# CARTOS OPERATING SYSTEM

**Group members:**
**9881-Shwen Coutinho**
**9883-Bournerich Dcunha**
**9900-Jonathan Gomes**
**9913-Mark Lopes**

# INTRODUCTION

Batteryless systems, often called intermittently-powered devices (IPDs), have the potential to revolutionize a variety of applications in Internet-of-Things (IoT), such as smart healthcare, agriculture, and building monitoring, where the use of batteries is undesirable or incurs significant maintenance costs.

These systems harvest energy from ambient sources such as sunlight, heat, and radio signals, charge a small energy buffer, and execute intermittently for a short period of time whenever the energy is available. Since no batteries are needed, they have a much longer lifespan than batterypowered systems and can be deployed in harsh environments where batteries cannot operate properly, e.g., due to cold and hot temperatures

**These mentioned benefits, however, come with several technical challenges.**

# Challenges

- Given the variability in energy availability from ambient sources, IPDs must ensure the forward progress of program execution across power failures.

- IPDs should provide accurate time behavior even in the presence of intermittent power disruptions. This is particularly important for periodic sensing applications. For instance, a power failure during the execution of a sensor-reading or data-logging task can result in incomplete task execution or the loss or corruption of data.

In this paper, we propose CARTOS, a charging-aware realtime operating system, as a practical platform to develop energy-resilient and reliable real-time sensing applications on IPDs.
To achieve forward progress guarantees without losing efficiency, CARTOS classifies tasks into two types, computation tasks and peripheral tasks, and introduces a mixedpreemption scheduling model.
Computation tasks are suspendable and resumable at any point of execution; hence, CARTOS schedules them preemptively and uses JIT checkpointing to store their states.
On the other hand, since peripheral tasks are not JIT-checkpointable, CARTOS checks before their execution if the system has enough energy to complete them, and it schedules them non-preemptively to prevent interruption.
CARTOS also adapts task scheduling based on environmental changes and ensures correct and timely execution under various conditions.

**The operation of an IPD can be broken down into the following three phases:**

1. Power-Off Phase: In this initial phase, the device remains entirely powered off. It actively collects energy from the environment using an energy harvester and subsequently stores the gathered energy in a capacitor.
2. Operation Phase: Once the capacitor attains a sufficient voltage level, the PMU provides a consistent voltage supply to the device. This, in turn, enables the MCU to execute processing tasks and capture sensor data. The device continues in this phase until it reaches the predefined power-off threshold (going back to the poweroff phase) or low-voltage threshold (moving on to the standby phase discussed next) depending on the design.
3. Standby Phase: In this phase, energy-intensive components like the MCU, sensors, and radio units are either powered down or transitioned into sleep mode. Only the timekeeping ability, e.g., MCU’s clock unit in lowpower mode or external programmable RTC, remains active as it plays a crucial role in maintaining the precise time behavior of tasks. The device remains in this state until the capacitor accumulates enough charge to return to the operation phase.

# Checkpointing for Reliable Operation of Batteryless Devices

| Methods | JIT checkpointing | Peripherals | Long computations | Real-time scheduling | Energy adaptive | Analysis |
|---|---|---|---|---|---|---|
| Mementos [6] | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ |
| Chinchilla [4] | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ |
| Capybara [24] | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| Quickrecall [8] | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ |
| Samoyed [10] | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ |
| InK [23] | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ |
| CatNap [9] | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ |
| Celebi [17] | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ |
| Rtag [16] | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ |
| Karimi et al. [27] | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ |
| CARTOS (this work) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

- Checkpointing is a technique for saving the state of a program at certain points in time.
- This allows the program to be resumed from that point if it is interrupted, such as by a power failure .
- In the context of IPDs, checkpointing can be used to ensure that the device can continue its task even if it runs out of power and then harvests more energy

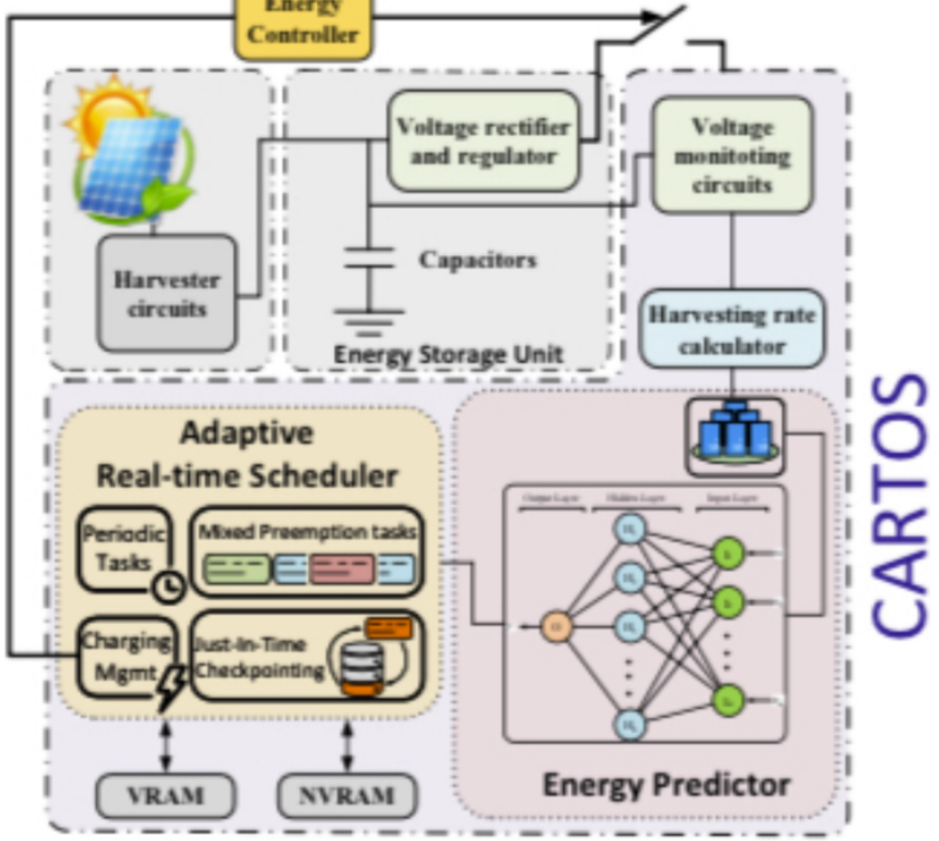
# ARCHITECTURE

## Hardware components

- solar harvester
- capacitor
- voltage rectifier and regulator

## Software components

- an adaptive real-time scheduler
- neural network-based energy predictor
- mixed-preemption model




## CARTOS

- a framework for managing the power consumption of IoT devices.
- works by predicting how much energy the device will be able to harvest in the near future and then scheduling tasks accordingly
- also gives priority to peripheral tasks over computational tasks

# FRAMEWORK

## 1. PERIODIC TASKS SUPPORT

- CARTOS PROVIDES APIS FOR PROGRAMMING AND EXECUTING PERIODIC TASKS.
- USERS CAN DEFINE TASKS AS PERIODIC AND SPECIFY THEIR PERIODS AND RELEASE OFFSETS.
- RELEVANT APIS INCLUDE `SETTASKPERIOD(PERIOD, OFFSET)` TO DECLARE A TASK AS PERIODIC AND `WAITFORNEXTPERIOD()` TO INDICATE THE COMPLETION OF THE CURRENT PERIOD.

## 2. CHAINED TASK EXECUTION

- CARTOS ALLOWS USERS TO CREATE CHAINS OF TASKS AND CONTROL THEIR EXECUTION ORDER.
- TASKS WITHIN A CHAIN ARE REMOVED FROM THE READY QUEUE AND MADE INACTIVE UNTIL TRIGGERED BY A TIMER OR EXTERNAL EVENT.
- UPON COMPLETION OF A TASK IN A CHAIN, THE SUBSEQUENT TASK IS ACTIVATED FOR EXECUTION.

# 3. ADAPTIVE REAL-TIME SCHEDULER

- CARTOS'S REAL-TIME SCHEDULER EFFICIENTLY UTILIZES AVAILABLE ENERGY AND SUPPORTS BOTH PERIPHERAL AND COMPUTATIONAL TASKS.
- IT PRIORITIZES HIGHER-PRIORITY TASKS OVER LOWER-PRIORITY ONES TO MEET TIMING REQUIREMENTS.
- UPON DEVICE BOOT-UP, THE SCHEDULER CHECKS FOR A VALID CHECKPOINT IN NVM AND RESTORES TASKS IF FOUND.
- I

**OVERALL, CARTOS'S SCHEDULING MECHANISM ENSURES TIMELY EXECUTION OF TASKS, OPTIMALLY UTILIZING SYSTEM RESOURCES AND PRIORITIZING HIGHER-PRIORITY TASKS WHILE SUPPORTING PERIODIC AND CHAINED TASK EXECUTION.**

# 4. JIT CHECKPOINTING

- JIT CHECKPOINTING IS PERFORMED BY A SYSTEM SERVICE TASK.
- IT MONITORS THE VOLTAGE LEVEL TO DETECT LOW-VOLTAGE CONDITIONS.
- WHEN TRIGGERED, IT INITIATES A POWER CYCLE AND SAVES THE CONTEXT, STACK, SRAM DATA AREAS, SYSTEM TIME, AND TIMER EVENTS OF NON-ATOMIC TASKS TO NON-VOLATILE MEMORY (NVM).

# 5. CHARGING MANAGEMENT

- CHARGING MANAGEMENT IS CRUCIAL FOR EXECUTING ATOMIC TASKS WITHOUT INTERRUPTION.
- EACH ATOMIC TASK IS ASSIGNED A VOLTAGE THRESHOLD REPRESENTING THE MINIMUM VOLTAGE REQUIRED FOR ITS EXECUTION.
- THIS THRESHOLD ENSURES THAT THE TASK CAN BE COMPLETED BEFORE REACHING THE LOW-VOLTAGE THRESHOLD USED FOR JIT CHECKPOINTING.

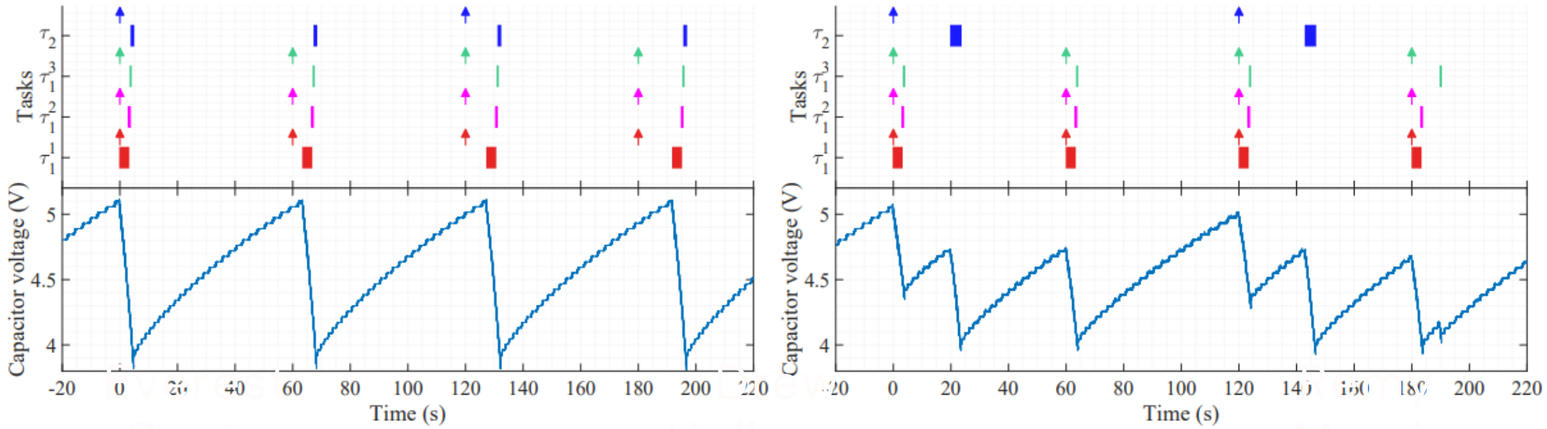
# REAL SYSTEM EXPERIEMENT

Experimental Scenario : Handling Peripheral Tasks with Long Execution Time

- Setup: The system is configured to run tasks  consisting of three tasks with high priority, where the first two are non-preemptable (peripheral) and the last task is preemptable (computational), however treated as non pre-emptable.
- **Execution Behavior Under STOS with jit-checkponting:**
  - After the execution of the first two high-priority tasks, the non preemptive atomic task (τ3) starts executing.
  - However, during τ3's execution, the device's energy level drops to the low-voltage threshold, causing it to shut down.
  - Upon reboot, new jobs from high-priority tasks are executed, but τ3 restarts its execution from the beginning each time and misses its deadline.
- **Execution Behavior Under CARTOS:**
  - CARTOS does not start the execution of τ3 immediately after the first three high-priority tasks.
  - It recognizes that the energy level is insufficient to complete τ3's execution and initiates power cycling to charge the demand of τ3.

# REAL SYSTEM EXPERIEMENT

Figure showing difference between task scheduling of STOS and Cartos OS



# EXPERIEMENT OUTCOMES

- **CARTOS outperforms** the SOTA method by up to **20%** in schedulability.
- This **improvement** is attributed to the **unnecessary blocking time** caused by lower-priority tasks in the **SOTA** method, which **treats all tasks** as **non-preemptable** even if some are fully computational tasks..

**In summary**, the **experiment** demonstrates that **CARTOS performs better** in maintaining schedulability compared to the SOTA method, particularly when dealing with tasks of varying energy demands and priorities. This is **achieved** by **optimizing** task **scheduling** and handling lower-priority tasks more effectively in CARTOS.

# CONCLUSION

CARTOS utilizes a mixed-preemption scheduling model to effectively handle periodic sensing applications, which consist of computational and peripheral tasks.
Key features of CARTOS include:

1. **Mixed-Preemption Scheduling**: CARTOS schedules ensure forward progress using just-in-time (JIT) checkpointing.
2. **Energy Management**: CARTOS incorporates energy management functionalities to calculate required charging times and adapt to changing environmental conditions
3. **Evaluation**: Results demonstrated CARTOS's ability to successfully schedule periodic task chains, outperforming state-of-the-art solutions by **up to 20%** in schedulability across various experimental settings.

# THANK YOU  FOR YOUR ATTENTION !

ANY QUESTIONS, PLEASE FEEL FREE TO ASK....