Table des matières

1. Accessibilité — µcBlockly

µcBlockly est un **outil auteur** Blockly pour Arduino. L'accessibilité repose sur trois couches : la **coque HTML** (barre d'outils, panneaux), les **champs personnalisés** (couleur, variables typées), et le **moteur Blockly 13** (navigation clavier, `FocusManager`, annonces lecteur d'écran).

Niveau visé : alignement progressif sur WCAG 2.2 niveau AA et les bonnes pratiques Blockly, en complément des Accessibility Conformance Reports (ACR) de Google Blockly.

Contact : Sébastien Canet (LinkedIn) — dépôt A-S-T-U-C-E/ucBlockly.

—

1.1. État des priorités

Lot	Contenu	État
P0 — Champs custom	`getAriaValue()` sur sélecteurs couleur, éditeurs iro accessibles	Fait
P0 — Blockly v13	Navigation clavier cœur, `FocusManager`, hints, mode lecteur d'écran	Fait (Blockly **13.2.x**)
P1 — Coque UI	Skip link, landmarks, boutons sémantiques, menu zones Ctrl+B, raccourcis fichier	Fait
P1 — Doc	Cette page, raccourcis, conformité	Fait
P1 — Intégration coque ↔ Blockly	Menu zones et skip link via `FocusManager` (raccourcis **W** / **T**)	Fait
P1 — Traductions Blockly v13	Clés `SHORTCUTS_*`, `KEYBOARD_NAV_*`, `SCREENREADER_*` en fr/es/ar	Fait
P2 — Thèmes	Sync coque + Monaco, contraste, persistance thème	Fait
P2 — Aide raccourcis	Modale coque + Blockly (F1, ?, bouton barre d'outils)	Fait
P2 — Navigation avancée	`registerNavigationShortcuts()` (Home, Page Up/Down, etc.)	Fait
P2 — Mode lecteur d'écran	Persistance `localStorage` (`ucbScreenreaderMode`)	Fait
P2 — Tutoriel clavier	Modale pas-à-pas au premier chargement (bouton barre d'outils)	Fait

—

1.2. Coque HTML

1.2.1. Lien d'évitement

Au premier **Tab** après chargement, le lien **« Aller au workspace Blockly »** (`A11Y_SKIP_TO_WORKSPACE`) délègue au `FocusManager` Blockly (équivalent **W**). Si le workspace n'est pas encore injecté, repli sur `<main id="blockly-workspace-main">`.

1.2.2. Repères (landmarks)

Zone	Repère
Barre supérieure	`role="banner"` (`#app_topbar`)
Workspace	`<main id="blockly-workspace-main">`
Code généré	`role="region"` sur `#div_content_code`
Console	`role="log"` + `aria-live="polite"` sur `#content_console`
Séparateurs flex	`role="separator"` avec `aria-orientation`

Implémentation : `src/shell_a11y.ts`, `src/blockly_shell_focus.ts`, `src/a11y_labels.ts`, `src/index.ts` (`applyA11yAttributes`, `registerShellBlocklyWorkspace`).

1.2.3. Menu de zones (Ctrl+B / Cmd+B)

1. **Ctrl+B** (Windows/Linux) ou **Cmd+B** (macOS) — ouvre le menu.
2. Message annoncé via `#shortcuts` (`aria-live="polite"`).
3. Touches **1** à **5** :

- **1** — barre d'outils
- **2** — boîte à outils Blockly (`FocusManager`, équivalent **T**)
- **3** — workspace (`FocusManager`, équivalent **W**)
- **4** — panneau code
- **5** — console

1. **Échap** — annule.

Les intégrateurs embarqués doivent appeler `registerShellBlocklyWorkspace(() ⇒ workspace)` après l'inject Blockly pour activer les zones **2** et **3** et le skip link.

1.2.4. Barre d'outils

Enregistrer, Charger, Annuler et Rétablir sont des `<button type="button">` focusables, avec `aria-label` et infobulles i18n incluant les raccourcis fichier le cas échéant.

Les listes déroulantes (langue, thème, rendu, carte) portent un `aria-label` localisé. Le groupe d'options plugins est un `role="group"`. Les liens externes annoncent l'ouverture dans un nouvel onglet.

`lang`, `xml:lang` et `dir` sur `<html>` sont synchronisés à chaque changement de langue (RTL pour l'arabe).

—

1.3. Raccourcis clavier

1.3.1. Coque µcBlockly

Raccourci	Action
Tab (premier focus)	Lien d'évitement vers le workspace
Ctrl+B / **Cmd+B**	Menu de navigation par zones
Ctrl+S / **Cmd+S**	Enregistrer le workspace (hors champs de saisie texte)
Ctrl+O / **Cmd+O**	Ouvrir un fichier (hors champs de saisie texte)
1 – **5** (après Ctrl+B)	Focus zone 1 à 5
Échap (menu zones ouvert)	Fermer le menu zones
F1 ou **?**	Ouvrir l'aide des raccourcis (hors champs texte)

Les raccourcis fichier sont gérés par `src/shell_shortcuts.ts` et **ne s'appliquent pas** dans Monaco ni dans les champs texte (`isTextEntryTarget`).

1.3.2. Blockly 13 (cœur)

À l'import de `blockly`, trois jeux de raccourcis sont enregistrés automatiquement :

Jeu	Rôle
`registerDefaultShortcuts`	Édition de base (copier, coller, annuler, supprimer…)
`registerKeyboardNavigationShortcuts`	Navigation flèches, focus workspace/toolbox, déplacement
`registerScreenReaderShortcuts`	Annonces et mode lecteur d'écran

Référence officielle : Blockly keyboard shortcuts.

1.3.3. Navigation et focus

Raccourci	Action
Flèches	Naviguer entre blocs, champs et connexions
W	Focus workspace
T	Focus boîte à outils (toolbox) ou flyout
Entrée / **Espace**	Éditer ou confirmer l'élément focusé
N / **B**	Stack suivante / précédente
M	Démarrer un déplacement de bloc
H / **Shift+H**	Titre (heading) suivant / précédent dans le flyout

1.3.4. Édition

Raccourci	Action
Ctrl+C / **Cmd+C**	Copier le bloc focusé
Ctrl+X / **Cmd+X**	Couper le bloc focusé
Ctrl+V / **Cmd+V**	Coller
Ctrl+Z / **Cmd+Z**	Annuler
Ctrl+Shift+Z / **Cmd+Shift+Z** ou **Ctrl+Y**	Rétablir
Suppr / **Retour arrière**	Supprimer le bloc focusé
Shift+X	Déconnecter le bloc
D	Dupliquer
C	Nettoyer le workspace (réorganiser les blocs)

Raccourci	Action
Échap	Fermer menus et popovers

1.3.5. Lecteur d’écran

Raccourci	Action
I	Annoncer une description de l’élément focusé
Shift+I	Annoncer une description détaillée
Alt+Shift+A	Activer / désactiver le mode lecteur d’écran

Le mode lecteur d’écran est **mémorisé** dans `localStorage` (`ucbScreenreaderMode`) et réappliqué après recréation du workspace. Au **premier focus clavier**, Blockly peut annoncer `SCREENREADER_HINT`.

1.3.6. Navigation avancée (sauts)

Activée explicitement via `Blockly.ShortcutItems.registerNavigationShortcuts()` (`src/blockly_a11y_extensions.ts`).

Raccourci	Action
Home	Début du bloc courant
End	Fin du bloc courant
Page Up	Haut de la pile
Page Down	Bas de la pile
Ctrl+Home / **Cmd+Home**	Premier bloc du workspace
Ctrl+End / **Cmd+End**	Dernier bloc du workspace

> **Note :** ces raccourcis ne font pas partie du jeu par défaut de Blockly ; µcBlockly les active au démarrage.

1.3.7. Coexistence coque / Blockly

Raccourci	Comportement
Ctrl+S / **Ctrl+O**	Coque uniquement (hors champs texte)
Ctrl+B	Menu de zones (coque)
Ctrl+C / **Ctrl+V**	Blockly dans le workspace ; Monaco dans l’éditeur de code
W / **T**	Blockly uniquement

L’ancien plugin `@blockly/keyboard-navigation` n’est **plus utilisé** : sa navigation est intégrée au cœur de Blockly 13.

1.3.7.1. Plugins de recherche

- **Recherche workspace** — `@blockly/plugin-workspace-search`
- **Recherche toolbox** — `@blockly/toolbox-search`

1.3.7.2. Zoom

Le zoom Blockly est activé jusqu'à **600 %** (`maxScale: 6`), au-delà du seuil 200 % recommandé pour l'accessibilité.

—

1.4. Thèmes et contraste (P2)

1.4.1. Thèmes recommandés pour l'accessibilité visuelle

Thème (menu)	Usage
deuteranopia	Daltonisme rouge–vert (deutéranopie)
tritanopia	Daltonisme bleu–jaune (tritanopie)
high_contrast	Contraste élevé Blockly + coque
dark	Mode sombre
blackWhite	Monochrome (contraste maximal des blocs)

Les thèmes **seshat**, **classic**, **modern**, **zelos** suivent aussi la synchronisation coque.

1.4.2. Synchronisation coque ↔ Blockly ↔ Monaco

`syncUiPaletteTheme()` (`src/ui_palette_theme.ts`) :

- dérive les couleurs des panneaux, barre d'outils et séparateurs depuis `componentStyles` du thème Blockly actif ;
- impose un **contraste minimal** (texte barre d'outils ≥ 4,5:1) ;
- pose `data-ui-theme` sur `<html>` (`light` | `dark` | `high-contrast`) ;
- aligne l'éditeur Monaco (`vs`, `vs-dark`, `hc-black`) via `µcB_codeEditor.applyTheme()`.

Le thème **noir et blanc** définit des `componentStyles` explicites pour une coque lisible.

1.4.3. Persistance

Le thème choisi est mémorisé dans :

- l'URL (`?theme=…`) ;
- `sessionStorage` (`paramTheme`).

Le changement de langue ou de carte ne réinitialise pas le thème.

—

1.5. Champs personnalisés (P0)

Champ	Fichier	ARIA
Sélecteur couleur (picker)	`field_colour_picker.ts`	`getAriaTypeName`, `getAriaValue`, éditeur iro
Sélecteur couleur HSV	`field_colour_hsv_sliders.ts`	idem + curseurs étiquetés
Helpers	`field_colour_a11y.ts`	format hex + RGB pour lecteurs d'écran

Tests : `tests/field-colour-a11y.test.ts`.

Les **blocs Arduino custom** ajoutés par des tiers doivent implémenter les mêmes API pour les champs qu'ils créent (doc Blockly — ARIA value).

—

1.6. Conformité Blockly (ACR)

µcBlockly s'appuie sur Blockly **13.1.x** (`blockly ^13.1.1`). Les rapports de conformité Google couvrent le **cœur Blockly** ; les extensions µcBlockly (coque, blocs Arduino, champs couleur) relèvent de la responsabilité de ce projet.

Ressource	Lien
Principes POUR	Blockly accessibility principles
Bonnes pratiques intégration	App integration — accessibility
Conformité & ACR	Compliance
Inspiration coque	MakeCode Accessibility

—

1.7. Limites connues

- **Menu zones / skip link** : repli DOM si le workspace Blockly n'est pas encore disponible.
- **Tutoriel clavier** : modale pas-à-pas (`src/shell_keyboard_tutorial.ts`), affichée au premier chargement sauf si « Ne plus afficher ».
- **Aide raccourcis in-app** : modale native (<dialog>) via le bouton **raccourcis**, **F1** ou **?** — voir `src/shell_shortcuts_help.ts`.
- **Annonces** du code généré à chaque modification : non activées (risque de spam lecteur d'écran). Une annonce brève suit l'enregistrement fichier (Ctrl+S).
- Éléments layout **<flex>** custom : peu sémantiques ; les repères ARIA compensent partiellement.
- **RTL** : arabe supporté pour l'UI ; vérifier manuellement la cohérence toolbox + coque.

Les **hints** et libellés de raccourcis Blockly 13 sont surchargés via `src/languages/blockly_a11y_messages.ts` (fr, en, es, ar), fusionnés dans `applyUcBlocklyLocale()`.

—

1.8. Tests manuels suggérés

1. Clavier seul : skip link → **W** (focus workspace) → flèches → créer un bloc → **Ctrl+S** → sélecteur couleur → **T** (toolbox).
2. Lecteur d'écran (NVDA, Narrator, VoiceOver) : menu zones Ctrl+B, hints Blockly, champs couleur, **Alt+Shift+A** (mode lecteur d'écran).
3. Chaque thème d'accessibilité : lisibilité barre d'outils, panneau code (pré + Monaco), workspace avec anneaux de focus clavier (`blocklyKeyboardNavigation`).
4. Zoom navigateur 200 % : pas de chevauchement bloquant des panneaux.

Commandes projet : `npm test`, `npm run check-translations`, `npm run lint`.

—

Voir aussi : Architecture · README