Table des matières

1. Répartition des fonctions entre fichiers – Analyse et améliorations 3
1.1. 1. Duplications identifiées (historique — corrigées depuis 2.1.5 / v3) 3
1.2. 2. Répartition actuelle – points corrects 4
1.3. 3. Améliorations structurelles optionnelles 4
1.4. 4. Plan d'action proposé 5
1.5. 5. Dépendances actuelles (grosses lignes) 5
1.6. Voir aussi 6

1. Répartition des fonctions entre fichiers – Analyse et améliorations

1.1. 1. Duplications identifiées (historique — corrigées depuis 2.1.5 / v3)

Les sections ci-dessous décrivent d'anciennes duplications **déjà résolues**. Voir §4 « Plan d'action ». Pour les pistes encore ouvertes en v3, voir CODE_IMPROVEMENTS.md.

1.1.1. 1.1 ''getBlocklyMsg'' / ''getBlocklyMsgOr''

Fichier	Rôle
categories/blockly/shared.ts	Source canonique (exportée).
generators/arduino/configure_standard_blocks.ts	Importe depuis shared.
generators/arduino/typed_variables_flyout.ts	Importe depuis shared.

—

1.1.2. 1.2 Labels / options de dropdown type variable

Fichier	Rôle
categories/blockly/variables.ts	getArduinoTypeLabel, getArduinoTypeDropdownOptions (exportés).
generators/arduino/configure_standard_blocks.ts	Importe getArduinoTypeDropdownOptions depuis variables.ts.

—

1.1.3. 1.3 Type ''VariableWithMutableType''

Fichier	Rôle
categories/blockly/shared.ts	Type exporté (avec __ucbManualType, etc.).
generators/arduino/configure_standard_blocks.ts	Import depuis shared.

—

1.1.4. 1.4 Wrapper « set type + manuel »

Helper **setVariableTypeAndMarkManual** dans shared, utilisé dans configure_standard_blocks et variables_set_type.

—

1.2. 2. Répartition actuelle – points corrects

- **`block_types.ts`** : règles de types pures (`TYPE_COMPATIBILITY`, `ARDUINO_TYPE_TO_INTERNAL`, `getInternalTypeForArduinoType`, `getCompatibleTypes`, `isTypeCompatible`, `getVariableTypeOptions`). Pas de Blockly. Documenté dans TYPES_ET_COMPATIBILITES.md.
- **`block_types_registry.ts`** : registre, `checkTypeCompatibility` (côté Blockly), `getCompatibleTypesForCheck`, application aux blocs. Utilise `block_types`. Bonne séparation.
- **`variable_utils.ts`** : `setVariableType`, `clearVariableType`, `normalizeVariableType`. Utilisé par shared, variables, logic, configure_standard_blocks, workspace manager. Rôle clair.
- **`shared.ts`** : i18n, inférence de type, helpers variables/blocs, avertissements, validation. Centralise le réutilisable entre categories/blockly. Cohérent.

—

1.3. 3. Améliorations structurelles optionnelles

1.3.1. 3.1 Découpage de ''shared.ts''

`shared` mêle i18n, inférence de types, handlers de variables, messages d'avertissement, validation. On pourrait extraire :

- **`blockly_i18n.ts`** (ou `i18n.ts` dans blockly) : `getBlocklyMsg`, `getBlocklyMsgOr` → utilisé partout.
- **`type_inference.ts`** : `inferConnectedValueType`, `normalizeBlocklyTypeToVariableType`, `getFirstCheckType`, `inferProcedureCallReturnType`.

Intérêt : responsabilités plus nettes. Coût : plus de fichiers et de refactoring. À faire seulement si on prévoit d'enrichir fortement i18n ou inférence.

1.3.2. 3.2 ''logic.ts'' (≈ 456 lignes)

Contient switch/case, mutator, `applySwitchCaseTyping`, tooltips, etc. On pourrait isoler la partie « switch » dans un `logic_switch.ts` et garder `logic.ts` pour le reste + orchestration. Optionnel, surtout si on touche souvent au switch.

1.3.3. 3.3 ''variables.ts'' (≈ 627 lignes)

Beaucoup de définitions de blocs (get, set, local, const, init, define, set_type). Un découpage en

`variables_get.ts`, `variables_set.ts`, `variables_const.ts`, etc. est possible, mais éparpille une logique déjà liée. À considérer seulement si le fichier devient difficile à maintenir.

—

1.4. 4. Plan d’action proposé

1.4.1. Priorité 1 – Dedup immédiate (impact rapide, peu de risque) FAIT

1. **`getBlocklyMsg` / `getBlocklyMsgOr`**

- `configure_standard_blocks` et `typed_variables_flyout` importent depuis `shared`, suppression des copies locales.

1. **`getArduinoTypeLabel` / `getArduinoTypeDropdownOptions`**

- Export depuis `variables.ts`, réutilisation dans `configure_standard_blocks`, suppression des doublons.

1. **`VariableWithMutableType`**

- Import depuis `shared` dans `configure_standard_blocks`, suppression de la définition locale.

1.4.2. Priorité 2 – Centraliser la logique « set type + manuel » FAIT

1. **Helper `setVariableTypeAndMarkManual`**

- Ajout dans `shared` (ou `variable_utils` selon choix d’architecture), utilisation dans `configure_standard_blocks` et `variables_set_type`. `handleVariablesSetChange` continue d’utiliser `setVariableType` + `__ucbManualType = false` (inférence, pas manuel).

1.4.3. Priorité 3 – Optionnel

1. Extraction **`blockly_i18n`** / **`type_inference`** si croissance de ces parties.
2. Découpage **`logic_switch`** / **`variables_*`** si besoin de clarté ou de collaboration sur des zones très ciblées.

—

1.5. 5. Dépendances actuelles (grosses lignes)

- **`categories/blockly`** → `generators/arduino` (shared utilise block_types_registry, variable_utils, debug).
- **`generators/arduino`** → **`categories/blockly`** : imports limités et explicites (`shared`, `variables`) depuis `configure_standard_blocks` et `typed_variables_flyout`. Documenté dans ARCHITECTURE.md et README.md.

—

1.6. Voir aussi

- ARCHITECTURE.md — organisation du dépôt et scripts npm ;
- TYPES_ET_COMPATIBILITES.md — typage des blocs ;
- CONTRIBUTING.md — guide contributeur.

From:
https://wiki.libreduc.cc/ - **LibrEduc**

Permanent link:
https://wiki.libreduc.cc/fr:arduino:ucblockly:repartition

Last update: **2026/08/11 00:18**