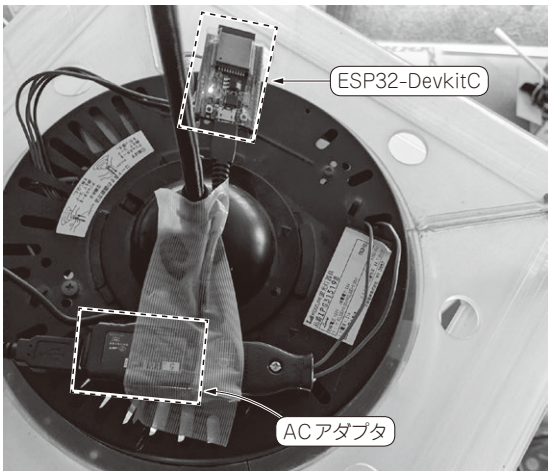
リアルタイムOS新時代に突入！
クラウド接続もスタンドアロンも

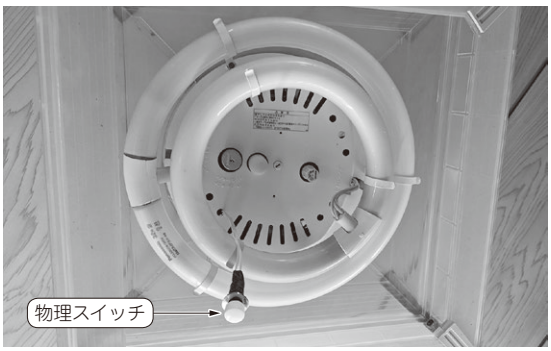
Amazon ×マイコン FreeRTOS入門

第4回　AWSで蛍光灯をIoT化する…［後編］アプリケーションの作成

小池 誠



（a）電源とESP32-DevKitCの実装部（上側）



（b）点灯／消灯スイッチ（下側）

写真1　製作したIoT蛍光灯

本連載では，リアルタイムOS初心者向けに，実際にマイコンを動かしながらFreeRTOSの基本的な使い方や製作事例を紹介します．

前回（2021年7月号）から，Wi-Fi機能付きマイコンESP32（Espressif Systems）とアマゾン ウェブ サービス（Amazon Web Service，略称AWS）を使って，一般家庭によくある蛍光灯をインターネット経由でプログラマブルに操作できるIoT蛍光灯を製作しています（**写真1**）．

後編では，前編で製作したハードウェアを動かすアプリケーション・ソフトウェアを作成します．その後，実際にAWSのクラウド・サービスと接続して動作確認を行います．〈編集部〉

システム構成

● 全体構成

図1に示すのは，今回製作するIoT蛍光灯のシステム構成です．

デバイスシャドウ・サービス（コラム参照）を使って，「蛍光灯の点灯状態」と「蛍光灯への点灯要求」の2つの情報をやり取りします．

デバイスシャドウはJSONフォーマットで記述され，豆電球（mame），30W蛍光灯（light1），32W蛍光灯（light2）の3つのモノごとに状態と要求を持ちます．蛍光灯には物理スイッチを接続し，もしデバイスシャドウ・サービスとの通信が不能になっても手動で点灯／消灯ができるようにしておきます．

リモート・アプリケーションからは，点灯／消灯の要求のみを送信します．蛍光灯のデバイスからは，物理スイッチによる点灯／消灯の要求と自身の状態を送信します．

● 状態遷移設計

図2に示すのは，蛍光灯デバイスの状態遷移図です．

物理スイッチが操作された場合は，押下するごとに消灯→全点灯→半点灯→豆点灯→…を繰り返します．リモート・アプリケーションからの点灯要求があったときは，どんな状態でもリモート点灯状態に遷移します．デバイスのリセット復帰時にはデバイスシャドウを取得し，要求（desired）に従って状態を決定します．

デバイスシャドウの更新は，全ての状態遷移が発生したタイミングで行っています．

▶プロジェクト・ファイルの入手方法

本アプリケーションのプロジェクト・ファイル一式は，本誌ウェブ・ページよりダウンロードできます．