Design of an IoT-PLC: A containerized programmable logical controller for the industry 4.0

IoT-PLC 的设计：工业 4.0 的容器化可编程逻辑控制器

**主要工作：**

**设计了一个为工业 4.0 革命量身定制的 PLC—— IoT-PLC 。**

1. IoT-PLC 作为一个容器化的设备运行，每个功能都封装在一个单独的容器中。
2. 具有监管控制功能、过滤和现场数据存储等雾计算功能。
3. 独立管理的多个无线接口。
4. 使用虚拟设备模型作为抽象方法来表示真实实体，以便 IoT-PLC 应用程序可以透明地交互并与上层云层直接兼容。
5. 实时迁移功能通过允许循环重新配置而无需重新启动控制器来增加灵活性。

**结论**

基于容器的模块化结构允许 **IoT-PLC**安全地隔离在其中运行的不同进程。此外，容器化促进了从传统的基于硬件的 PLC 转变为具有数据控制、处理和可视化的虚拟功能的 PLC，它可以充当工业自动化应用和 IT 技术之间的桥梁。

**背景：**

从由经典 ICS（industrial control systems） 过渡到工业 4.0 ，调整 ICS 中设备适应 ICPS（Industrial Cyber–Physical Systems） 范式，需要进行改造的设备包括可编程逻辑控制器 (PLC)。

PLC 一直是 ICS 的关键组成部分，其作用主要是指挥低级时间敏感的监管反馈控制回路，为流程提供第一个控制层。

工业 4.0 强加了一些在生产过程自动化或数字化中必须考虑的设计原则 [6]，即：

1. 整合，特别是生产流程与价值链的整合，使智能工厂能够对从供应链到服务和生命周期管理的整个制造流程进行端到端管理；
2. 互操作性，指所有系统组件使用标准化通信和数据透明地连接、通信和交互的能力
3. 权力下放，使不同的子系统能够做出旨在实现最终组织目标的自主本地决策；
4. 实时性能，这要求生产过程以适当的速度收集、交流和整合数据，以便及时做出战略决策；
5. 模块化，确保可以更换、扩展或改进单个组件，同时最大限度地减少对生产过程的干扰，从而使智能工厂能够轻松适应不断变化的物理环境和生产要求。

满足操作技术 (OT) 领域中典型的严格及时性约束是 PLC 的关键功能，在任何新开发中都应确保这一点。及时性限制因应用领域而异，典型的周期时间从微秒到几秒不等，如表 1 所示。此外，工业 4.0 PLC 还必须具有互操作性，并准备好与信息技术 (IT) 领域的平台集成。

**现代 PLC 是促进 OT-IT 集成的元素**

为了解决 IT 和 OT 领域之间的集成问题，同时满足他们的要求，雾计算范式已成为一种有吸引力的选择，可以将控制系统、存储和通信部署在更靠近现场设备的位置。可以减少 IT 层的响应时间和非确定性行为，因为在 IT 领域执行的部分任务被移到更靠近传感器和执行器的位置 [11]。在这种情况下，**将现代 PLC 设想为雾节点**似乎很有希望。

**IoT-PLC 将每个现场设备描述为边缘和云应用程序的资源（虚拟设备）**

**设计：**

包括各种**无线接口**，用于与传感器和执行器进行现场级连接。此外，它在内部包含一个用于数据流预处理的**过滤模块**，以及一个用于短期趋势分析和通过**人机界面** (HMI) 显示的轻量级本地**数据库**。**管理器**进程负责使用现场传感器和执行器生成控制回路，并以容器的形式实例化控制器以进行低级监管控制。对于高水平连接，它包含两个接口，即一个**用于云连接**OPC-UA的专用接口，能够提供用于与各种控制系统集成的标准化信息模型，以及一个用于与现场部署的**其他 IoT-PLC 水平连接**CoAP的轻量级接口。

IoT-PLC 旨在通过不同协议与设备交互，提取相关信息以进一步整合云，并提供标准数据格式以透明地集成到云中。

IoT-PLC 被设想为 ICPS 雾层中的一个节点，通过这种方式，它充当 (i) **现场设备的网关**，这些设备使用各种传感器和面向执行器的协议进行通信，特别是无线; (ii) **第一级控制器**，根据从传感器获取的信息和从云端接收的命令操纵执行器。

IoT-PLC 的内部结构如图 1 所示，其中每个块代表一个在容器内运行的功能脚本或进程，该容器独立工作并根据发布者/订阅者通信方案与其他块交换信息。

通过在容器中运行每个块，可以直接从云中管理它们。这使 IoT-PLC 可以灵活地动态更改控制器的配置，而无需停止控制过程，只需重新加载控制器块。

**1、OPC-UA 连接**

使用 OPC-UA 协议可以连接到云。

OPC-UA 是一个独立于平台的标准，不同的系统和设备通过它进行通信。 OPC-UA 可用于所有工业级别，例如现场（传感器、执行器）、控制系统、制造执行系统 ​​(MES) 和企业资源规划 (ERP)。

通过 OPC-UA，收集的数据被发送到云中的服务器。此外，设备、控制器和回路的描述可以从云端发送，以便在控制过程中添加或修改。

使用 OPC-UA 块，还可以将必要的命令从云端发送到 IoT-PLC，以继续将容器迁移到另一个设备。

**2、与现场设备的接口**

IoT-PLC 将其与现场设备的连接建立在双向接口模块上，该模块将数据从高级 IoT-PLC 模块转换到现场设备，反之亦然。每个接口块只能使用一种协议进行通信，例如 Zigbee、蓝牙、无线 Hart 或 Wi-Fi。接口块通过发布/订阅事件消息与上层块管理器和过滤器块进行通信。通过这样做，上层模块可以在需要时获取数据，而无需直接依赖于现场设备或 IoT-PLC 的其他构建模块的传输速率。

**3、CoAP接口**

CoAP 接口块旨在提供与其他 IoT-PLC 或雾设备的连接。使用此接口，可以交换设备和控制器的数据，并向其他 IoT-PLC 发出控制命令，例如控制回路终止信号。这是通过使用 CoAP 多播服务器发现功能实现的，从而在网络中的设备之间实现分散的发现过程。

CoAP 协议是一种成熟的面向物联网的解决方案，因其在网关设备中经过验证的特性而被选中，例如低开销、减少延迟和分散的数据交换 [22]。这些功能确保与其他 IoT-PLC 的通信不会降低本地控制过程的性能。

**4、数据过滤**

数据过滤器块作为接口块和上层元素之间的抽象层运行。数据过滤器块从接口收集信息并对其进行预处理以供上层块稍后使用。来自接口块的传入数据流经过预处理以**减少噪声、修复丢失的数据并使用过滤器屏蔽异常值**。此块中应用的过滤器是根据从云端发送的配置实例化的，因此可以根据传感器和流程的需要应用不同的过滤器。然后将经过验证的数据发布到需要数据的上层块。

**5、数据库**

数据库块充当整合单元。当信息被发送到云端或被 HMI 获取时，信息会被转换以使其更有意义。此块中整合的数据对应于来自传感器的过滤信息、IoT-PLC 内部控制算法生成的数据以及从 HMI 块发送的数据。

数据库块负责将数据发送到 OPC-UA 块以传输到云端。它还从 OPC-UA 块接收 IoT-PLC 将连接到的设备的描述和配置，并保存它们。数据库块提供了一个适配层，使数据独立于现场设备运行的时间尺度。

1. **设备和控制器描述**

对于操作，设备和控制器表示为虚拟设备（编程对象）。这些虚拟设备旨在用作与真实实体一起工作的抽象方法，因此 IoT-PLC 应用程序可以从高级角度与数据进行交互。为了保存和管理虚拟设备的配置和属性，使用了 json 格式，这为用户和开发人员提供了存储复杂数据类型的灵活性，而不会受到关系数据库的限制。

虚拟设备基于[23]中提出的资源对象模型，该模型旨在反映物理世界和物联网对象信息空间的集成。用于 IoT-PLC 的资源模型如图 2 所示。在该模型中，每个 IoTPLC 在 OPC-UA 服务器中都有一个文件夹对象，并分配了其控制器，在该文件夹中，每个控制器都包含属性和变量，以及虚拟代表传感器和执行器的设备（现场设备）。

对于现场设备，有两种描述，一种由 IoT-PLC 内部使用，另一种通过 OPC-UA 由云端使用。第一种是图2中带有三个青色方块的设备资源，其含义如下：''attributes''是一个不会正常变化的值（如制造商或规格），''status''是指某些参数在某个时间的值，“控制”是指设备提供的访问控制接口信息。第二种描述包括三个青色方块并添加绿色方块，其中“历史”是指设备过去上传的操作或数据。

1. **管理器**

管理器进程负责使用现场传感器和执行器生成控制回路，并以容器的形式实例化控制器以进行低级监管控制。

管理器块负责从数据库中的设备和控制器的描述中获取数据以实例化虚拟设备和控制器，它们具有组合在一起形成控制回路的能力。虚拟设备、控制器和控制回路在管理器块中被表示并用作编程对象。

当创建控制循环时，管理器在专用于控制任务的容器内启动一个进程，将虚拟设备和控制循环传送到该进程，因此循环对象可以独立操作并将生成的数据发送到数据库块管理者。

这样，如果需要更改一个或多个参数，管理人员可以随时修改控制回路，或者可以停止控制过程，而不会干扰其他控制回路或 IoT-PLC 块。

1. **Human–machine interface**

将人机界面 (HMI) 添加到 IoT-PLC 中，为系统的本地或外部用户提供配置控制算法中涉及的所有参数的选项，并从只要给予适当的授权，任何地方。

HMI 只有一个带有数据库块的数据流，因此，在从该块进行修改的情况下，管理器和控制过程可以交互。当HMI需要数据时，它向数据库块发出请求，然后数据库检索信息，最终由HMI呈现给用户。

**9、启动和恢复系统**

启动和恢复系统 IoT-PLC 的启动过程从云向设备中的脚本发出信号开始，以启动包含 IoT-PLC 每个块的进程的容器，以防它们没有它们，它们可以在本地构建、从附近的 IoT-PLC 或直接从云端获取。

启动时容器的顺序，如图 3 所示，然后定期调用相同的脚本以检查每个块是否正在运行，如果块已停止，则立即重新启动以保持操作干净尽可能。

**10、IoT-PLC的操作流程**

IoT-PLC 的操作遵循图 4 所示的时序图。控制过程从云端发送要使用的现场设备的配置开始。然后，IoT-PLC 将开始寻找设备，如果它们可用，它会以广播消息的形式接收可用性指示符。然后 IoT-PLC 指示现场设备它已被分配可用状态。

下一步是在控制器中实例化一个虚拟设备并建立连接。

当从云或本地添加控制器的配置参数并且可选地，云订阅事件时，该过程的下一部分开始。基于这些参数，IoT-PLC 实例化一个虚拟控制器，如果存在必要的传感器和执行器，则实例化一个虚拟控制回路。然后，启动来自传感器的数据流，控制器开始向执行器传输控制信号。过程数据也被发送到数据库和云端。

控制过程的第二种情况是，当设备配置数据已经存储在 IoT-PLC 中但设备不存在时，IoT-PLC 等待现场设备可用，然后开始实例化其虚拟实体。

**11、IoT-PLC 之间的容器迁移**

IoT-PLC 的设计允许使用有状态的实时迁移过程在两个设备之间迁移容器。为此，云负责编排迁移过程，这涉及两个设备：一个带有运行控制过程的容器及其分配的设备，另一个没有容器。

图 5(a) 显示了迁移过程的初始状态。

迁移的操作流程首先选择一个空闲的 IoT-PLC 作为迁移目标，它必须具有部署所有容器所需的资源。然后，云端通过 OPC-UA 向第一个 IoT-PLC 设备发送指令，启动迁移过程。第一个 IoT-PLC 负责 (i) 通过 CoAP 发现验证第二个 IoT-PLC 可用； (ii) 使用原始容器生成存储库，没有内存状态。然后，云端命令第二个 IoT-PLC 从第一个 IoT-PLC 获取容器，当此任务成功完成时，通知云端。

图 5（b）显示了这个中间阶段的状态。

接下来，从云端，为了保存容器自启动以来的进度，启动内存状态过程，这称为容器检查点。然后，这些状态被传输到第二个 IoT-PLC 并立即启动。在检查点过程中，每个容器都会暂停以获取内存状态，然后恢复以尽可能长时间地保持系统运行。

最后，云命令第一个 IoT-PLC 与其分配的设备进行通信，以将数据流处理迁移到第二个 IoT-PLC 的接口。最后，容器在第一个 IoT-PLC 处停止。从此时起，控制过程在第二个 IoT-PLC 继续进行。该过程的最终状态如图5（c）所示。

**容器的好处**

容器故障会降低性能，但一旦管理器重新启动容器，系统就会迅速恢复，而无需完全重启设备。无容器设备出现故障只能重启设备。

容器化和发布/订阅通信的使用有助于避免级联故障，将影响限制在 IoT-PLC 的单个功能块。

可以实时迁移。

图4

设备配置数据：json文档传输，

设备搜索：等待广播消息

可用性指示器：设备发送广播消息通信其可用性

额外数据配置（如果loT PLC尚未存储其配置）

虚拟设备实例化：根据资源对象模型设置属性、状态、控制和历史记录

连接建立：loT PLC向设备指示其已被分配可用状态

控制器参数和事件订阅（json配置传输和可选的OPC-UA变量订阅）

控制器参数（从本地数据库获取的json配置）

虚拟控制器实例化（设置了控制器属性和变量）

虚拟控制回路实例化：如果有必要的现场设备，则将其虚拟设备分配给虚拟控制器

测量开始：loT PLC启动控制过程并等待传感器测量

测量和设备状态从传感器发送到loT PLC

处理传感器测量并向致动器发送控制信号

致动器在收到控制信号后发送其状态

添加一个时间戳并格式化后，过程数据保存在本地数据库中

预处理的数据被上传到云，如果云订阅了一个流程变量事件，那么数据也被上传到它

设备发送广播消息，告知其可用性

lOT PLC在本地数据库中搜索设备配置

设备参数（json配置从本地数据库获得）

虚拟设备实例化：根据资源对象模型设置属性、状态、控制和历史记录

连接建立：loT PLC向设备指示其已被分配可用状态