**. En papel primero, luego en MDF.

Índice: 1) El agua, un recurso que se acaba · 2) La pregunta · 3) Lo que vamos a construir · 4) Mapa del libro · 5) MP1: El circuito que avisa · 6) MP2: El sensor y el umbral · 7) MP3: La bomba que riega · 8) MP4: La carcasa y la integración · 9) Del papel al MDF · 10) Socialización · 11) Rúbrica final · 12) Anexo recortable · 13) Glosario.

Orden de ejes: Comprender → Diseñar (boceto + esquemático) → Construir (sensor/actuador) → Programar (condicional con umbral) → IA (¿cómo decide una máquina “regar”?) → Integrar (papel) → MDF → Socializar.

Microproyectos: - **MP1 · El circuito que avisa** — un LED se enciende cuando la tierra está seca. - **MP2 · El sensor y el umbral** — leer la humedad y decidir con un condicional. *(desarrollado abajo)* - **MP3 · La bomba que riega** — activar la bomba/servo cuando se cruza el umbral. - **MP4 · La carcasa y la integración** — proteger el circuito y unir todo en papel.

Proyecto final (MDF): estación de riego en MDF con Arduino, sensor de humedad y bomba.

▶ Microproyecto 2 desarrollado · “El sensor y el umbral”

- **Objetivos:** (1) leer un sensor de humedad; (2) entender qué es un **umbral**; (3) escribir un **condicional** que decida; (4) diseñar el esquemático del circuito.
- **Materiales:** *Del anexo:* plantilla **B-2** (soporte del sensor). *Componentes:* Arduino, sensor de humedad, LED, protoboard, cables.
- **Construcción (papel):** recorta el soporte **B-2** ✂, dobla por ┄, pega y fija el sensor en la ranura marcada; conecta según el esquemático de la página.
- **Programación (mBlock):** `si (humedad < umbral) entonces encender LED, si no apagar`. Espacio para anotar el valor de umbral que probaste.
- **Diseño:** dibuja el **esquemático** (sub-hilo B) en el recuadro; marca dónde va cada cable.
- **Preguntas en el libro:** ¿Qué valor de humedad usaste como umbral y por qué? ¿Qué pasa si el umbral es muy alto? Si tuvieras dos plantas distintas, ¿usarías el mismo umbral?

- **Evaluación:** rúbrica de microproyecto (`11…§5.1`), /20, nota y firma.

Libro · Inventores · 7° — ODS 2

Título: *“El huerto que se cuida solo”*

ODS 2 · Hambre cero. Pregunta que detona: *¿Cómo producir más alimento usando menos recursos y menos esfuerzo?*

El reto (prototipo de solución): un **invernadero autónomo** que sensa su ambiente, decide y actúa, y **registra datos** para mejorar el cultivo. Papel → MDF.

Índice: 1) El hambre y el cultivo inteligente · 2) La pregunta · 3) Lo que vamos a construir · 4) Mapa del libro · 5) MP1: Leer el clima del huerto · 6) MP2: Decidir solo · 7) MP3: La bitácora de datos · 8) MP4: Modelar y armar · 9) Del papel al MDF · 10) Socialización · 11) Rúbrica final · 12) Anexo recortable · 13) Glosario.

Orden de ejes: Comprender → Diseñar (modelado) → Construir (sensores/actuadores) → Programar (funciones, MicroPython con BIPES de puente) → IA y datos (leer tendencias) → Integrar → MDF → Socializar.

Microproyectos: - **MP1 · Leer el clima del huerto** — sensores de temperatura/humedad en pantalla OLED. - **MP2 · Decidir solo** — lazo cerrado: si hace calor, ventila; si hay sequedad, riega. - **MP3 · La bitácora de datos** — registrar lecturas y leer una tendencia. *(desarrollado abajo)* - **MP4 · Modelar y armar** — modelar el invernadero (FreeCAD/papel) e integrar.

Proyecto final (MDF): invernadero en MDF con Raspberry Pi Pico, multisensor y actuadores.

▶ Microproyecto 3 desarrollado · “La bitácora de datos”

- **Objetivos:** (1) guardar lecturas en una **lista**; (2) calcular un promedio con una **función**; (3) leer una **tendencia** en los datos; (4) conversar sobre **calidad de los datos**.
- **Materiales:** *Del anexo:* plantilla **C-3** (panel para la OLED). *Componentes:* Pico, sensor DHT, pantalla OLED.
- **Construcción (papel):** recorta y arma **C-3** ✂┄; fija la OLED en la ventana marcada; conecta el sensor.
- **Programación (MicroPython):** función `promedio(lista)`; cada lectura se añade a la lista y se muestra el promedio en la OLED. Espacio para pegar/escribir 5 lecturas.
- **Diseño:** organiza la pantalla (qué se muestra y dónde) — sub-hilo C/experiencia.
- **Preguntas en el libro:** ¿Qué tendencia ves en tus datos? ¿A qué hora conviene regar según ellos? ¿Un solo dato basta para decidir, o necesitas varios? ¿Por qué?

- **Evaluación:** rúbrica de microproyecto (`11…§5.1`), /20, nota y firma.

Libro · Innovadores · 10° — ODS 11

Título: *“El aire que respira mi barrio”*

ODS 11 · Ciudades y comunidades sostenibles. Pregunta que detona: *¿Cómo puede una comunidad saber, en tiempo real, si el aire que respira es sano — y decidir con ese dato?*

El reto (prototipo de solución): una **estación IoT** que mide la calidad del aire, la envía por internet y la muestra en un **dashboard** que el barrio puede consultar. Papel → MDF.

Orden de ejes: Comprender → Diseñar (sistema + UI) → Construir (nodo ESP32) → Programar (WiFi/MQTT, SQL, web) → IA y datos (alertas/predicción opcional) → Integrar → MDF → Socializar con la comunidad.

Microproyectos: - **MP1 · El nodo que mide** — ESP32 + sensor de aire, lectura local. - **MP2 · Que hable por WiFi** — enviar la medición por la red (HTTP/MQTT). - **MP3 · Guardar y mostrar** — base de datos + dashboard web. *(desarrollado abajo)* - **MP4 · La experiencia para el barrio** — UI/UX del tablero e integración del sistema.

Proyecto final (MDF): estación de monitoreo de aire en MDF con ESP32 + dashboard web conectado.

▶ Microproyecto 3 desarrollado · “Guardar y mostrar”

- **Objetivos:** (1) enviar datos del nodo a una **base de datos**; (2) consultar con **SQL**; (3) mostrarlos en un **dashboard** web; (4) decidir qué visualización comunica mejor.
- **Materiales:** *Del anexo:* plantilla **D-3** (marco del tablero/maqueta de pared). *Componentes:* ESP32 con el nodo del MP1–2; computador con el dashboard.
- **Construcción (papel):** arma **D-3** ✂┄ como soporte de la pantalla donde se proyecta el dashboard del barrio.
- **Programación:** el nodo publica la medición → se guarda en una tabla SQL → el dashboard (HTML/CSS/JS) consulta y grafica. Espacio para anotar la consulta SQL usada.
- **Diseño:** boceta el dashboard (sub-hilo C/UI-UX): ¿qué ve primero un vecino? ¿colores de alerta?

- **Preguntas en el libro:** ¿Qué pregunta responde tu dashboard de un vistazo? ¿Cómo mostrarías una alerta sin asustar? Con una semana de datos, ¿qué decisión podría tomar tu barrio?
- **Evaluación:** rúbrica de microproyecto (`11…§5.1`) /20 + al cierre, **rúbrica final de convergencia** (`04`) sobre el prototipo en MDF.

Nota de réplica

Estos cuatro son **modelos**. Cada uno de los demás grados tendrá su propio libro con su ODS y su proyecto (del repositorio `07`), siguiendo exactamente esta estructura y la anatomía de `11` . Lo siguiente, cuando entremos a **kits**, es asociar a cada libro su lista de materiales (papel troquelado + componentes + piezas MDF) y su costo.

Fin del documento v0.1.

Línea Ares — Comodines: Cartillas y Recursos de Conocimiento

ROBOTSchool · Documento maestro (fuente única) Versión 0.2 — Junio 2026

> Los **comodines** son piezas de conocimiento **reutilizables**, **una por cada elemento**: una cartilla para cada **placa**, una para **cada sensor**, una para **cada actuador** y una para **cada módulo**. No viven dentro de los libros: cada libro **llama** los comodines de los elementos que usa. Cada comodín es una **cartilla** / **libro digital imprimible** que vive en la **plataforma** (Academy), acompañada de recursos digitales (video, H5P, mini-quiz); el estudiante o el colegio la **imprime solo si quiere**, cuando la necesita. Los **libros llevan los microproyectos**; los **comodines llevan el conocimiento**.

---

1. ¿Qué es un comodín? (uno por elemento)

Un comodín explica **un solo elemento** (una placa, un sensor, un actuador o un módulo) de forma completa y reutilizable. Ventajas de hacerlo por elemento:

- **Reutilización máxima:** la cartilla “Sensor de humedad” sirve en *todos* los grados, bandas y proyectos donde aparezca ese sensor — se escribe una vez.
- **El libro llama solo lo que usa:** un proyecto de riego llama Arduino + sensor de humedad + bomba + relé; no carga teoría que no necesita.
- **Mantenimiento barato:** si cambia un componente, se actualiza **una** cartilla, no doce libros.
- **Formato digital imprimible:** la cartilla vive en la plataforma como **PDF/libro digital** que se puede imprimir *just-in-time*; va acompañada de recurso digital (video, H5P, mini-quiz en Academy). No se imprime ni se empaca de fábrica.
- **Dos profundidades:** completo (primera vez) y repaso (currículo en espiral).

---

2. Catálogo de comodines

2.1 Placas

Código	Comodín
`COM-PL-MK`	Makey Makey
`COM-PL-ARD`	Arduino UNO
`COM-PL-MB`	micro:bit (banda alterna a Arduino)
`COM-PL-PIC`	Raspberry Pi Pico
`COM-PL-ESP`	ESP32

2.2 Base (transversales de concepto)

Código	Comodín
`COM-BS-CIR`	Circuitos y electricidad (pila, cable, polaridad)
`COM-BS-MEC`	Mecánica y mecanismos (palancas, engranajes, poleas)
`COM-BS-PROG`	Programación (bloques → texto)

2.3 Sensores (uno por cada uno)

Código	Comodín	Mide
`COM-SE-PUL`	Pulsador / botón	Si se presiona (digital)
`COM-SE-LDR`	Fotorresistencia (LDR)	Luz (analógico)
`COM-SE-POT`	Potenciómetro	Posición/giro (analógico)
`COM-SE-HUM`	Sensor de humedad de suelo	Agua en la tierra (analógico)
`COM-SE-ULT`	Ultrasónico (HC-SR04)	Distancia (pulsos)
`COM-SE-DHT`	DHT11 / DHT22	Temperatura y humedad del aire
`COM-SE-PIR`	PIR	Movimiento
`COM-SE-GAS`	Sensor de gas / calidad de aire (MQ)	Gases / calidad de aire

2.4 Actuadores (uno por cada uno)

Código	Comodín	Hace
`COM-AC-LED`	LED	Da luz
`COM-AC-RGB`	LED direccionable (NeoPixel)	Luz de colores programable
`COM-AC-BUZ`	Zumbador (buzzer)	Sonido
`COM-AC-MDC`	Motor DC	Giro continuo
`COM-AC-SRV`	Servomotor	Giro a un ángulo exacto
`COM-AC-BMB`	Mini bomba de agua	Mueve agua

2.5 Módulos (uno por cada uno)

Código	Comodín	Para qué
`COM-MO-REL`	Módulo relé	Encender aparatos de más corriente
`COM-MO-DRV`	Driver de motor (L298N)	Controlar motores DC
`COM-MO-OLED`	Pantalla OLED	Mostrar datos
`COM-MO-BT`	Módulo Bluetooth	Comunicación inalámbrica corta
`COM-MO-WIFI`	Módulo / WiFi (en ESP32)	Conexión a internet
`COM-MO-RTC`	Reloj de tiempo real (RTC)	Hora y fecha

> Catálogo inicial; crece a medida que se desarrollan proyectos. Cada fila = **una cartilla + un recurso digital**.

---

3. Plantilla de una cartilla de comodín (estructura fija)

Toda cartilla de elemento tiene **siempre** estas secciones (cabe en 1–2 páginas):

1. **Nombre, código e imagen** del elemento.
2. **¿Qué es?** — definición simple (1–2 frases).
3. **¿Qué hace** / **qué mide?** — su función.
4. **¿Cómo se conecta?** — tipo de señal y pines (digital / analógico / PWM / datos).
5. **Dato de programación** — cómo se lee o se controla (bloque o instrucción).
6. **Aparece en proyectos** — ejemplos del repositorio donde se usa.
7. **Compatibilidad** — en qué placas se usa (Arduino / Pico / ESP32).
8. **Mini-quiz** / **dato curioso** — para verificar y enganchar.

---

4. Comodines desarrollados (ejemplos)

`COM-PL-ARD` · Arduino UNO *(placa)*

- **¿Qué es?** Una placa programable: un “cerebro” que **recibe → decide → actúa**.
- **Partes:** pines digitales (sí/no), analógicos A0–A5 (0–1023), USB (programa + energía), pines 5V/GND.
- **Cómo se le ordena:** se sube un programa por USB (bloques/C++) y lo ejecuta en bucle.
- **Mini-quiz:** ¿qué hace un pin digital? ¿para qué los analógicos?

`COM-SE-HUM` · Sensor de humedad de suelo *(sensor)*

- **¿Qué es?** Un sensor que detecta cuánta agua hay en la tierra.
- **Qué mide:** humedad del suelo, como un número.
- **Cómo se conecta:** salida **analógica** → pin **A0**; alimentación 5V/GND.
- **Programación:** `leer analógico (A0)` → da un valor; tierra seca y húmeda dan números distintos.
- **Aparece en:** Riego automático (ODS 6), Invernadero autónomo (ODS 2).
- **Compatibilidad:** Arduino · Pico · ESP32.
- **Dato:** no “sabe” que la planta tiene sed; solo entrega un número que tú comparas con un **umbral**.

`COM-AC-SRV` · Servomotor *(actuador)*

- **¿Qué es?** Un motor que gira a un **ángulo exacto** (0°–180°), no en vueltas libres.
- **Qué hace:** mover una pieza a una posición precisa (un brazo, una compuerta).
- **Cómo se conecta:** señal **PWM** a un pin digital (~); alimentación 5V/GND.
- **Programación:** `mover servo (pin) al ángulo (90)`.
- **Aparece en:** Brazo robótico (ODS 9), Dispensador para mascota (ODS 15).
- **Compatibilidad:** Arduino · Pico · ESP32.

`COM-AC-LED` · LED *(actuador)*

- **¿Qué es?** Una pequeña luz que se enciende con poca corriente.
- **Cómo se conecta:** pin **digital** + **resistencia** (¡siempre!); respeta la polaridad (pata larga +).
- **Programación:** `encender/apagar LED en pin (13)`.
- **Aparece en:** Semáforo (ODS 11), y como señal en casi todos los proyectos.

`COM-MO-REL` · **Módulo relé *(módulo)***

- **¿Qué es?** Un interruptor controlado por el Arduino que permite encender aparatos de **más corriente** (una bomba, un foco) que la placa no podría mover sola.
- **Cómo se conecta:** señal a un pin **digital**; el aparato va al lado de potencia del relé.
- **Programación:** `encender/apagar relé en pin (9)`.
- **Aparece en:** Riego automático (controlar la bomba), Hogar inteligente.

> Los demás elementos del catálogo (§2) siguen exactamente esta misma plantilla.

---

5. Cómo los llaman los libros

Cada libro abre con **"Antes de empezar: tus comodines"**, listando los **elementos** que usa y su profundidad. Ejemplo:

Libro Constructores 4° "Ni una gota de más" (riego): > `COM-PL-ARD` (repaso) · `COM-SE-HUM` (completo, es nuevo) · `COM-AC-BMB` (completo) · `COM-MO-REL` (completo) · `COM-AC-LED` (repaso) · `COM-BS-PROG` (repaso).

Así el estudiante repasa lo conocido y aprende a fondo solo los **elementos nuevos** del proyecto.

5.1 Espiral por elemento

Un mismo elemento se llama **completo** la primera vez que aparece en la trayectoria del estudiante, y **repaso** después. Ej.: el LED se ve completo en 3° y solo se repasa en adelante; el sensor de humedad se ve completo cuando el primer proyecto lo usa.

---

6. El comodín es digital e imprimible (en la plataforma)

El comodín **no se imprime ni se empaca de fábrica**: vive en la plataforma y el colegio/estudiante lo imprime *just-in-time* si lo necesita.

Pieza	Qué es	Dónde
Cartilla digital imprimible	Libro/PDF de 1–2 páginas por elemento, listo para imprimir	Plataforma (Academy)
Recurso digital	Video corto + **H5P interactivo** + **mini-quiz** autocalificable	Plataforma (Academy)

> **Entrega:** los comodines llegan al estudiante por el **acceso anual a la plataforma** (junto con el repositorio). Lo físico que recibe es el **kit de hardware** (una vez por nivel) y el **libro con sus piezas MDF** (cada año).

---

7. Pendiente

1. Redactar **todas** las cartillas del catálogo (§2) con la plantilla de §3.
2. Producir cada una como **cartilla digital imprimible** + su recurso digital (video/H5P/quiz) en la plataforma.
3. Diagramas/fotos de cada elemento (conexión y pines) para diseño.
4. Tabla maestra “elemento → en qué proyectos del repositorio aparece” para priorizar cuáles producir primero.

Fin del documento v0.2. Complementa el modelo de libros (`11`) y la capacitación (`10`).

Línea Ares — Kits y Entrega de Material

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Cómo se arman los kits, qué incluye cada uno, y **cómo llega el material a cada estudiante** a lo largo de su nivel (3 grados). El kit de **hardware** se entrega **una sola vez por nivel** y es reusable; las **estructuras MDF** llegan con el **libro de cada año**; los **comodines y la plataforma** se acceden en digital con la renovación anual; y hay **venta de repuestos** para piezas dañadas o perdidas.

---

1. Cómo se construyen los kits

- **Hardware importado:** ROBOTSchool importa directamente las placas (Makey Makey, Arduino, Raspberry Pi Pico, ESP32, micro:bit), sensores, actuadores, módulos y cables.
- **Estructuras fabricadas en casa:** las carcasas y cuerpos de cada artefacto se **diseñan y se cortan** en **láser** (MDF) y se imprimen en **3D**. Esto permite innovar rápido y mantener costos bajo control.
- **Ventaja competitiva:** importar el hardware + fabricar las estructuras propias = márgenes mejores y capacidad de crear/ajustar proyectos sin depender de terceros.

---

2. Qué es el kit (y qué no)

- **Uno por estudiante y por nivel** (cubre los **3 grados** de la banda).
- **Se entrega una sola vez**, al inicio del nivel, con **hardware suficiente y durable** para todos los proyectos de los tres libros.
- **Los componentes se reúsan** entre proyectos y entre años: el estudiante **desarma** un proyecto para construir el siguiente. Lo que conserva armado es la **estructura MDF** de cada año.
- **No incluye** las estructuras MDF de los tres años de golpe: cada estructura llega con **su** libro (para que no se dañen ni se pierdan).

---

3. Lista de materiales (BOM) por nivel

> Hardware **durable** del kit (una vez por nivel) + **estructuras fabricadas** que llegan con cada libro. (Costos/precios: pendientes con el área de compras.)

Nivel 1 · Exploradores (Makey Makey)

Kit de hardware (durable)	Estructuras por año (con el libro)
Placa Makey Makey · set de cocodrilos y cables · lámina conductora · motor DC + portapilas + pilas · interruptor	Trans.: piano · 1°: tablero clasificador · 2°: molino con aspas

Consumibles del aula (los pone el colegio/familia): frutas, plastilina, cartulina.

Nivel 2 · Constructores (Arduino)

Kit de hardware (durable)	Estructuras por año
Arduino UNO + cable · protoboard · set de cables · LEDs + resistencias · pulsador · sensor de humedad de suelo · mini bomba + manguera · módulo relé · servomotor · fuente/ portapilas	3°: semáforo + intersección · 4°: estación de riego · 5°: brazo / mano

Nivel 3 · Inventores (Raspberry Pi Pico)

Kit de hardware (durable)	Estructuras por año
Raspberry Pi Pico (W) + cable · protoboard · cables · sensor DHT · pantalla OLED · sensor de humedad · mini bomba + relé · ventilador pequeño · sensor ultrasónico · servomotor	6°: estación meteorológica · 7°: invernadero · 8°: clasificador

Nivel 4 · Innovadores (ESP32)

Kit de hardware (durable)	Estructuras por año
ESP32 + cable · protoboard · cables · módulo relé · sensor de gas / calidad de aire (MQ) · sensor de humedad · LEDs	9°: casa inteligente · 10°: estación de aire · 11°: huerta conectada

*El proyecto integrador de 11° usa **varios nodos**: se puede trabajar en grupo o sumar ESP32 adicionales como repuesto/ampliación.*

---

4. Cómo se entrega al estudiante (año a año)

	Año 1 del nivel	Año 2	Año 3
Físico	**Kit de hardware** (una vez, reusable) **+** libro del grado 1 **+ piezas MDF** del año	Libro del grado 2 **+ MDF** del año	Libro del grado 3 **+ MDF** del año
Digital (plataforma)	Acceso: comodines imprimibles, repositorio, videos, H5P, quizzes	Renovación del acceso	Renovación del acceso

- El **hardware** se reúsa los 3 años (no se vuelve a entregar).
- Cada **libro** trae **sus** estructuras MDF.
- La **plataforma** (Academy) se renueva cada año con el nuevo libro.

---

5. Comodines y plataforma (digital imprimible)

Los **comodines** (una cartilla por elemento: placa, sensor, actuador, módulo) **no son físicos**: viven en la **plataforma** como **cartillas/libros digitales imprimibles** + sus recursos (video, H5P, quiz). El estudiante o el colegio **los imprime solo si los necesita**, *just-in-time*, cuando el libro los llama. (Detalle en `13`.)

---

6. Venta de repuestos

Como el kit debe durar **3 años**, se ofrece **venta de repuestos** para piezas que se dañen o se pierdan:

- **Componentes sueltos:** placa, sensores, actuadores, módulos, cables (los mismos que se importan).
- **Piezas MDF / 3D:** reposición de estructuras dañadas (fabricación propia).

- Es a la vez **servicio al colegio** y **ingreso adicional**.

> **Nota de mentor:** conviene definir un **paquete de repuestos recomendado por nivel** (lo que más se daña: cables, LEDs, sensores) que el colegio pueda comprar de entrada, para no frenar una clase por una pieza rota.

---

7. Modelo comercial (resumen)

Concepto	Cuándo se paga	Tipo de ingreso
Kit de hardware	Una vez, al iniciar el nivel	Único (con margen import + fabricación)
Libro + piezas MDF	Cada año (los 3 grados)	**Recurrente**
Acceso a la plataforma	Cada año	**Recurrente**
Repuestos	Según necesidad	Variable

> El modelo combina una **inversión inicial de hardware** con **ingresos recurrentes** (libro + plataforma) durante los 3 años — base sólida y predecible.

---

8. micro:bit — banda opcional (paralela a Constructores)

- **Opcional:** alternativa a Arduino en la banda Constructores, para colegios que la prefieran.
- **Ventaja:** el micro:bit v2 trae **muchos sensores integrados** (acelerómetro, brújula, temperatura, luz, micrófono, botones, matriz de LEDs, radio) → **kit con menos componentes externos** y **menos cableado**.
- **Costo:** la placa es **más cara**, pero se ahorra en sensores sueltos.
- **Estado:** la banda micro:bit **aún no está desarrollada**; queda como exploración. Al volver a Inventores, ambos caminos (Arduino y micro:bit) confluyen en MicroPython.

---

9. Pendiente

1. **Costos y precios:** costear cada kit por nivel (import + fabricación) y fijar precio de kit, libro, plataforma y repuestos.
2. **Empaque y troquelado:** definir empaque del kit y el troquel del anexo recortable de los libros.
3. **Paquete de repuestos** recomendado por nivel.
4. **Garantía/reposición:** política de qué cubre y qué se cobra.
5. Decidir si se desarrolla la **banda micro:bit**.

Fin del documento v0.1. Complementa los libros (`11`, `12`, `14`) y los comodines (`13`).

Anexo · Libros y kits

Los 15 libros (HTML + PDF), uno por grado más la banda micro:bit, son archivos independientes:

Exploradores - Transición · El mundo que responde (ODS 4) - 1° · Mi planeta sin basura (ODS 12) - 2° · La fuerza del viento (ODS 7)

Constructores - 3° · Ciudad que ordena (ODS 11) - 4° · Ni una gota de más (ODS 6) - 5° · Manos que ayudan (ODS 9)

micro:bit (alterna) - 3° · Cruce seguro (ODS 11) - 4° · El termómetro del salón (ODS 13) - 5° · Muévete sano (ODS 3)

Inventores - 6° · Leer el cielo (ODS 13) - 7° · El huerto que se cuida solo (ODS 2) - 8° · Datos que clasifican (ODS 12)

Innovadores - 9° · Casa que ahorra (ODS 7) - 10° · El aire de mi barrio (ODS 11) - 11° · Comunidad conectada (ODS 2/11)

La producción, BOM y modelo comercial de los kits están en el documento de la Fase 7 (Kits y entrega).