Flicker是一个可扩展、模块化、“即插即用”的架构。

* 支持多种能量收集方式：RFID、太阳能、动能。
* 支持主动和被动的无线通信。
* 支持广泛的传感器。
* 支持先进技术：容错计时、分散能量存储。
* 支持动态的能量优先级。

电池体积大、重量重、价格昂贵、寿命短，导致维护成本和环境成本高。

无电池计算系统对开发者的挑战：

能量故障频繁且无法预测。数据处理、传感和通信经常被干扰，存储器的内容会丢失。

相关文章：

检查点（checkpointing）

【2】2015

Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems.

【29】2011

Mementos: System Support for Long-Running Computation on RFID-Scale Devices.

一致性（consistent execution）

【11】2016

Chain: Tasks and Channels for Reliable Intermittent Programs.

【24】2015

A Simpler, Safer Programming and Execution Model for Intermittent Systems.

计时（timekeeping）

【21】2016

Persistent Clocks for Batteryless Sensing Devices.

能量管理

【20】2015

Tragedy of the Coulombs: Federating Energy Storage for Tiny, Intermittently-Powered Sensors.

测试

【19】2014

Ekho: Realistic and Repeatable Experimentation for Tiny Energy-Harvesting Sensors.

调试

【12】2015

Energy-interferencefree System and Toolchain Support for Energy-harvesting Devices.

WISP的诞生（2008）：【31】

Design of an RFID-based battery-free programmable sensing platform.

及其后代（2011）：【35】

Moo: A batteryless computational RFID and sensing platform.

有电池的IOT设备难维护，成本高，需要很专业的人员进行开发，硬件高度定制，难以迭代

论文解决的挑战:（设计无电池传感系统的挑战）

**有限的硬件选择**：WISP只支持RFID能量采集，如果需要其他能量形式，要么修改WISP以满足需求, 要么从头创建新的硬件。（引文[8]就是魔改WISP）

**有限的灵活性：**现有的无电池平台集成度高，将能量收集、能量管理、传感、数据处理和通信紧密集成在一块电路板上，很难修改。

**缺乏现代设施（Lacking Modern Amenities）：**其实就是缺乏最新开发的功能，比如零功耗计时，分散式储能（Federating Energy Storage），也叫UFoP。

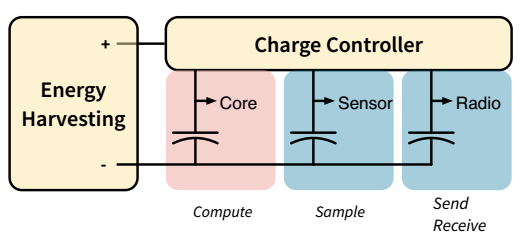
[21]硬件辅助的零功耗计时（2016）Persistent Clocks for Batteryless Sensing Devices.

计时的重要性：比如说安全方面的DOS攻击就是通过计时功能来防御的。

UFoP与集中式储能相对应。

UFoP: 将收集到的能量自动划分为多个独立的小型能量存储并进行优先级排序.

UFoP是这篇文章的作者在2015年提出来的，也就是引文[20]



**可用性或社区性差：**新手开发者很难构造有用的设备或验证它们是否工作，没有一个好用的平台，开发者社区就不会出现。

Flicker: 一个用于能量收集，间歇供电，无电池传感设备的模块化硬件平台。

Flicker支持UFoP, 允许开发人员以可编程方式设置唤醒点和任务触发点，并更改单个传感器和其它外围设备的相对优先级。还支持故障阻抗计时。

贡献：

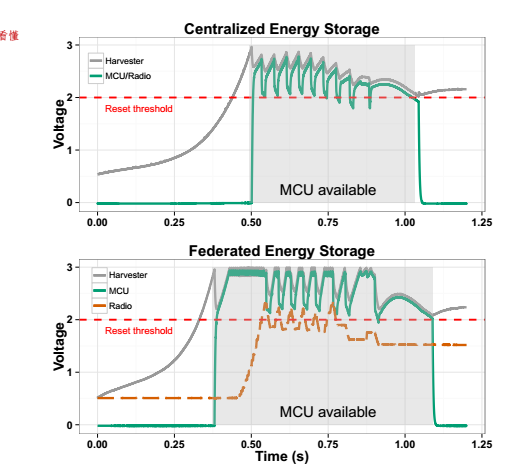
贡献1:模块化, 可扩展的硬件平台, 无电池可能量收集的传感器

贡献2: 扩展了联邦能量存储技术, 自动管理外围设备, 在运行时可动态重新分配任务和重新确定能量资源的优先级.

UFoP

每个外设使用单独的电容供电。在设计硬件设备时，为每个电容器设置电压阈值。微控制器（轮询地）监控每个电容器的充电状态，当所需的外设充电充足时，执行与外设相关的任务。

图三尝试解释联合能量存储和集中式能量存储对系统运行的不同影响，但是并没有看懂。



现有的UFoP的缺点：

特定程序的硬件设计是脆弱的：

开发者需要确定外设电压阈值，电容容量和充电阈值是静态的，硬件成型后不可改变。任务的优先级一旦部署就不能更改。导致系统的软硬件依赖性很强。

静态的UFoP在运行时不灵活：

不在需要使用的外设不能关闭，会继续充电储能，阻碍了更重要的任务的运行。外设的优先级不能在运行时改变。从根本上限制了应用程序的复杂性和现场重新分配传感器任务的能力。

显著的硬件复杂性依然存在

从电压调节、逻辑电平和能量管理中体现出来。不同的外设对电压有不同的需求，当前UFoP的解方案时不管理MCU和外围设备之间通信的逻辑电平，设计者必须同时仔细调整电容和能量采集器，使电压电平保持在逻辑范围内。另外，UFoP的能量管理基于**轮询**方法来管理能量级别，这会在频繁的阈值检查中浪费能量。

本文如何解决无电池感知设备原型设计的挑战?

1. 增加联合存能量存储的灵活性和效率

2. 将这些改进集中到一个新的通用平台(flicker)

flicker的目标:(对应于四点挑战)

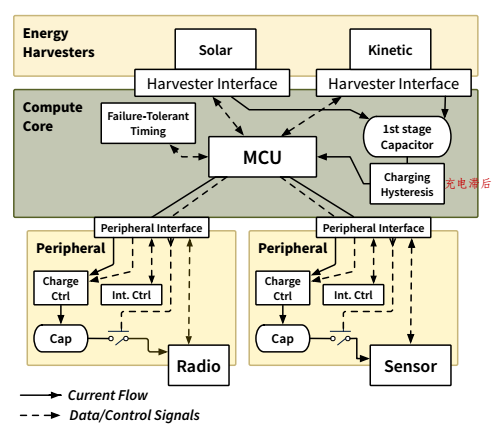
1. 提供多种外围设备和采集技术的硬件选择

2. 实现运行时和设计时的灵活性

3. 在通用平台上实现计时和能量管理的最新技术

4. 提供一个专注于软硬件可用性的平台

Flicker的硬件架构：



能量的收集采用多种方式（太阳能、动能），采集到的能量汇入第一级电容器，为整个板卡供电。外围设备通过标准接口连接，接口分为电源可数据/控制信号。

Flicker由计算核心、外围设备和能量收集器构成。

计算核心：包括微控制器，计时功能，外设接口和能量接口。MCU建议使用MSP430 FRAM系，能很好的与能量采集器配合使用并支持有效检查点（efcient checkpointing）（并不懂这是什么）。

外围设备：外设与外设端口连接，外设端口提供模拟信号线、数字信号线（IIC、SPI、UART）、电源线和控制信号线。外设可以射频设备、传感器。每个外设都存储自己的能量，并包含控制能量如何存储和使用电路。

能量采集器：提供电压足够高的直流电源。

一个关键创新：可重新配置的联合能量（Reconfgurable Federated Energy）：

可重新配置是指程序员（或编译器）能够在编译时或运行时为每个外围设备分配要收集的能量、充电优先级以及满足任务执行条件的能量阈值。

实现可重配置的联合能量机制的基本哲学（philosophy）：

（1）不使用轮询，使用由定制硬件生成的基于中断的“外设就绪”信号。

（2）使用数字可编程电阻分配器来改变电压阈值。

（3）根据外围设备使用专用的电压调节器（？？）

其实就是每个外设有自己的储能电容以及一个可编程充电控制器。只有当电容的输入电压达到MCU设定的特定阈值时，对电容器充电（允许分配优先级，具有较高优先级的外设在较低的电压阈值下开始充电）。使用可编程中断控制器，当充电水平超过设定的阈值时，向MCU发出中断信号（而不是前面提到的轮询）。

外设接口：由于引脚的稀缺，一些接口可能只能在外设端口的子集上提供一些有限的硬件资源，并且可能不支持一些需要过多控制信号的外设，该挑战的应对在第4节。

容错计时：

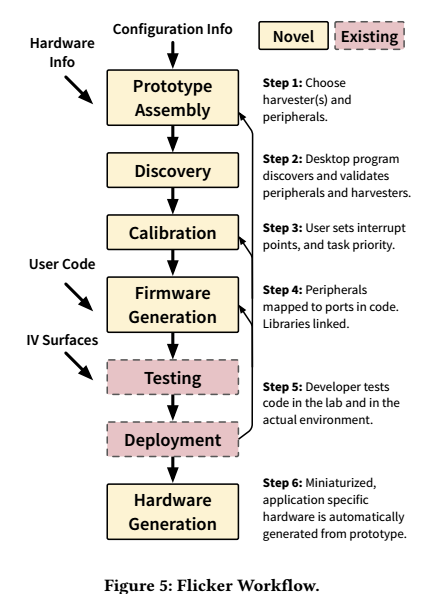
就是整个系统在发生故障（比如没电了）以后还具备计时功能。

计时对于感知、安全性、数据来源和数据使用非常重要。还可以提升用户的使用体验。

可用的技术：基于剩磁的SRAM定时器，使用了SRAM存储单元的衰减特性；具备热特性的电容器，但精度会随着停机时间延长而降低；独立供电的RTC，但会增加成本和硬件空间。

自动探测配置

Flicke可以自动探测到哪个模块连接到了哪个端口。个模块用电阻值来标识，该电阻值由MCU上一个特定的校准固件来测量。自动检测过程并不节能，自动检测过程只在校准和测试期间进行，而不是在运行时进行。

Flicker的工作流程：  


1. 原型组装：开发者根据应用需求和直觉，从可用的外设和采集器中选择一种特定的配置。然后组装外设、采集器形成一个Flicker设备草稿。

2. 发现过程：核心模块连接到编程器后，发现过程会检测到连接什么外设和收集器，生成配置文件，并检测不兼容的连接。配置文件包含外设和端口之间的映射，电压阈值等。开发者可根据自己的需求更改配置文件。

3. 校准：根据应用程序优先级、开发人员的初始设定调整阈值

4. 5. 将开发者代码与库代码和配置文件相结合，编译连接并安装在设备进行测试和部署。

6. Flicker生成原理图和电路板布局，开发者可以自由开发。

实现

硬件设计：

计算核心：MSP430FR5989，128K FRAM， 2K SRAM， 通信接口和模拟接口。 3个外设插槽，每个都支持SPI，两个支持UART，一个支持IIC。配置10uF电容的RTC，

通用外设接口：每个外设都有充电和优先级控制器和中断控制器，储能电容，ID电路，能量开关。

环境感知

运动感知

通信

用户接口

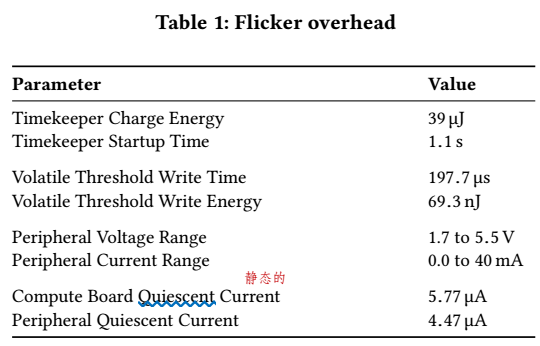
能量采集

软件设计：

探测和校准：由python脚本和自定义固件负责探测和校准过程。用户搭建好固件以后，特定的探测固件上传到MSP430，该固件通过ADC读取每个外设和收集器插槽上的电压，识别连接在计算核心上的模块。电压值将会存储在MSP430上一块预先约定好的空间。MSP430进入等待模式，在桌面端，一旦等待模式开始，python脚本从MCU上获取电压值，使用电压值进行查表，就可知得到外设的型号。python脚本向用户输出配置信息，如果外设不兼容也会输出警告。工具链进行错误检查以后，使用配置文件中断和充电的阈值。

运行时库：针对外设模块开发了运行时库。这些库包含模块的最低功能，允许执行基本的感知和通信任务。用户可以调用API更改中断和充电优先级。

设计自动化：使用python脚本额EAGLE CAD来生成硬件原理图和配置文件，是开发人员可以脱离Flicker硬件平台进行开发。

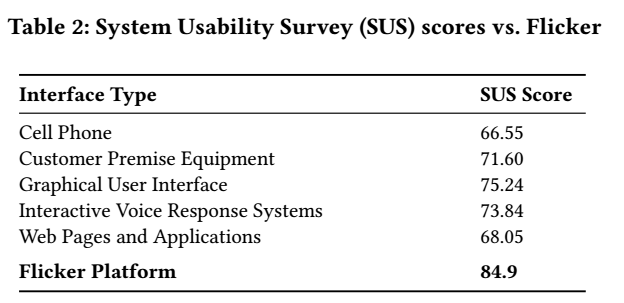


性能

射频能量采集：瞬时功率最大为1.4mW, 最大电压6.3V，最大电流1.0mA。与UMich Moo收集器和WISP相媲美。

计时：使用极低功耗RTC，10uF的电容充电1.1s，可支撑RTC工作19‘40’‘

用户学习：19个学生建立76个设备使用了9.5小时



70分是平均分，结论：Flicker的可用性高于平均水平