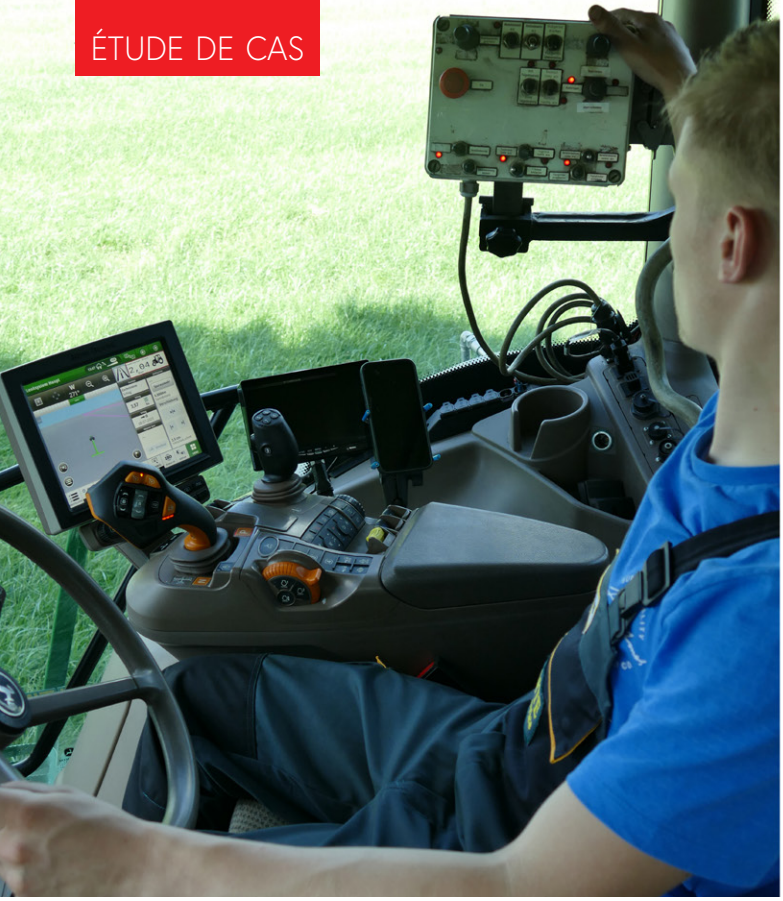

une API avec les solutions d'Espressif

avec les capacités et les fonctionnalités du protocole ISOBUS

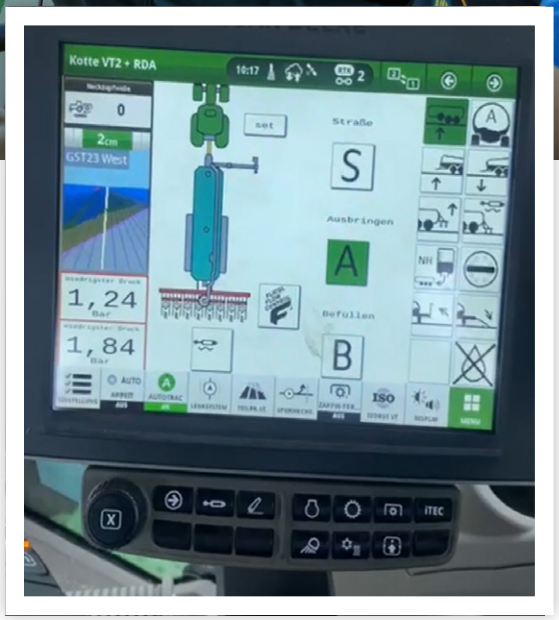

Figure 1. Interface utilisateur ISOBUS (ISO 11783) sur un moniteur John Deere. Il contrôle un slurry tanker.

Franz Höpfinger, HR Agrartechnik

HR Agrartechnik GmbH avait besoin d'une API peu coûteuse mais puissante, dotée de capacités ISOBUS (IOS 11783). La combinaison d'un ESP32 d'Espressif avec l'ESP-IDF, ainsi que le cadre Eclipse 4diac™, a donné naissance au projet logiBUS®.

Le défi

Les machines agricoles modernes nécessitent des systèmes de commande flexibles. Les nombres sont généralement faibles et la durée de vie des machines est longue, c'est pourquoi l'adaptation des systèmes de commande aux machines existantes sur le terrain est assez courante. Dans la **figure 1**, vous pouvez voir l'interface utilisateur du moniteur de contrôle du système après la mise à niveau.

L'apprentissage du langage C ou C++ et la gestion des bibliothèques et des systèmes sont complexes. Le débogage avec JTAG n'est pas flexible et nécessite des outils supplémentaires et un accès au microcontrôleur. En revanche, de nombreux programmeurs sont disponibles pour la programmation d'automates, et la programmation graphique est plus facile à apprendre.

ISOBUS, également connu sous le nom d'ISO 11783, est un protocole de communication développé pour les machines agricoles afin de faciliter l'échange de données entre différents types de machines et d'outils agricoles. ISOBUS se compose d'ISO (International Standardisation Organisation) et de BUS, qui est un système de communication entre plusieurs appareils.

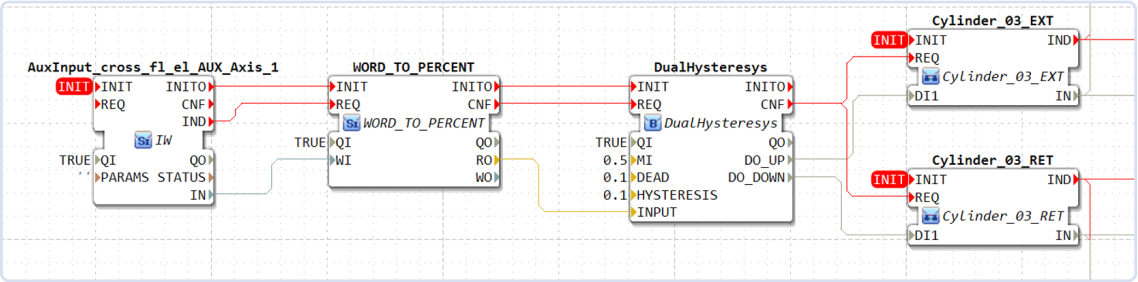

Figure 2. Ce listage du code source montre comment fonctionne la programmation dans la CEI 61499/Eclipse 4diac™. Vous n'écrivez pas de lignes de code, mais vous utilisez des blocs fonctionnels à la place.

Dans ce contexte, nous voulions concevoir un système API composé d'un système d'exécution sur le contrôleur lui-même et d'un EDI sur le PC qui fournirait un système d'automatisation flexible, convivial et rapide avec une programmation graphique à code bas basé sur des modèles et des bibliothèques puissants. Une séparation stricte entre un environnement d'exécution réutilisable et flexible et l'application figurait en tête de notre liste de souhaits. Le deuxième point était la possibilité d'observer réellement les variables et les blocs fonctionnels en ligne. D'autres points importants étaient la qualité du support des fournisseurs pour la pile logicielle et l'utilisation des ressources. Nous voulions éviter un système LINUX coûteux et entièrement intégré, principalement en raison du temps de démarrage. Les équipements agricoles sont susceptibles d'être allumés et éteints des dizaines de fois par jour.

Nous avons analysé plusieurs solutions, à la fois à code fermé et à code ouvert, et avons finalement choisi Eclipse 4diac™ [1]. « Pourquoi pas le projet OpenPLC ? », se demanderont peut-être certains lecteurs d'Elektor. La réponse est évidente : les deux principales exigences ne sont pas remplies : OpenPLC ne présente pas de séparation stricte entre l'application et le temps d'exécution, ni de fonctionnalité de veille en ligne.

La solution

Pour la solution logiBUS® [2], nous avons combiné l'aspect logiciel (cité ci-dessous) avec notre application de base :

- Cadre de développement ESP-IDF (actuellement Version 5.1.1)
- Runtime IEC 61499 open-source: Eclipse 4diac™ FORTE PLC
- Pilote CCI ISOBUS, une pile de protocoles ISO 11783

Il en résulte un système d'exécution d'automate indépendant de l'application souhaitée. La programmation se fait avec l'EDI 4diac™ d'Eclipse. Le flashage de l'application peut se faire via une connexion TCP/IP, donc via le Wifi ou l'Ethernet. Les solutions apportent de véritables fonctions d'automate telles que la surveillance et la modification en ligne.

Pour le matériel, un ESP32 avec PSRAM est un bon choix parce que le modèle utilise beaucoup de RAM, donc pour les modèles plus grands, un manque de PSRAM est un problème. Pour la connexion ISOBUS, nous utilisons le périphérique TWAI de l'ESP32 comme contrôleur de bus CAN, ainsi qu'un émetteur-récepteur CAN TLE9251VSJ externe d'Infineon. Certaines versions de matériel, en particulier ceux destinés à l'enseignement, sont libres [6].

Le projet logiBUS® ne cessant de se développer et de s'étendre, nous avons déjà réalisé une douzaine de projets clients réels ; la **figure 2** présente un exemple de listage du code source pour l'un d'entre eux.

Le système semble également adapté à d'autres secteurs, tels que l'automatisation de bâtiment [5]. Nous avons donc créé une version open-source de logiBUS® sans la pile ISOBUS, mais avec les mêmes fonctions et la même puissance.

Nous avons également mis en libre accès quelques applications de démonstration, telles que le Bale Counter [4] et le Slurry Tanker [5].

Nous espérons créer une communauté autour du thème des API open source [6].

230610-04

Questions ou commentaires ?
Contactez Elektor (redaction@elektor.fr).

LIENS

[1] Eclipse 4diac™ : https://eclipse.dev/4diac
[2] Page d'accueil de logiBUS® : https://www.logibus.tech
[3] Dérivés open-source de logiBUS® pour, par exemple, l'automatisation de bâtiment (pas ISOBUS) : https://gitlab.com/meisterschulen-am-ostbahnhof-munchen
[4] Bale Counter : https://github.com/Meisterschulen-am-Ostbahnhof-Munchen/4diac_EasyExampleCounter
[5] Slurry Tanker : https://github.com/Meisterschulen-am-Ostbahnhof-Munchen/4diac-SlurryTanker-sample
[6] Ressources et Wiki : https://github.com/Meisterschulen-am-Ostbahnhof-Munchen