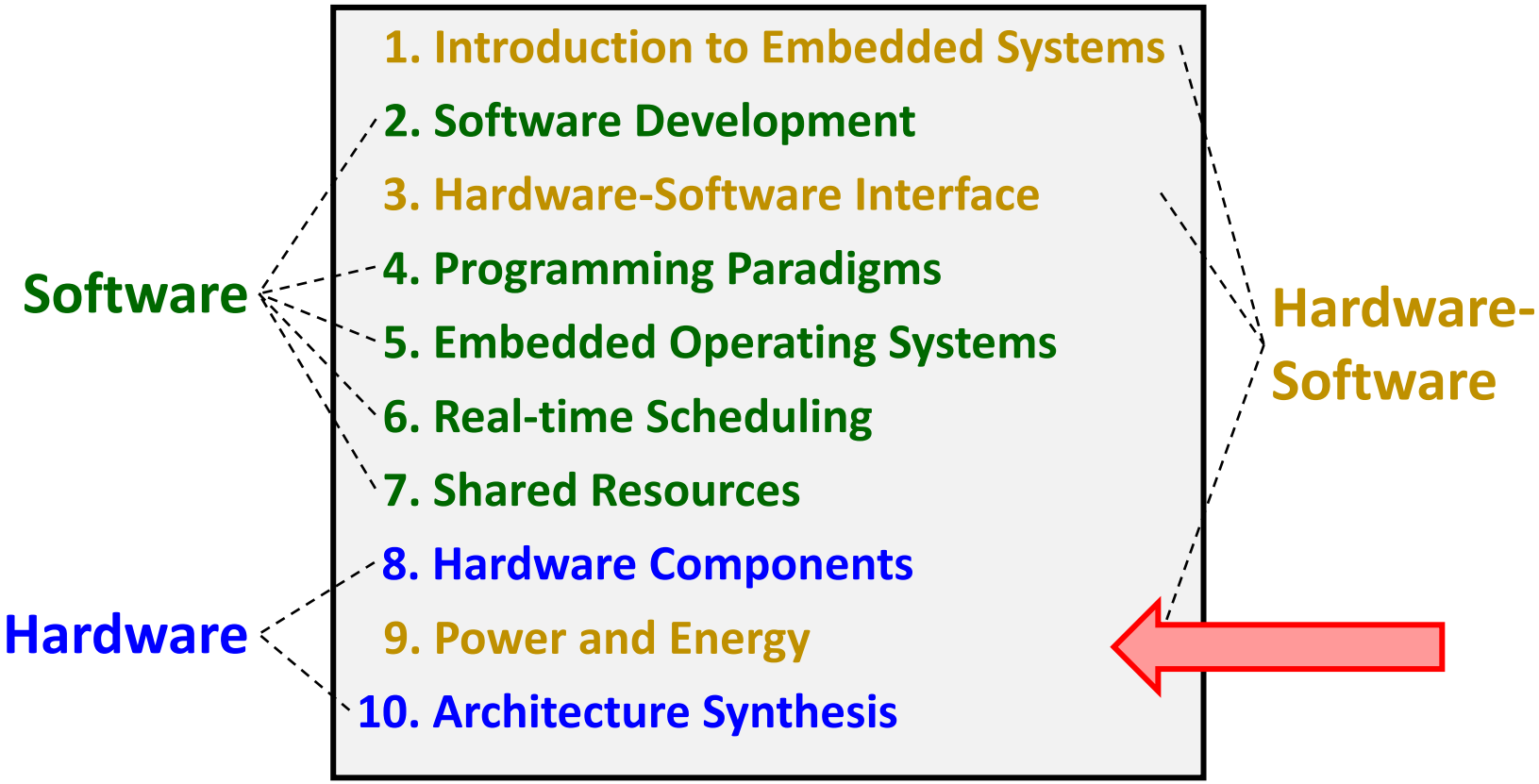
Introduction to Embedded Systems

9. Power and Energy

Prof. Dr. Marco Zimmerling

Where we are …


1. Introduction to Embedded Systems
2. Software Development
3. Hardware-Software Interface
4. Programming Paradigms
5. Embedded Operating Systems
6. Real-time Scheduling
7. Shared Resources
8. Hardware Components
9. Power and Energy
10. Architecture Synthesis
Software
Hardware
Hardware-
Software

Dynamic Power Management

Dynamic Power Management (DPM)

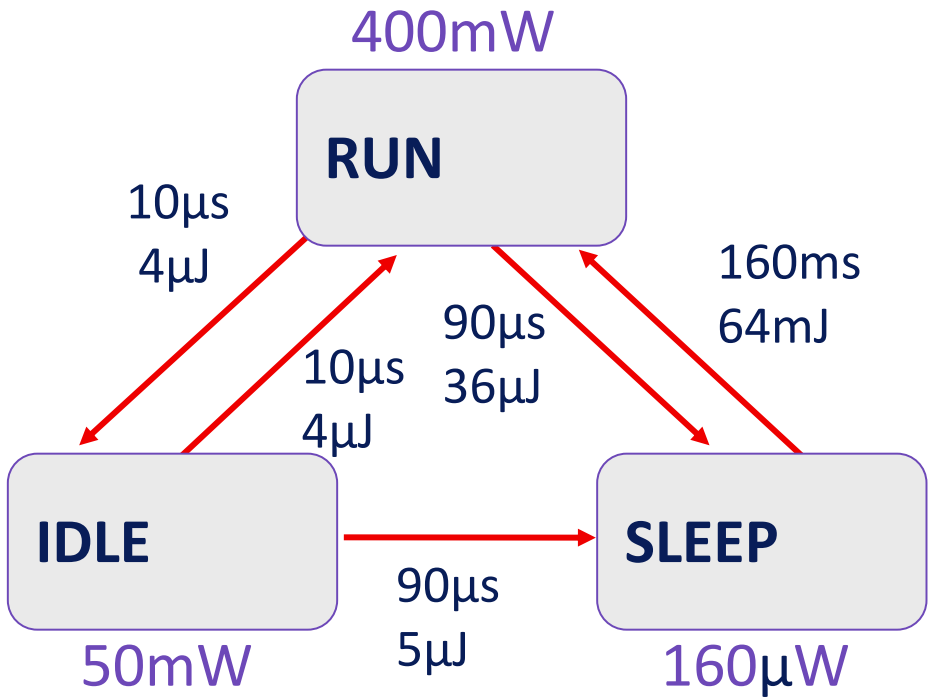
- Dynamic power management tries to assign optimal **power saving states during program execution**
- DPM requires hardware and software support

Example: StrongARM SA1100

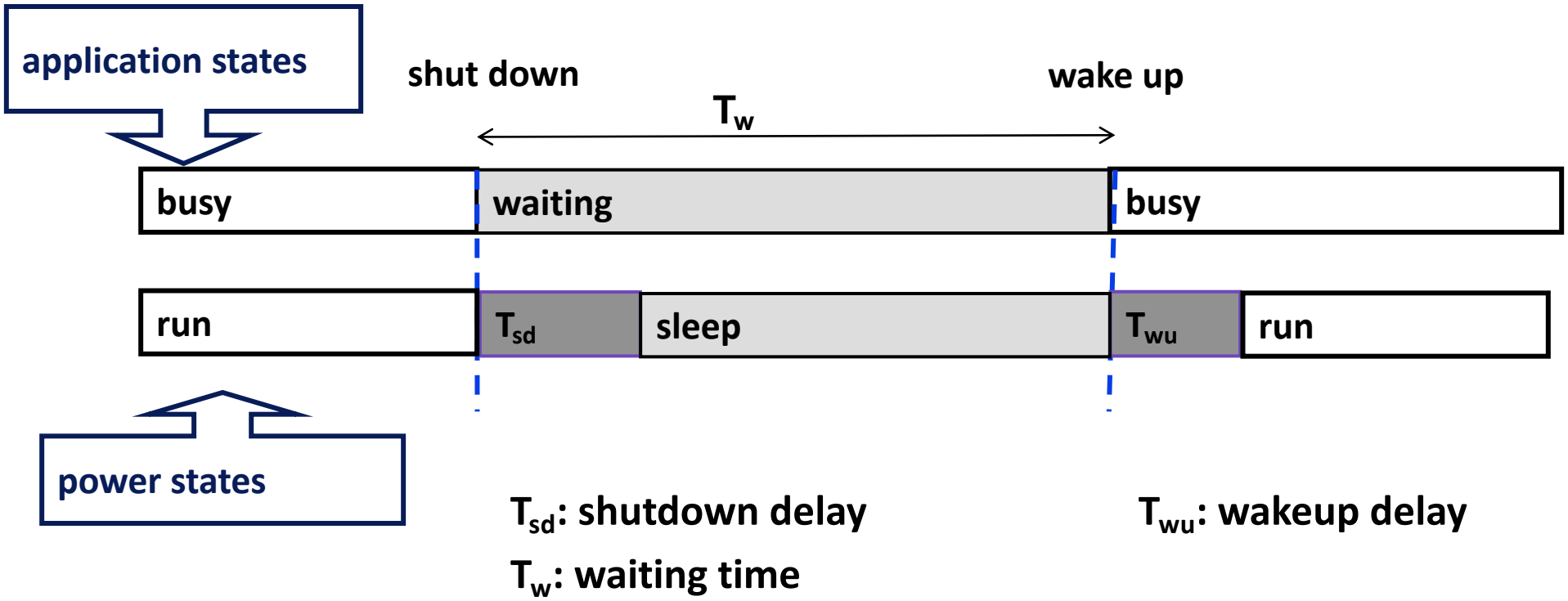
RUN: operational

IDLE: a SW routine may stop the CPU when not in use, while monitoring interrupts

SLEEP: Shutdown of on-chip activity



Dynamic Power Management (DPM)

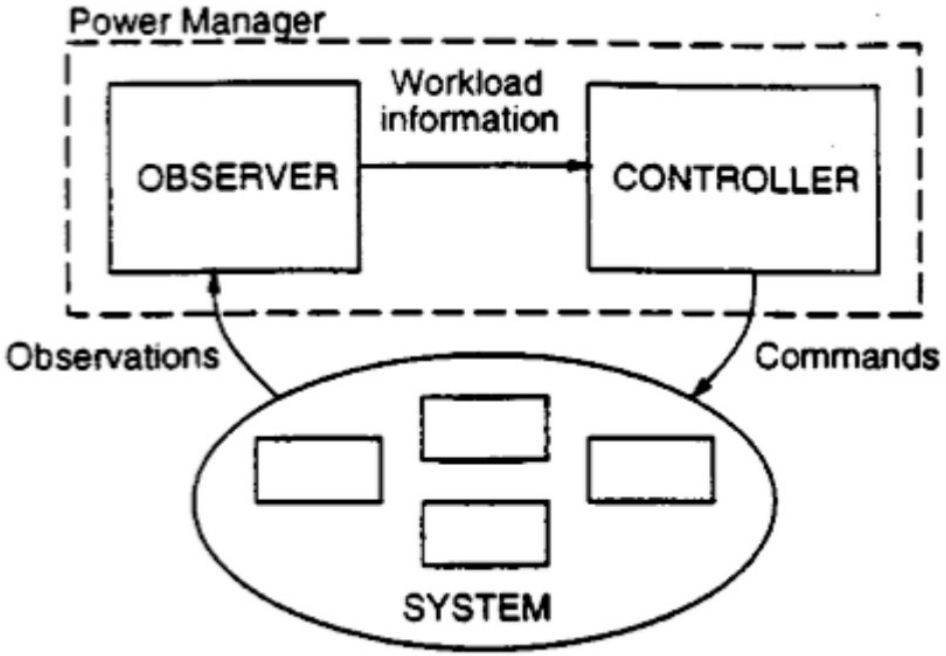


Desired: Shutdown only during long waiting times. This leads to a tradeoff between energy saving and overhead.

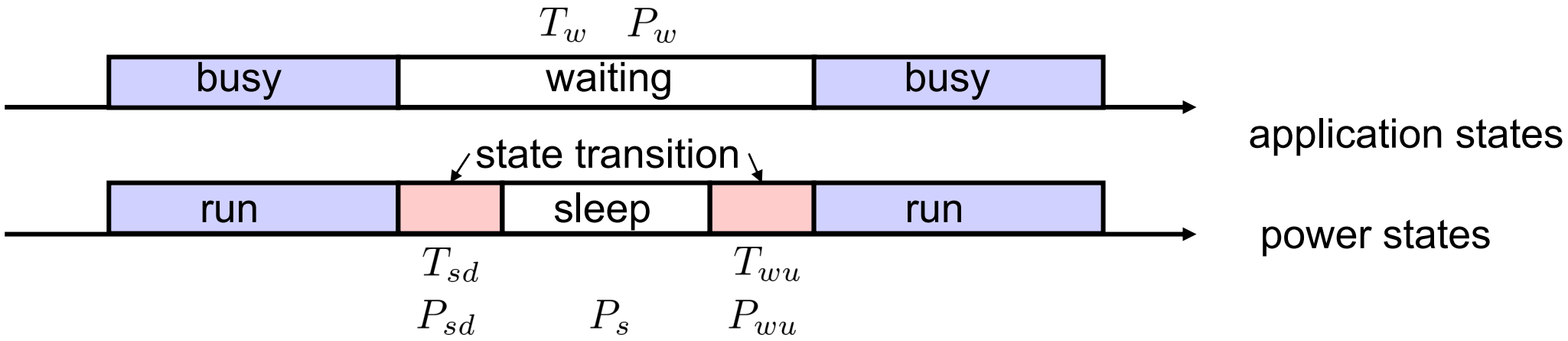
Break-Even Time

Definition: The minimum waiting time required to compensate the cost of entering an inactive (sleep) state.

- Enter an inactive state is beneficial only if the waiting time is longer than the break-even time
- Assumptions for the calculation:
  - No performance penalty is tolerated.
  - An ideal power manager that has the *full* knowledge of the future workload trace. On the previous slide, we supposed that the power manager has *no* knowledge about the future.



Break-Even Time



Scenario 1 (no transition): $E_1 = T_w \cdot P_w$

Scenario 2 (state transition): $E_2 = T_{sd} \cdot P_{sd} + T_{wu} \cdot P_{wu} + (T_w - T_{sd} - T_{wu}) \cdot P_s$

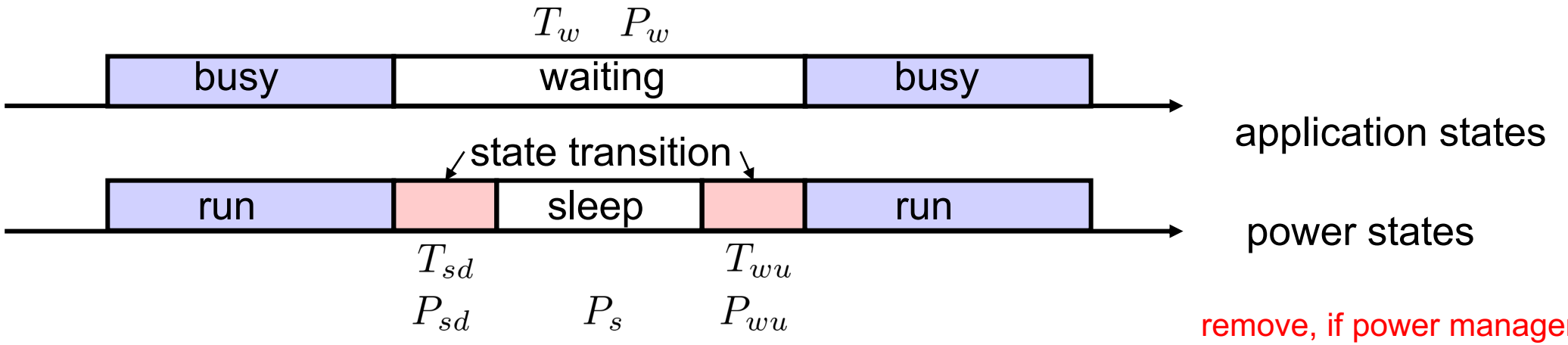
Break-even time: Limit for T_w such that $E_2 \leq E_1$

Break-even constraint: $T_w \geq \dfrac{T_{sd} \cdot (P_{sd} - P_s) + T_{wu} \cdot (P_{wu} - P_s)}{P_w - P_s}$

break-even time

Time constraint: $T_w \geq T_{sd} + T_{wu}$

Break-Even Time



Scenario 1 (no transition): $E_1 = T_w \cdot P_w$

Scenario 2 (state transition): $E_2 = T_{sd} \cdot P_{sd} + T_{wu} \cdot P_{wu} + (T_w - T_{sd} - T_{wu}) \cdot P_s$

Break-even time: Limit for T_w such that $E_2 \leq E_1$

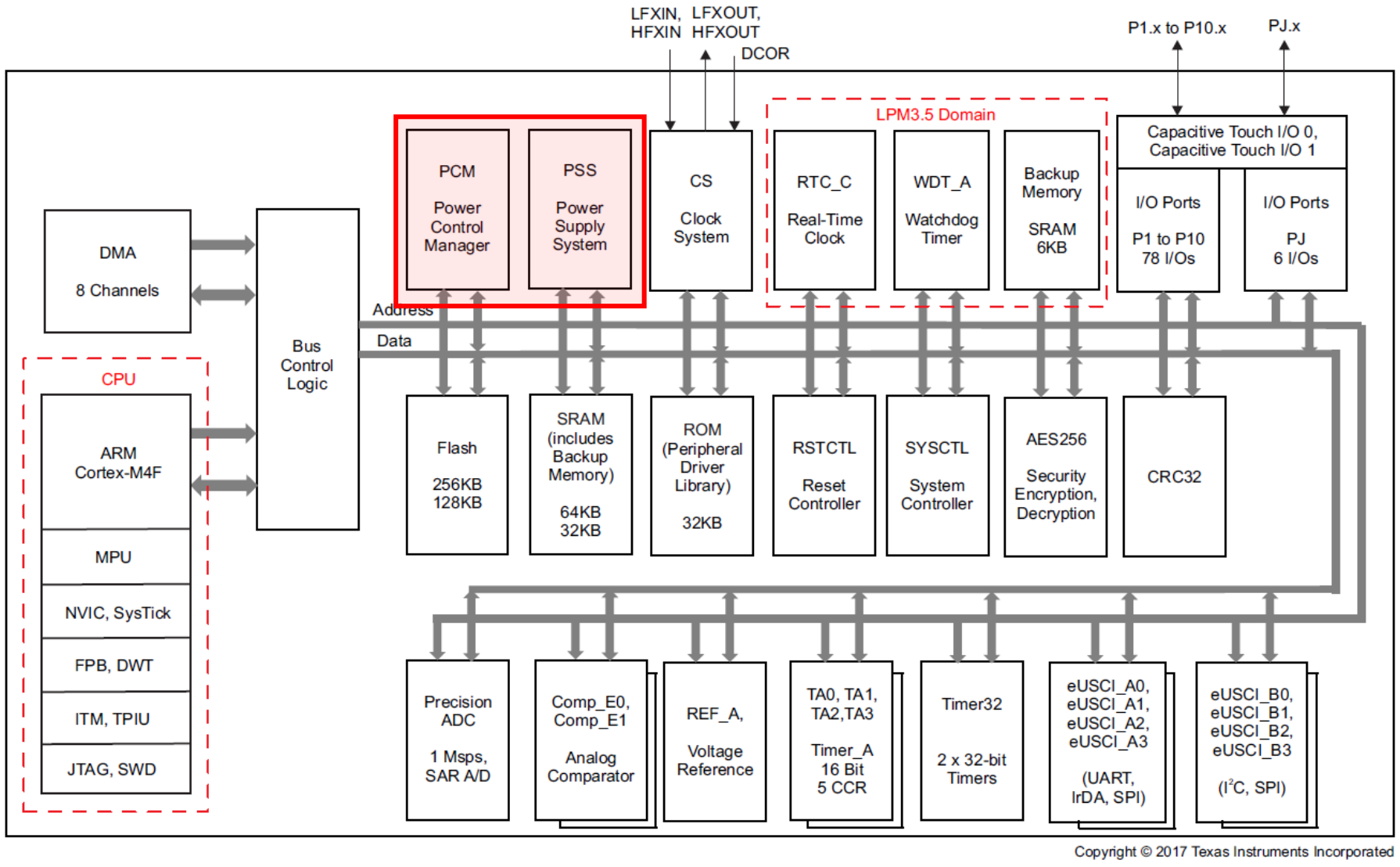
remove, if power manager has no knowledge about future

break-even time

Break-even constraint: $$T_w \geq \frac{T_{sd} \cdot (P_{sd} - P_s) + T_{wu} \cdot (P_{wu} - P_s)}{P_w - P_s}$$

Time constraint: $T_w \geq T_{sd} + T_{wu}$

Power Modes in MSP432



The MSP432 has one active mode in 6 different configurations which all allow for execution of code.

It has 5 major low power modes (LP0, LP3, LP4, LP3.5, LP4.5), some of them can be in one of several configurations.

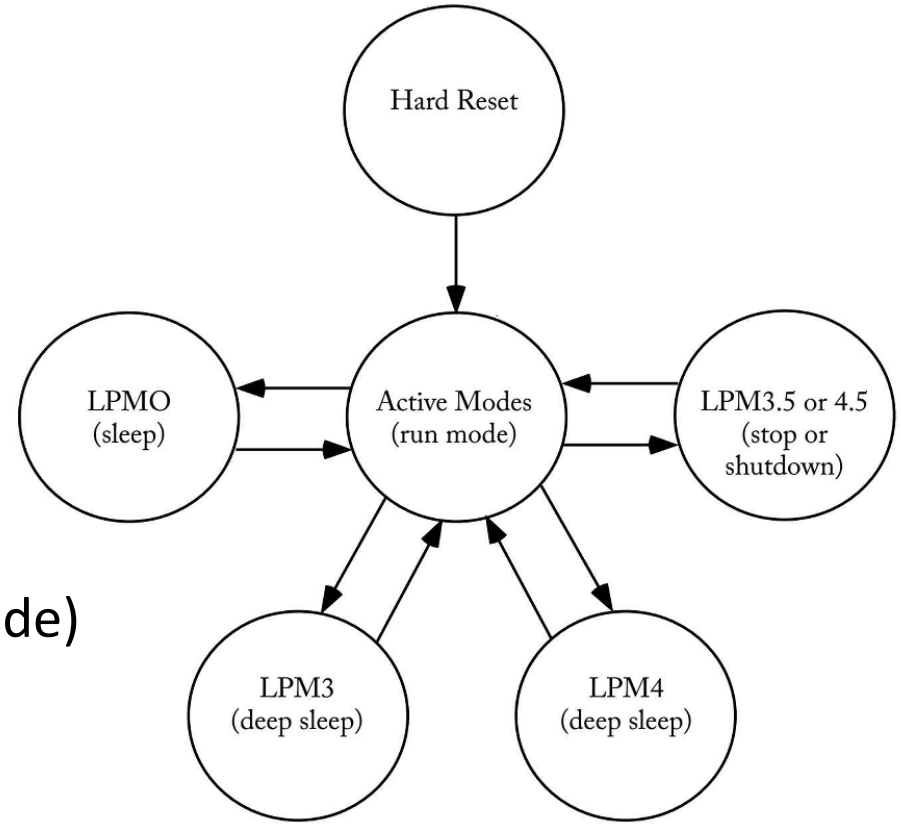
In total, the MSP432 can be in 18 different low power configurations.

active mode (32MHz): 6 - 15 mW ; low power mode (LP4): 1.5 – 2.1 µW

Power Modes in MSP432

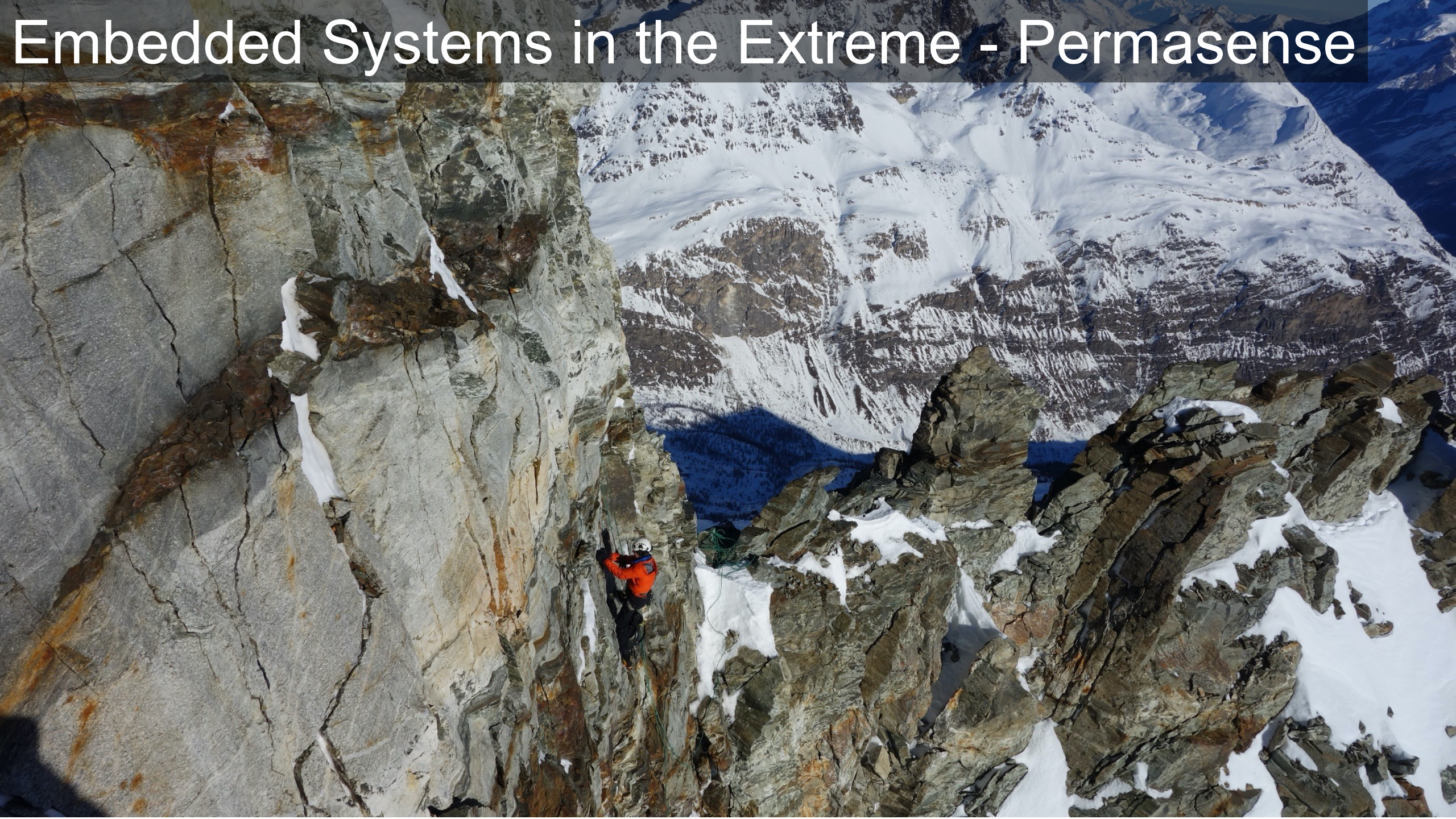
- Transition between modes can be handled using C-level interfaces to the power control manger.

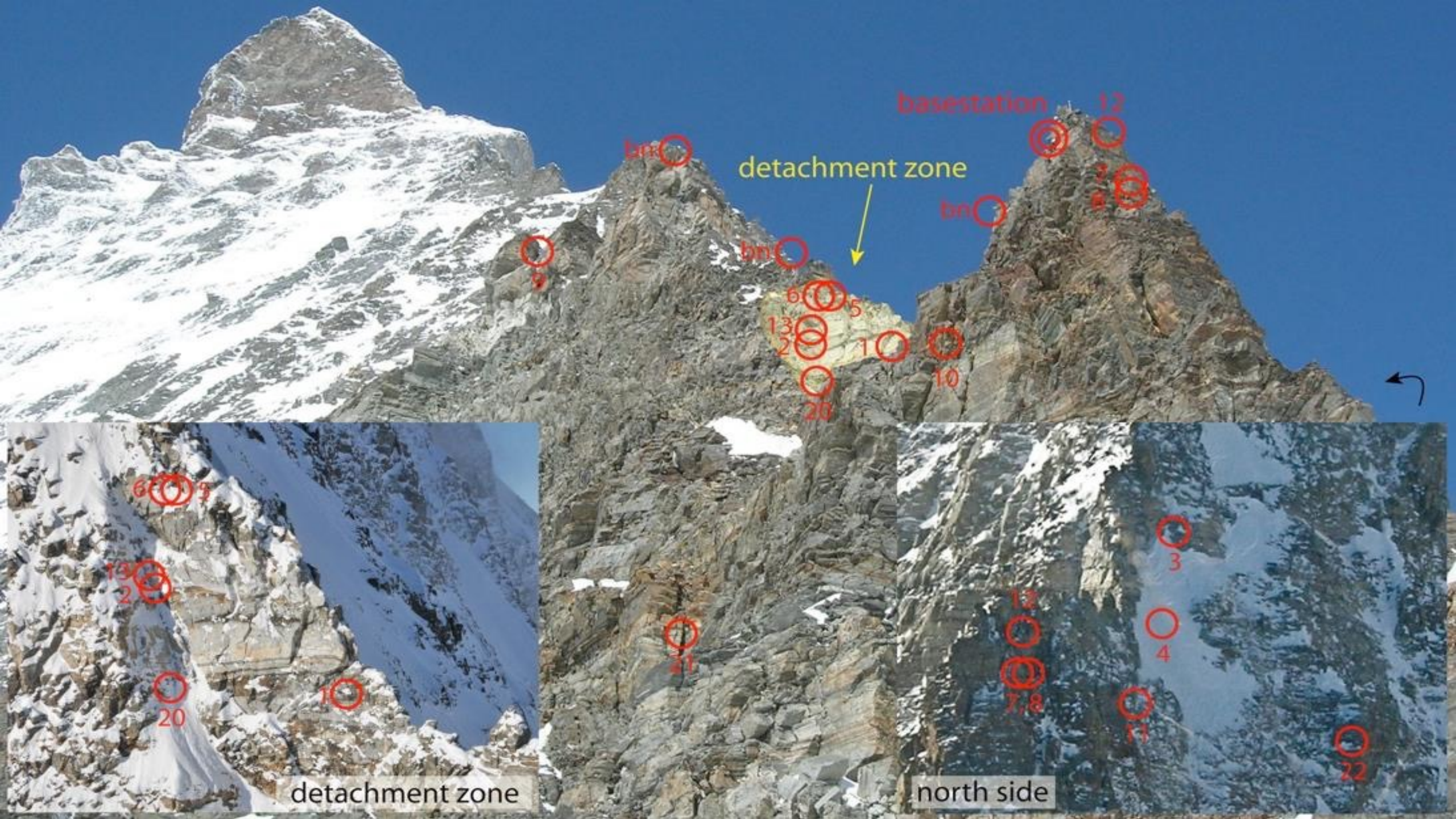
- Examples of interface functions:
  - uint8_t PCM_getPowerState (void)
  - bool PCM_gotoLPM0 (void)
  - bool PCM_gotoLPM3 (void)
  - bool PCM_gotoLPM4 (void)
  - bool PCM_shutdownDevice (uint32_t shutdownMode)



Battery-Operated Systems and Energy Harvesting

Embedded Systems in the Extreme - Permasense

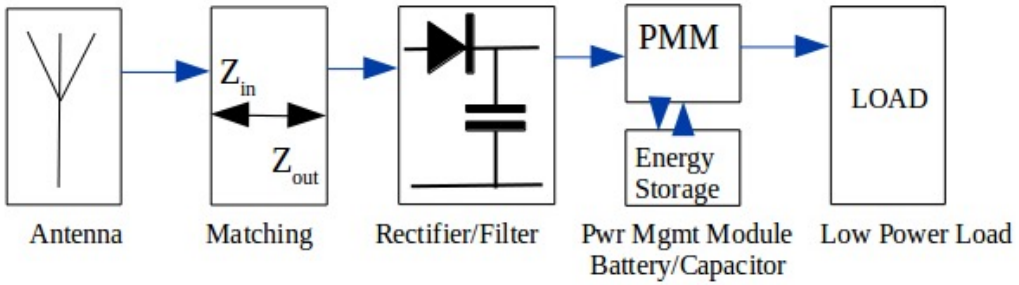



basestation
12
bn
detachment zone
bn
bn
9
6
5
13
2
1
10
20
21
3
4
12
7,8
11
22
detachment zone
north side

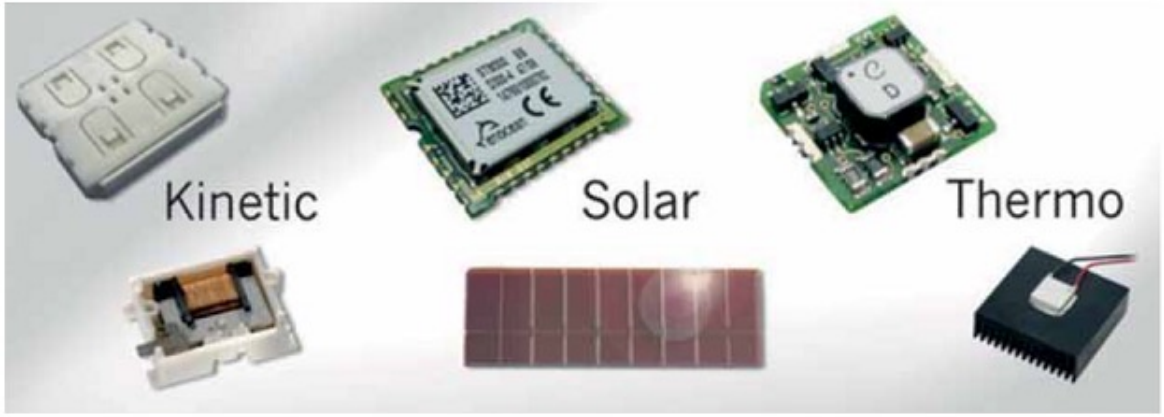
Reasons for Battery-Operated Devices and Harvesting

- Battery operation:
  - no continuous power source available
  - mobility

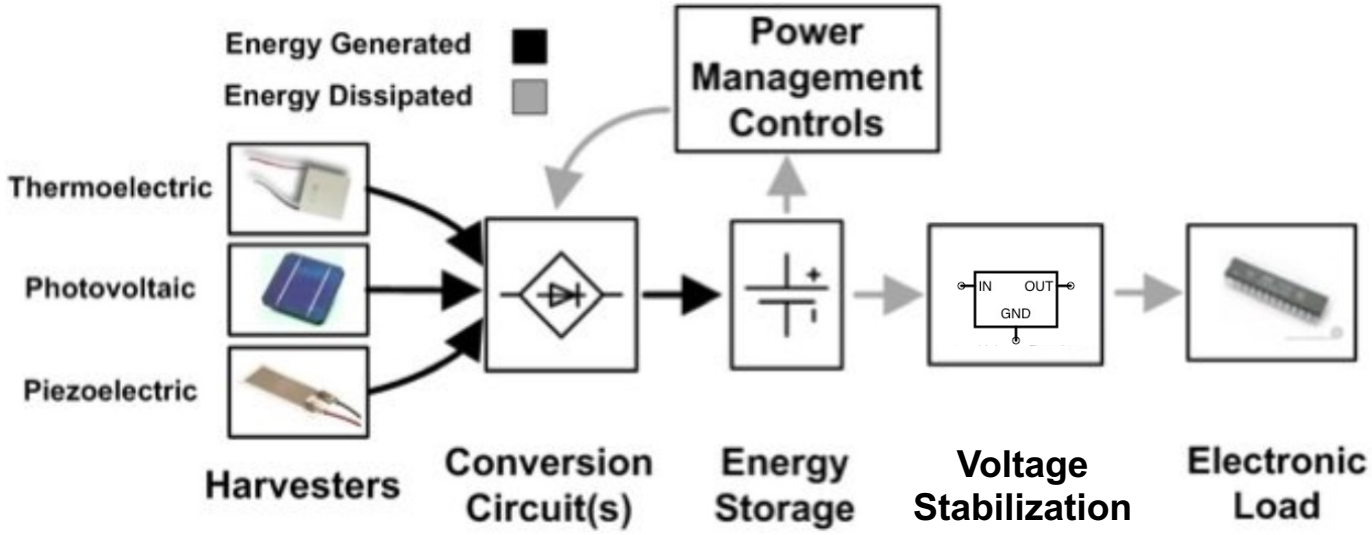
- Energy harvesting:
  - prolong lifetime of battery-operated devices
  - infinite lifetime using rechargeable batteries
  - autonomous operation



radio frequency (RF) harvesting



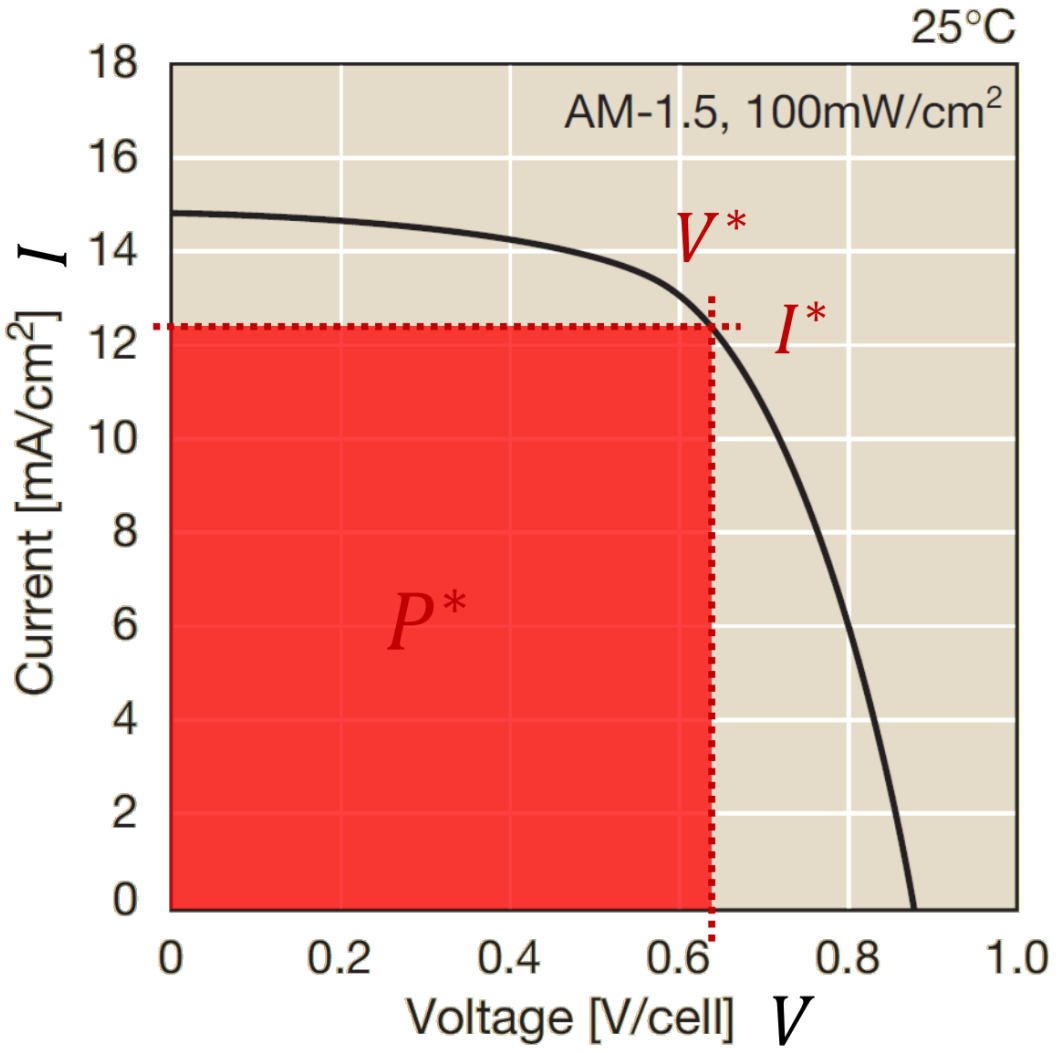
Typical Power Circuitry – Power Point Tracking



power point tracking / impedance matching; conversion to voltage of energy storage

rechargeable battery or supercapacitor

Solar Panel Characteristics

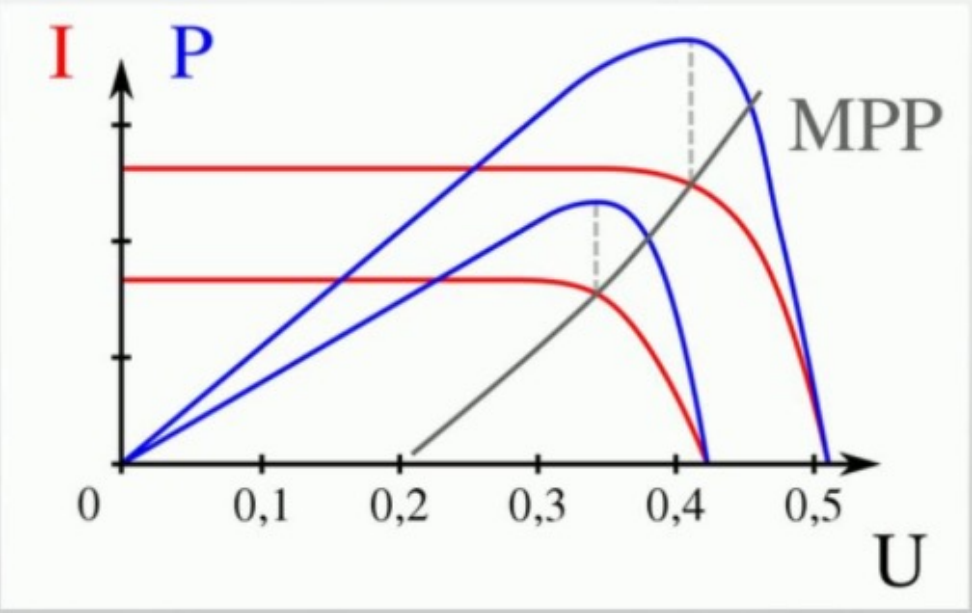


- Variable output power
  - Illuminance level
  - Electrical operation point
  - (Temperature, age, …)
- I-V-Characteristics
  - Non-linear
  - Dependent on ambient
- Maximum Power Point Tracking
  - Dynamic algorithm to find P^*

Diagram: *Amorton Amorphous Silicon Solar Cells Datasheet,* © Panasonic

Typical Power Circuitry – Maximum Power Point Tracking

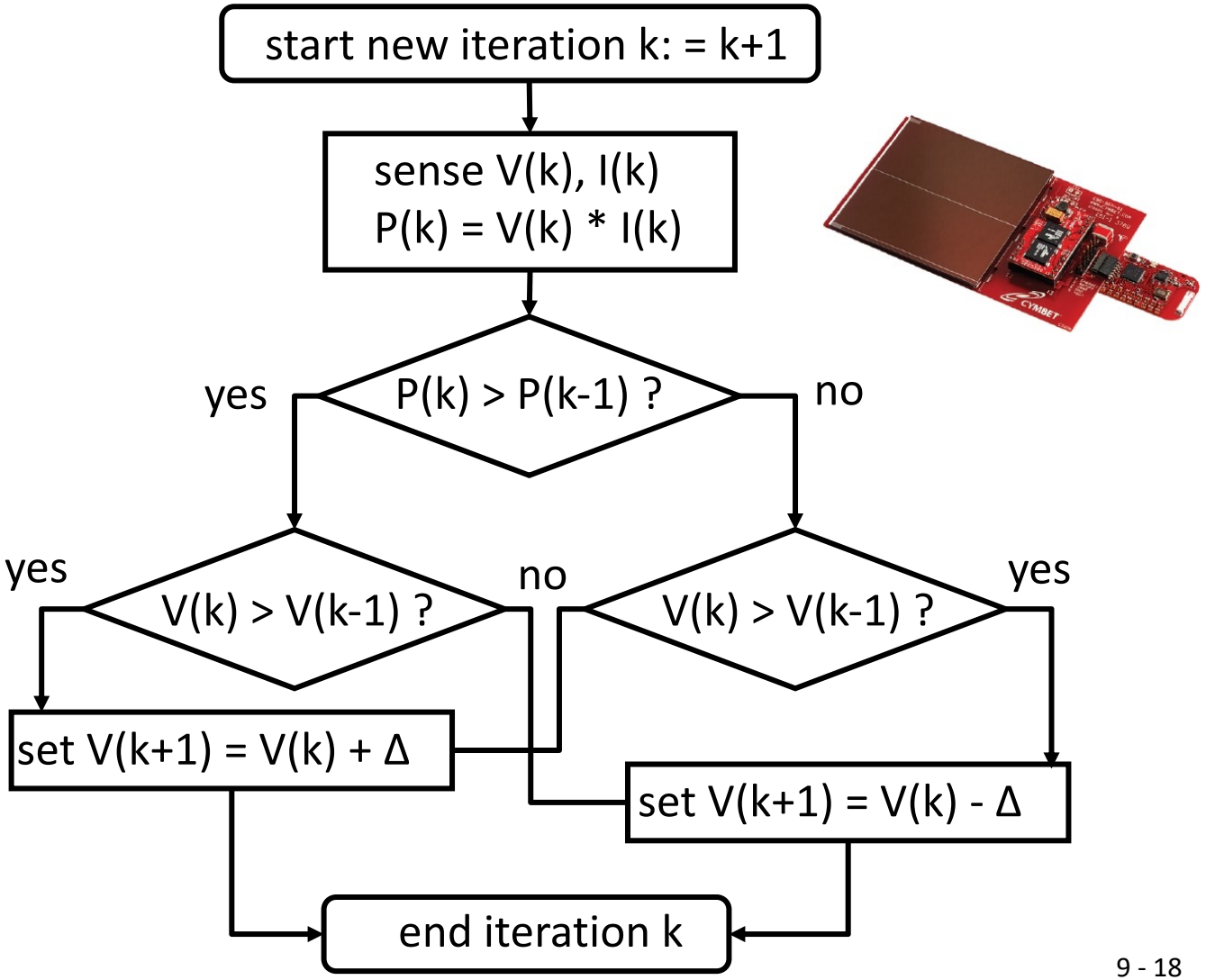
U/I curves of a typical solar cell:



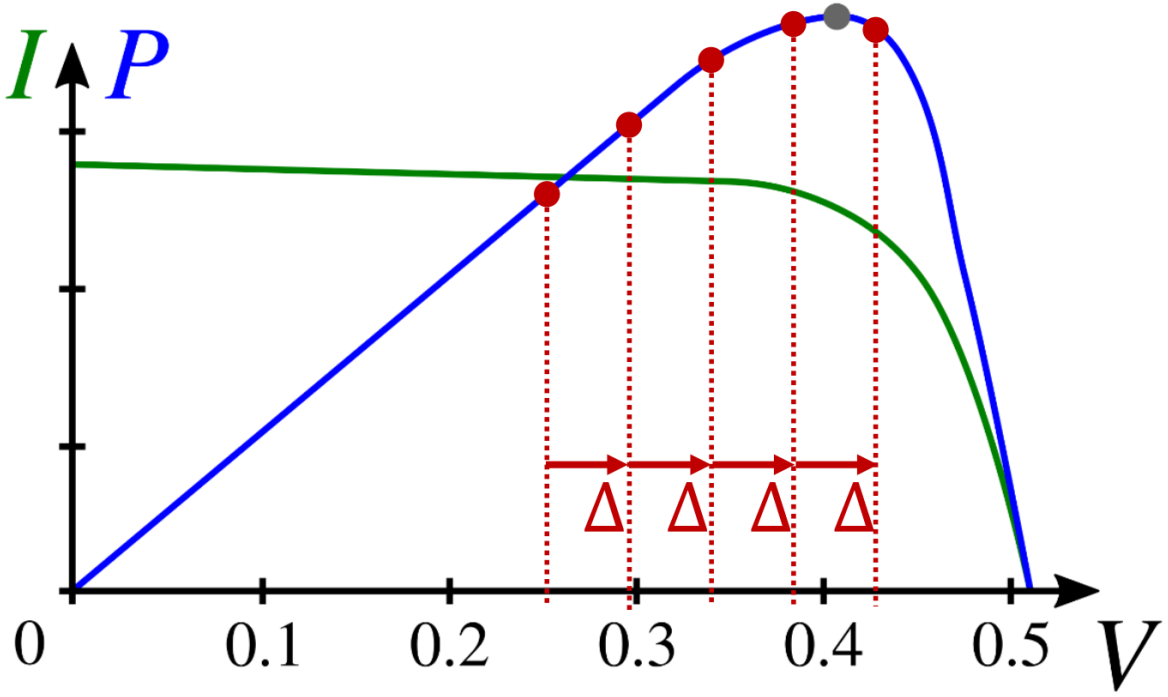
red: current for different light intensities
blue: power for different light intensities
grey: maximal power

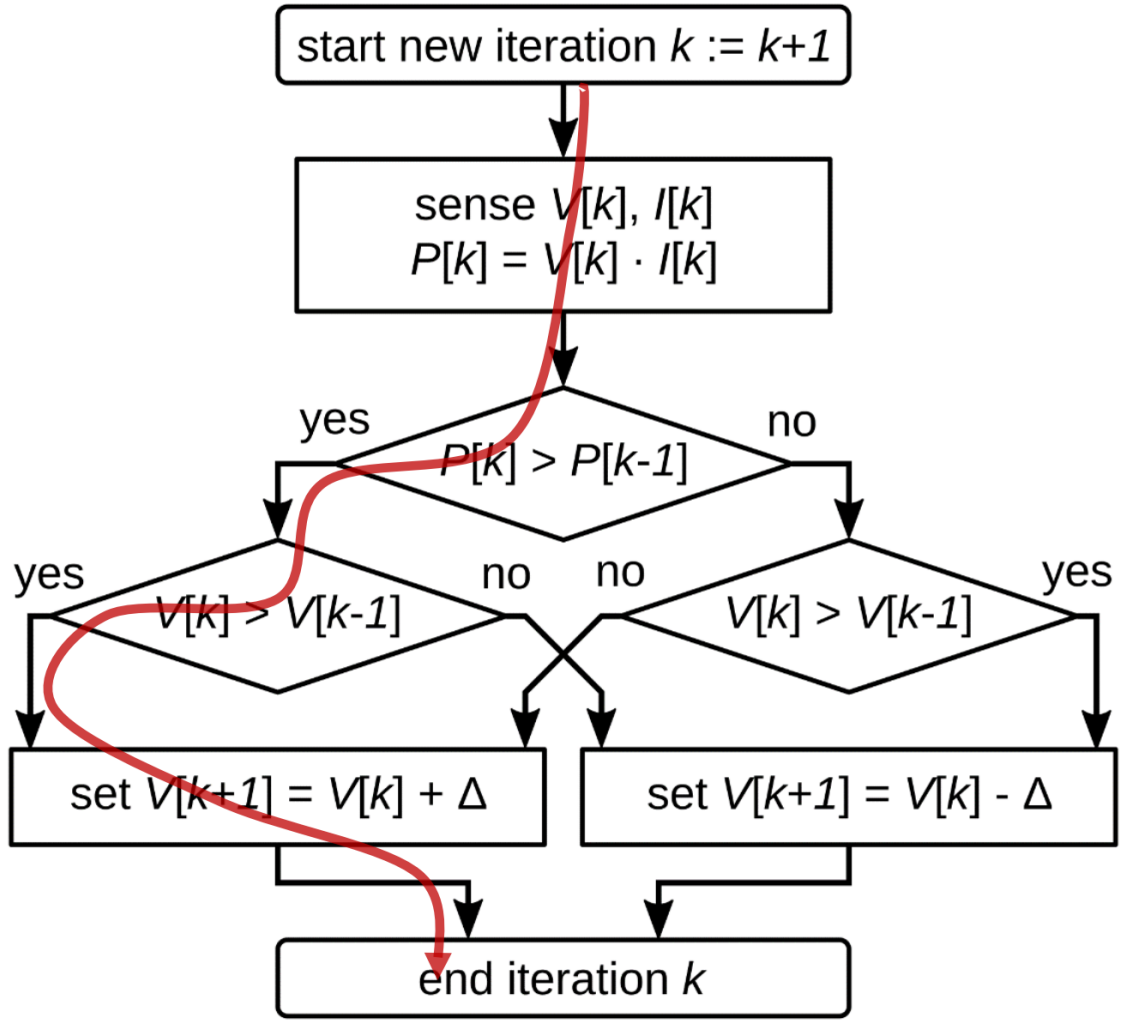
tracking: determine optimal impedance seen by the solar panel

simple tracking algorithm (assume constant illumination) :

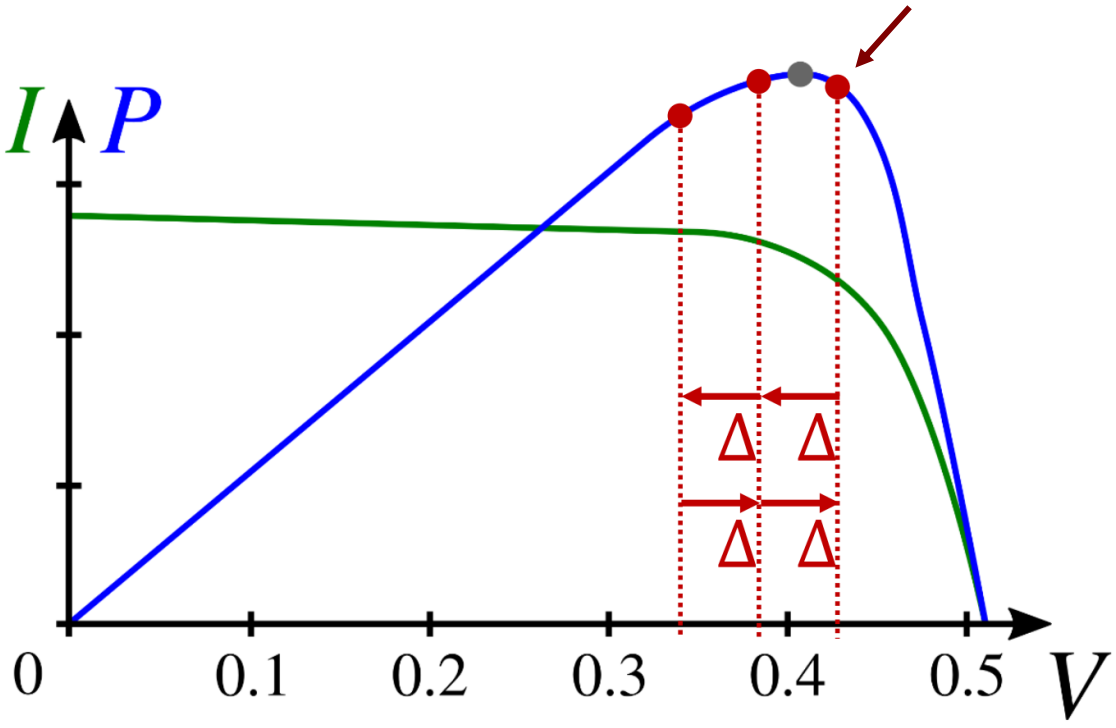


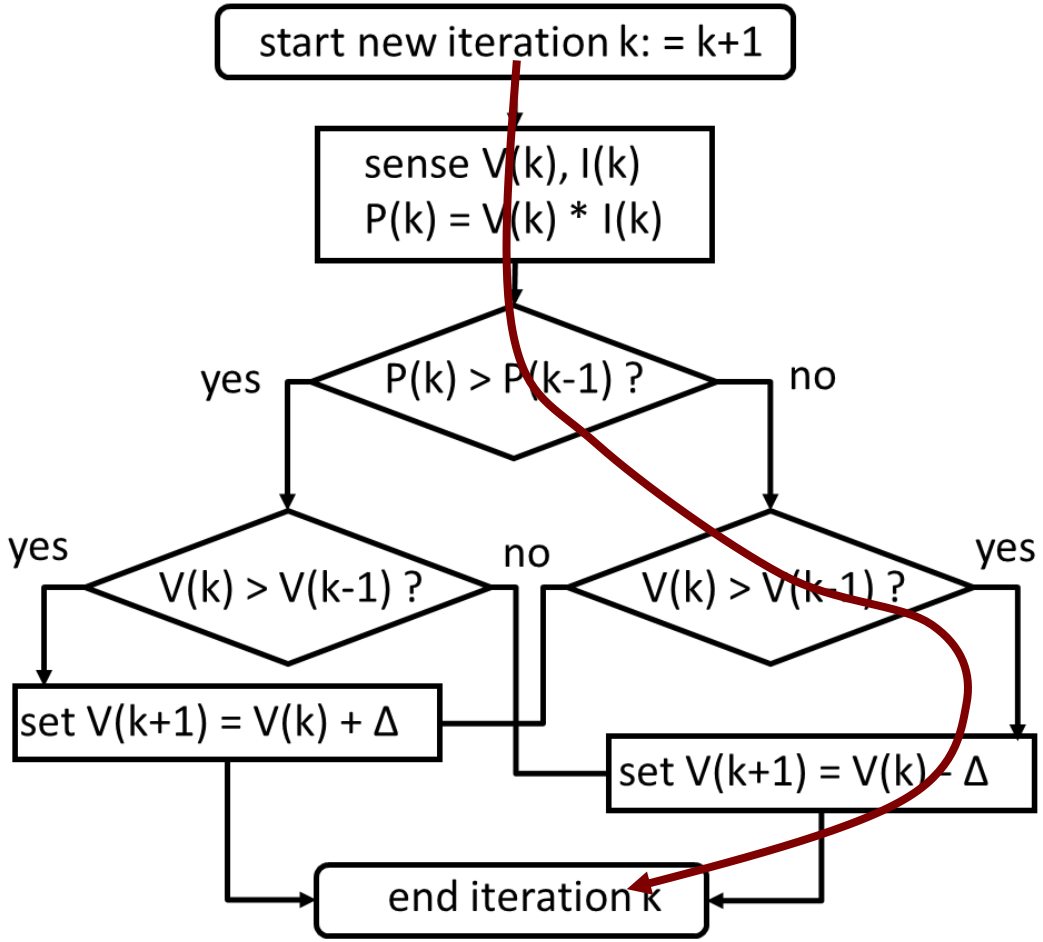
Maximal Power Point Tracking



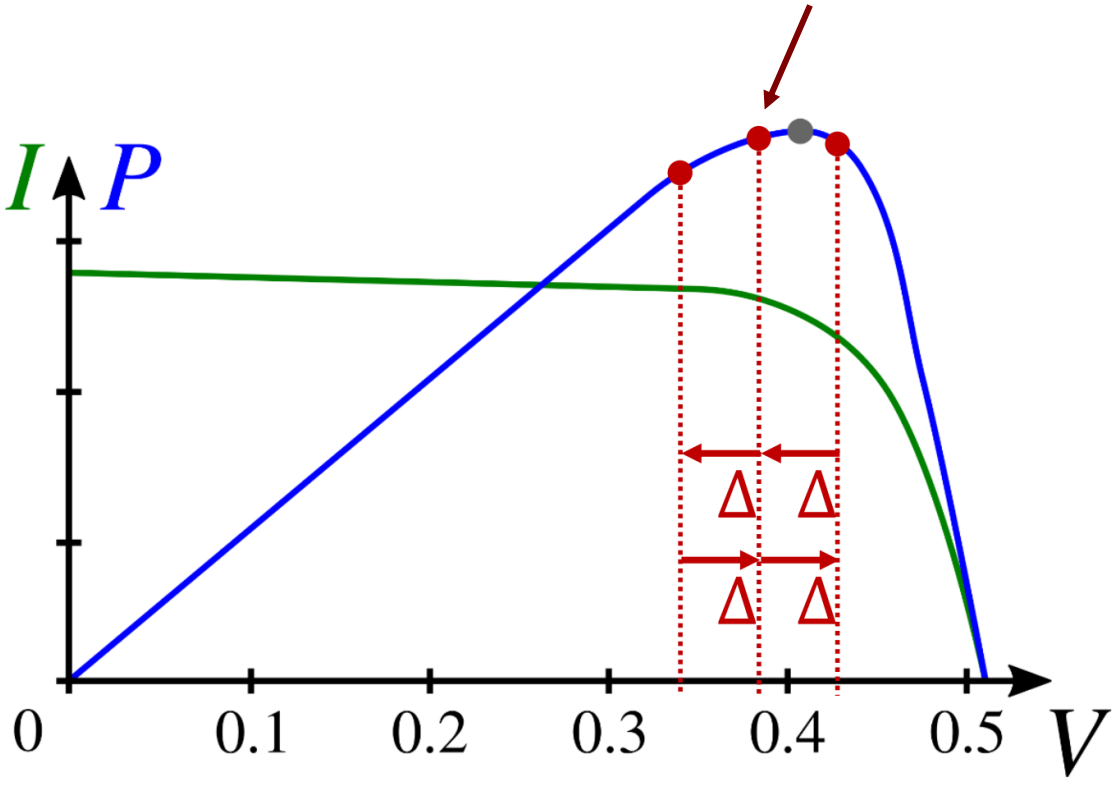


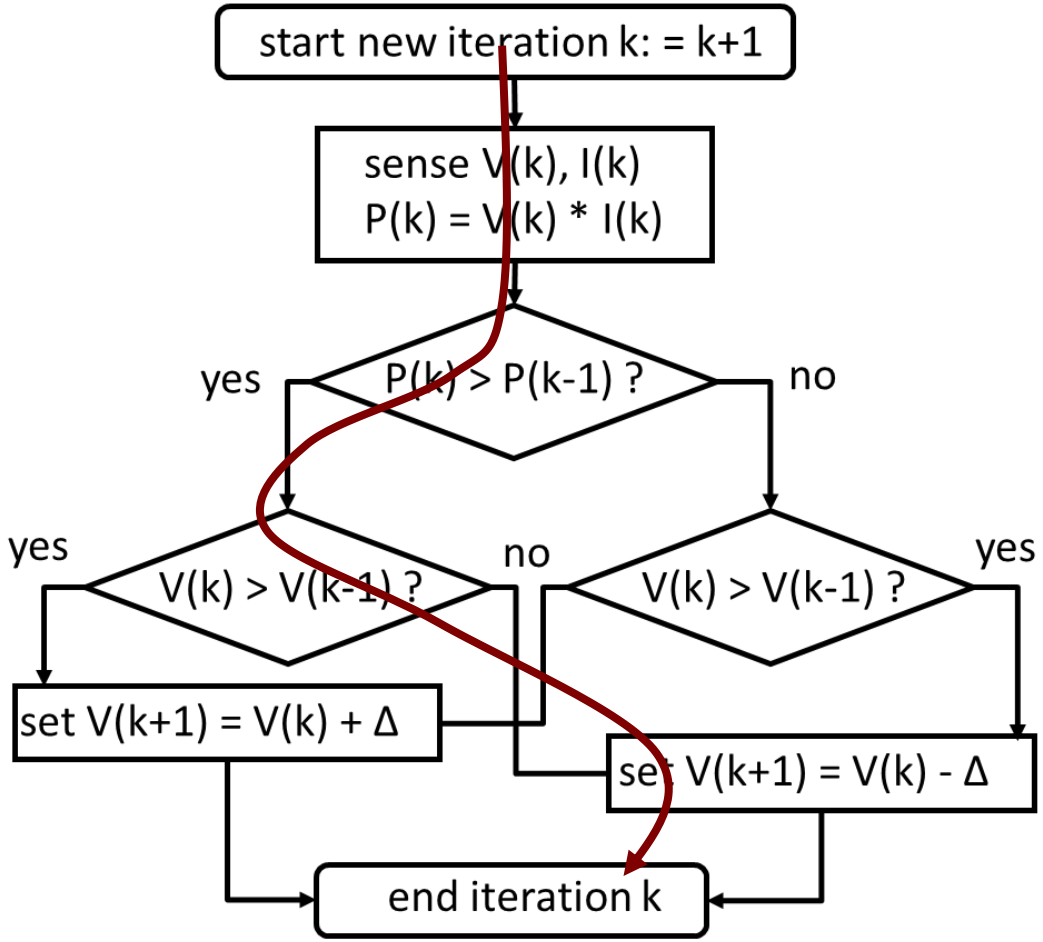
Maximal Power Point Tracking



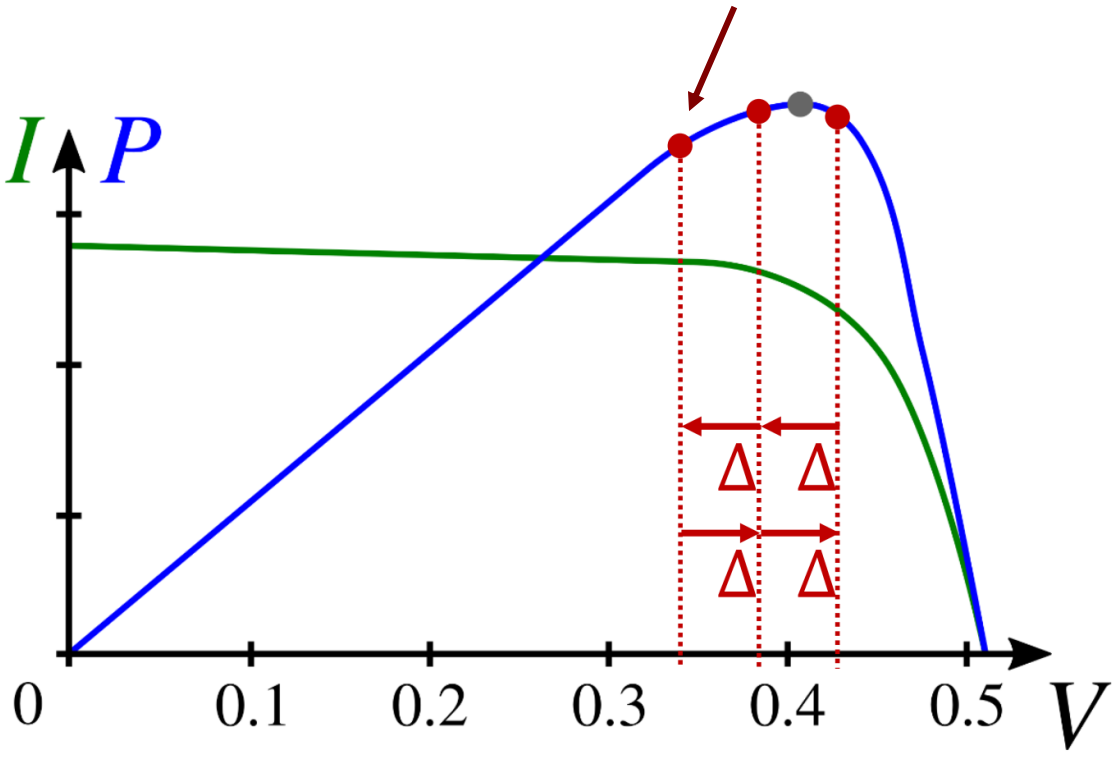


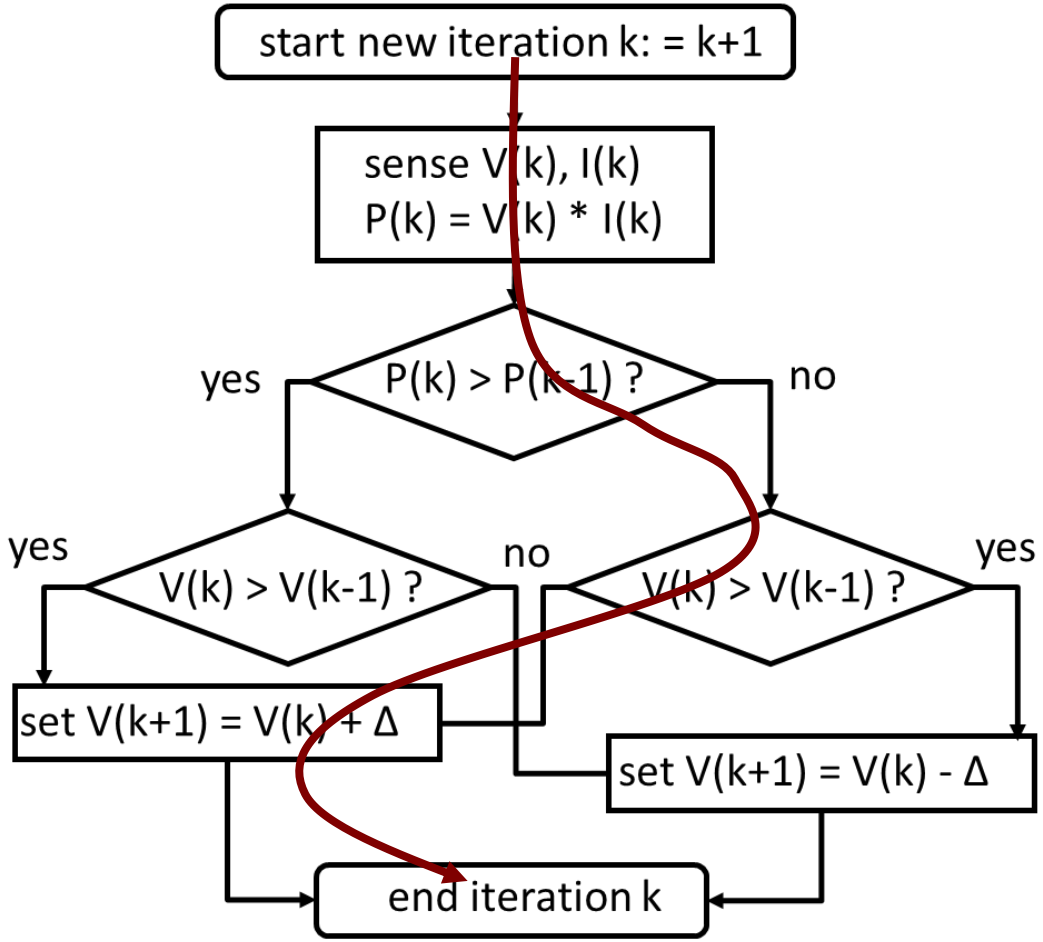
Maximal Power Point Tracking



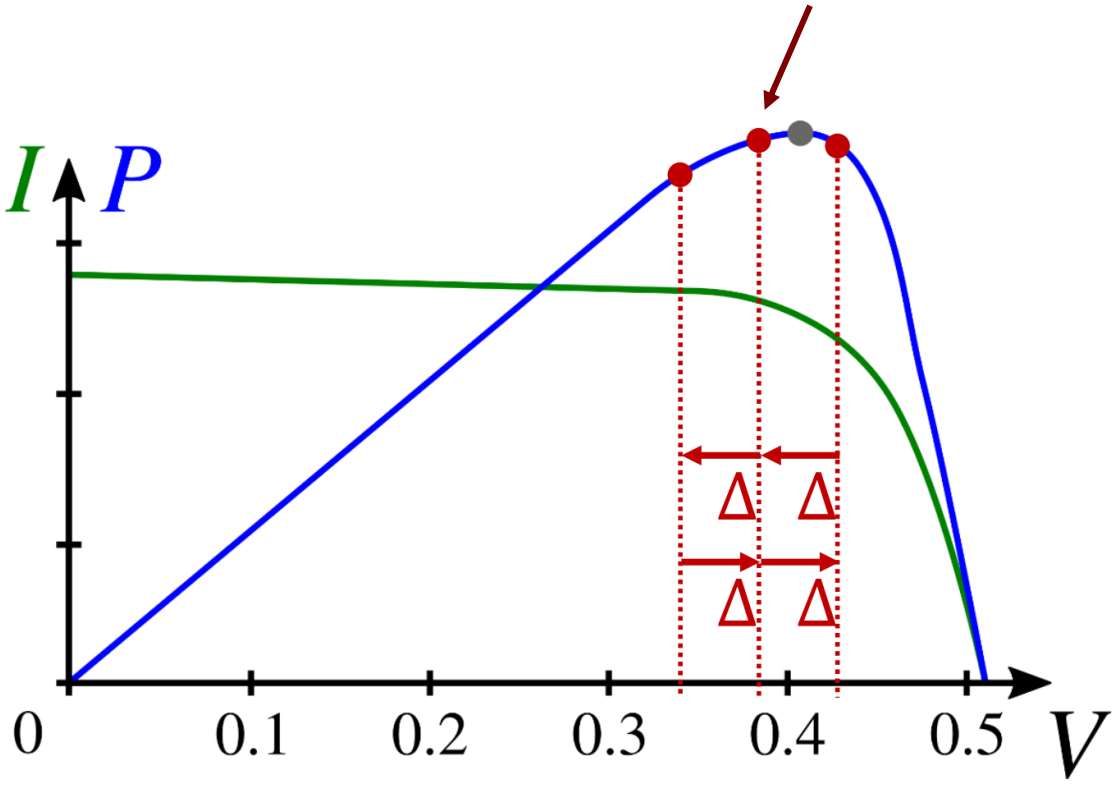


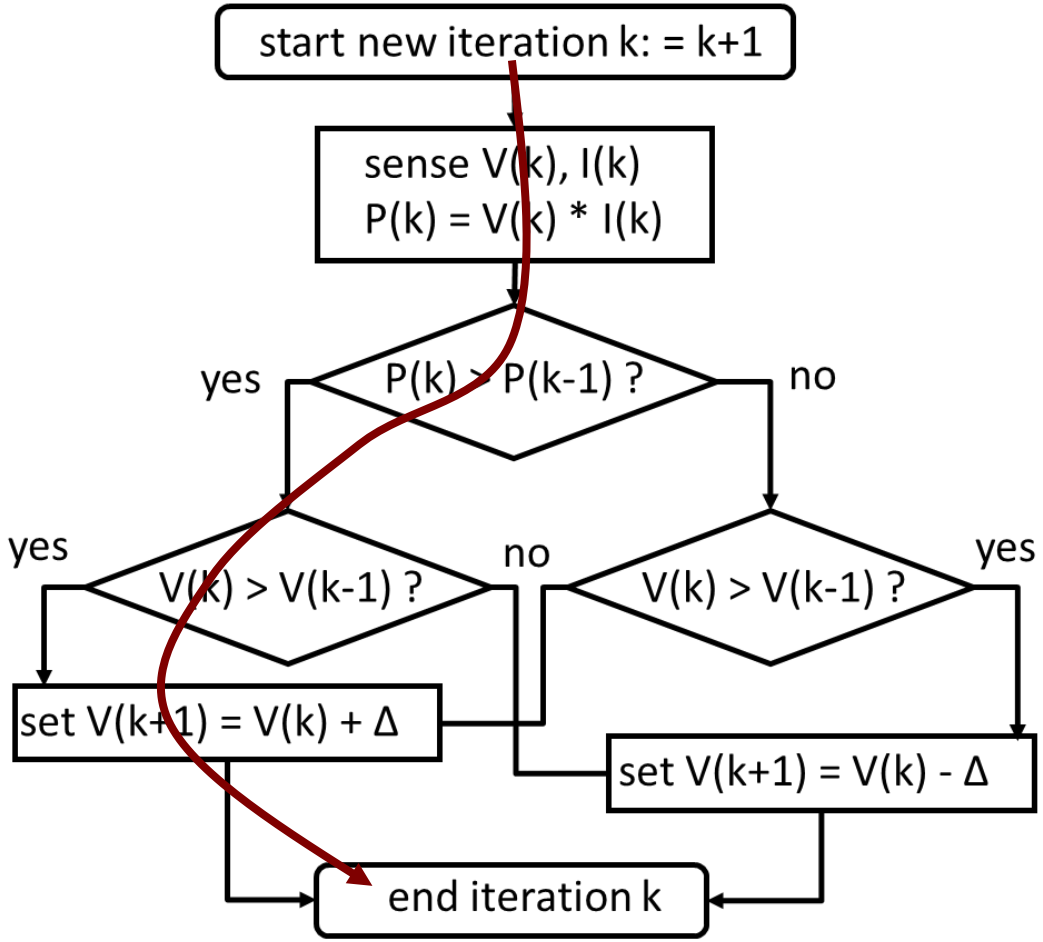
Maximal Power Point Tracking





Maximal Power Point Tracking



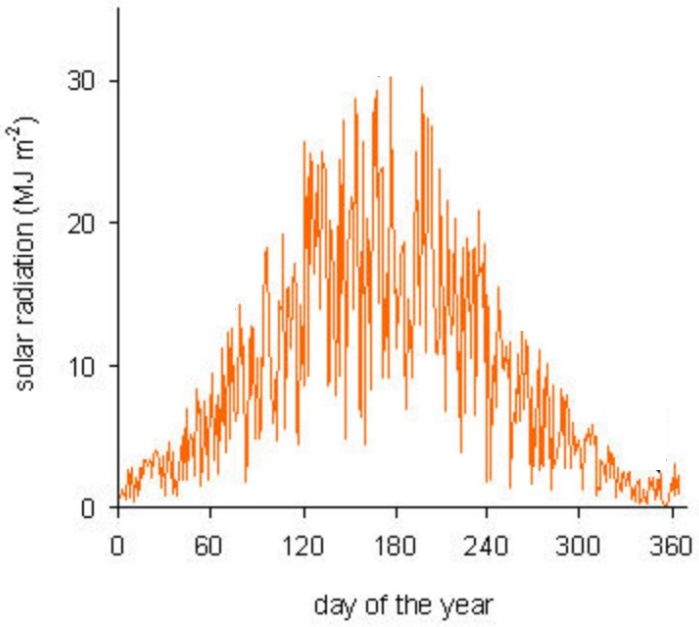


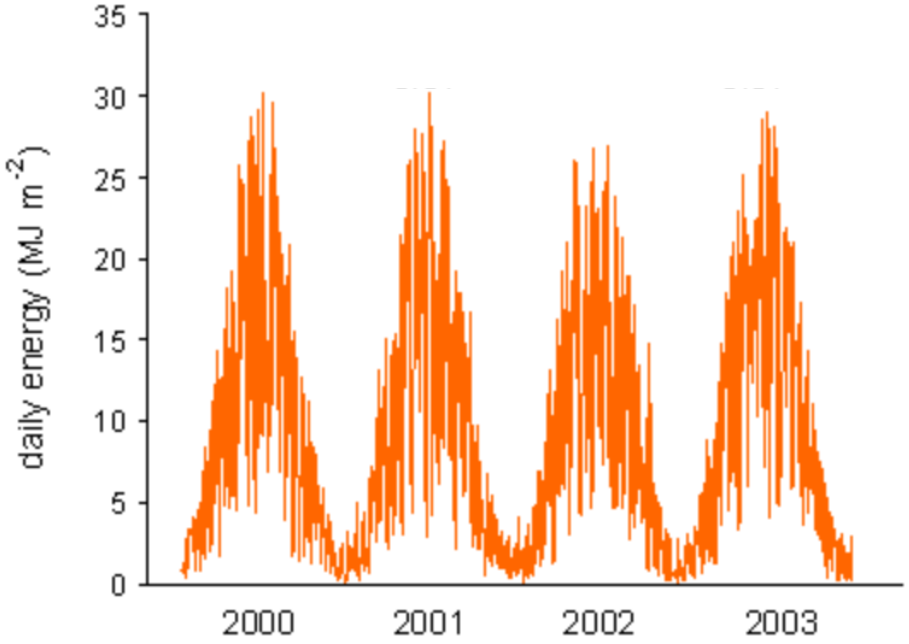
Typical Challenge in (Solar) Harvesting Systems

Challenges:

- What is the optimal maximum capacity of the battery?
- What is the optimal area of the solar cell?
- How can we control the application such that a continuous system operation is possible, even under a varying input energy (summer, winter, clouds)?

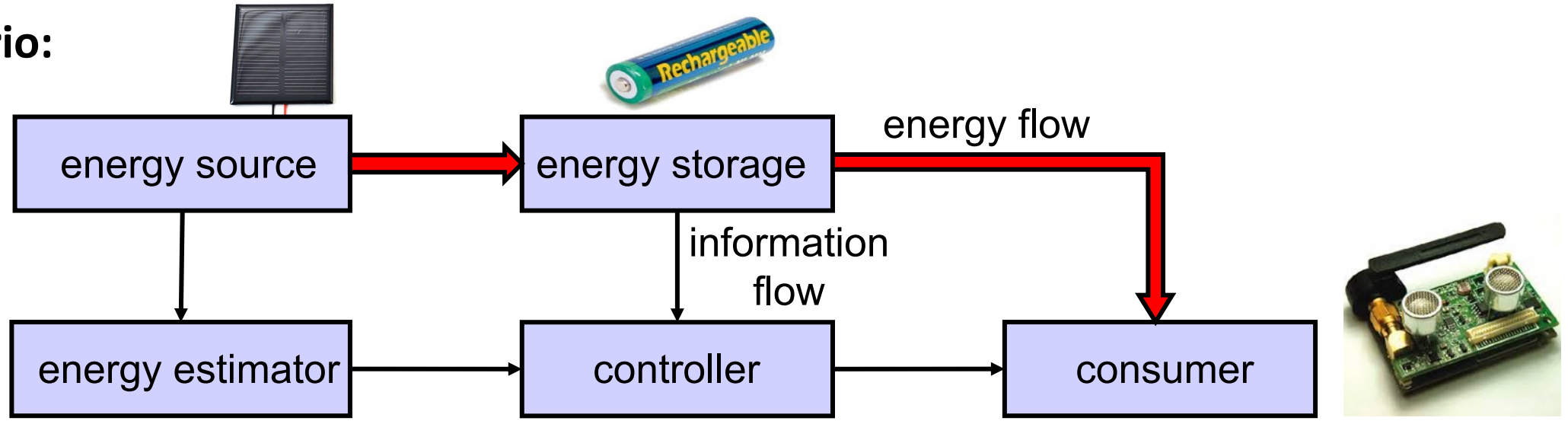
Example of a solar energy trace:





Example: Application Control

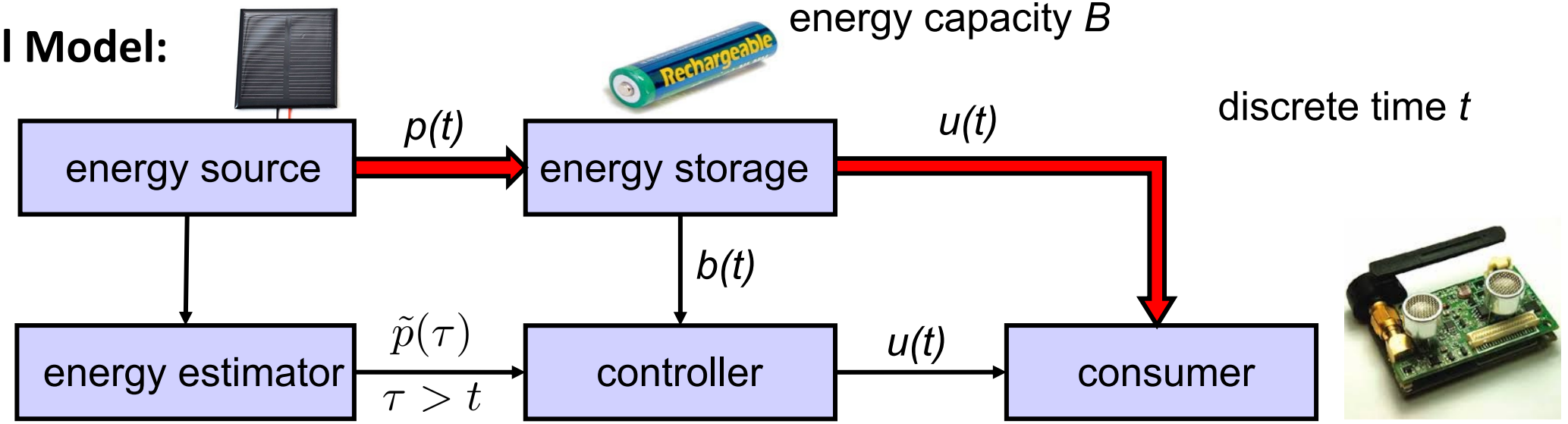
Scenario:



- The controller can adapt the service of the consumer device, for example the sampling rate for its sensors or the transmission rate of information. As a result, the power consumption changes proportionally.
- **Precondition for correctness** of application control: Never run out of energy.
- **Example for optimality criterion**: Maximize the lowest service of (or equivalently, the lowest energy flow to) the consumer.

Application Control

Formal Model:



- harvested and used energy in *[t, t+1)*: *p(t), u(t)*
- battery model: $b(t+1) = \min\{b(t) + p(t) - u(t), B\}$
- failure state: $b(t) + p(t) - u(t) < 0$
- utility:

$$U(t_1, t_2) = \sum_{t_1 \leq \tau < t_2} \mu(u(\tau))$$

μ is a strictly concave function; higher used energy gives a reduced reward for the overall utility.

Application Control

- **What do we want?** We would like to determine an optimal control *u*(t)* for time interval *[t, t+1)* for all *t in [0, T)* with the following properties:
  - $\forall\, 0 \leq t < T \;:\; b^*(t) + p(t) - u^*(t) \geq 0$
  - There is no feasible use function u(t) with a larger minimal energy:

    $$\forall u \;:\; \min_{0 \leq t < T} \{u(t)\} \leq \min_{0 \leq t < T} \{u^*(t)\}$$

  - The use function maximizes the utility *U(0, T).*
  - We suppose that the battery has the same or better state at the end than at the start of the time interval, i.e., *b*(T) ≥ b*(0).*
- We would like to answer two questions:
  - Can we say something about the characteristics of *u*(t)* ?
  - How does an algorithm look like that efficiently computes *u*(t)* ?

Application Control

Theorem: Given a use function u*(t), $t \in [0, T)$ such that the system never enters a failure state. If u*(t) is optimal with respect to maximizing the minimal used energy among all use functions and maximizes the utility U(t, T), then the following relations hold for all $\tau \in (0, T)$:

$$u^*(\tau - 1) < u^*(\tau) \Longrightarrow b^*(\tau) = 0$$

empty battery

$$u^*(\tau - 1) > u^*(\tau) \Longrightarrow b^*(\tau) = B$$

full battery

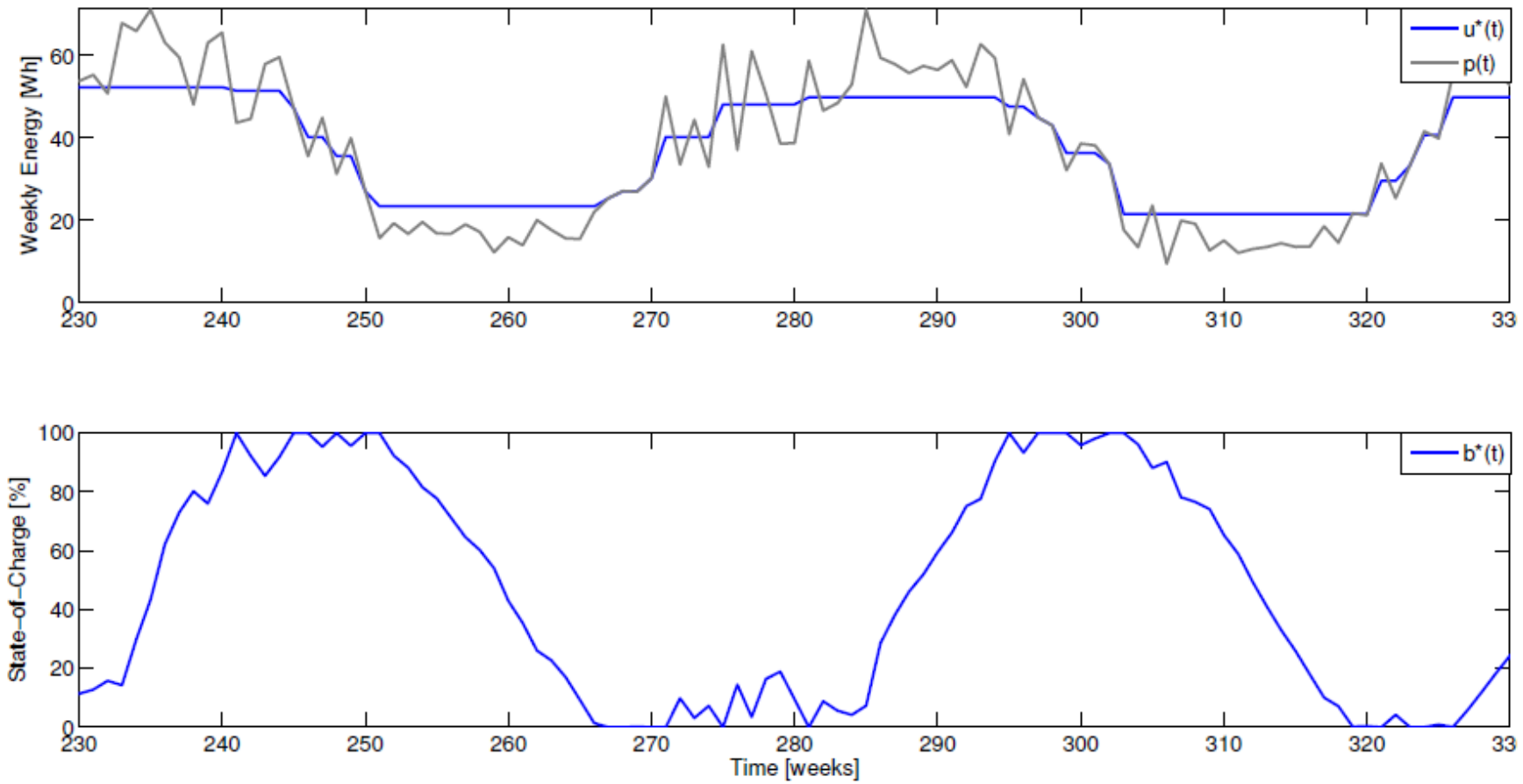
Sketch of a proof: First, let us show that a consequence of the above theorem is true (just reverting the relations):

$$\forall \tau \in (s, t] \ : \ 0 < b^*(\tau) < B \implies \forall \tau \in [s, t] \ : \ u^*(\tau) = u^*(t)$$

In other words, as long as the battery is neither full nor empty, the optimal use function does not change.

Application Control

- Proof sketch cont.:

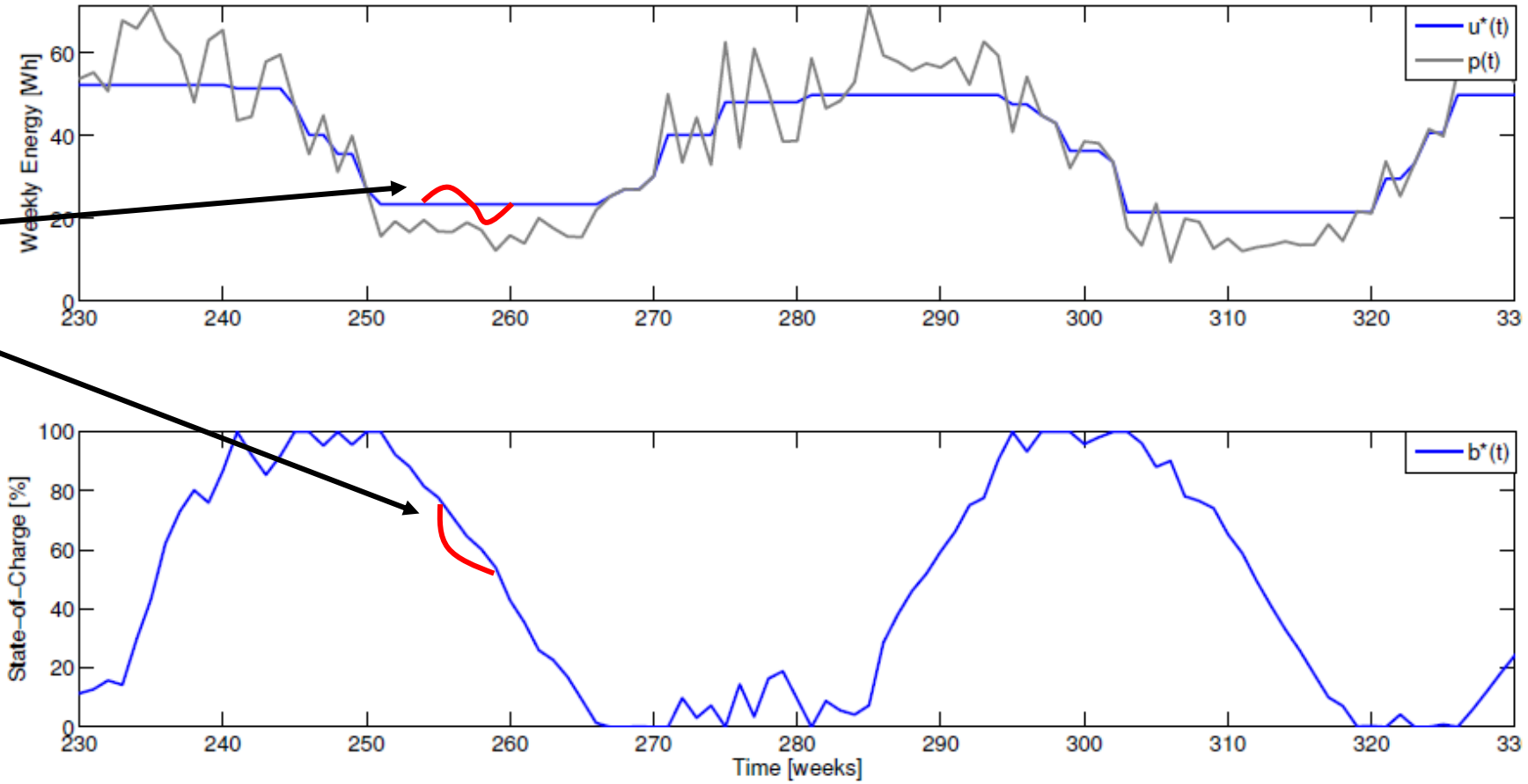


(top) Example of an optimal use function $u^*(t)$ for a given harvest function $p(t)$ and (bottom) the corresponding stored energy $b^*(t)$.

Application Control

- Proof sketch cont.:

suppose we change the use function locally from being constant such that the overall battery state does not change



(top) Example of an optimal use function $u^*(t)$ for a given harvest function $p(t)$ and (bottom) the corresponding stored energy $b^*(t)$.

then the utility is worse due to the concave function μ: diminishing reward for higher use function values; and the minimal use function is potentially smaller

Application Control

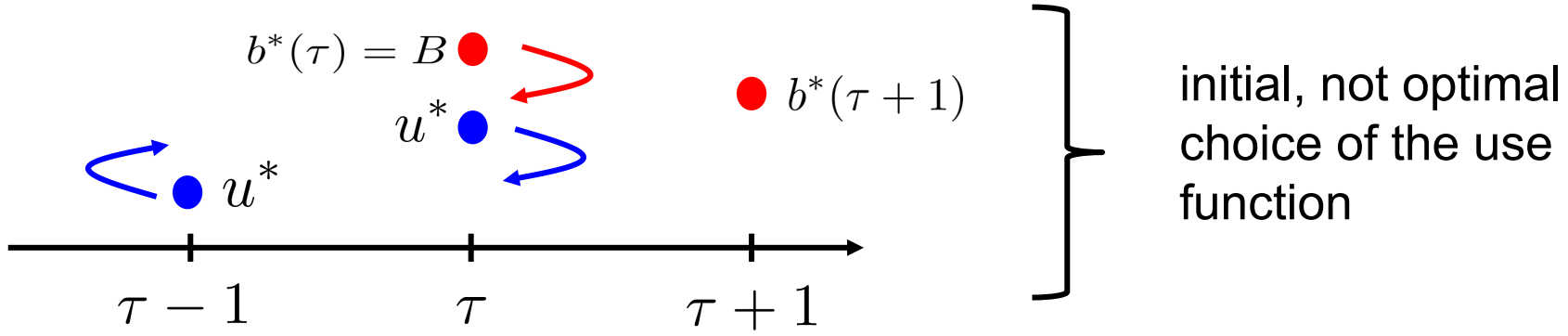
- Proof sketch cont.: Now we show that for all $\tau \in (t, T)$

$$u^*(\tau - 1) < u^*(\tau) \Longrightarrow b^*(\tau) = 0$$

or equivalently

$$b^*(\tau) > 0 \Longrightarrow u^*(\tau - 1) \geq u^*(\tau)$$

We already have shown this for $0 < b^*(\tau) < B$. Therefore, we only need to show that $b^*(\tau) = B \Longrightarrow u^*(\tau - 1) \geq u^*(\tau)$. Suppose now that we have $u^*(\tau - 1) < u^*(\tau)$ if the battery is full at τ. Then we can increase the use at time $\tau - 1$ and decrease it at time τ by the same amount without changing the battery level at time $\tau + 1$. This again would increase the overall utility and potentially increase the minimal use function.



Application Control

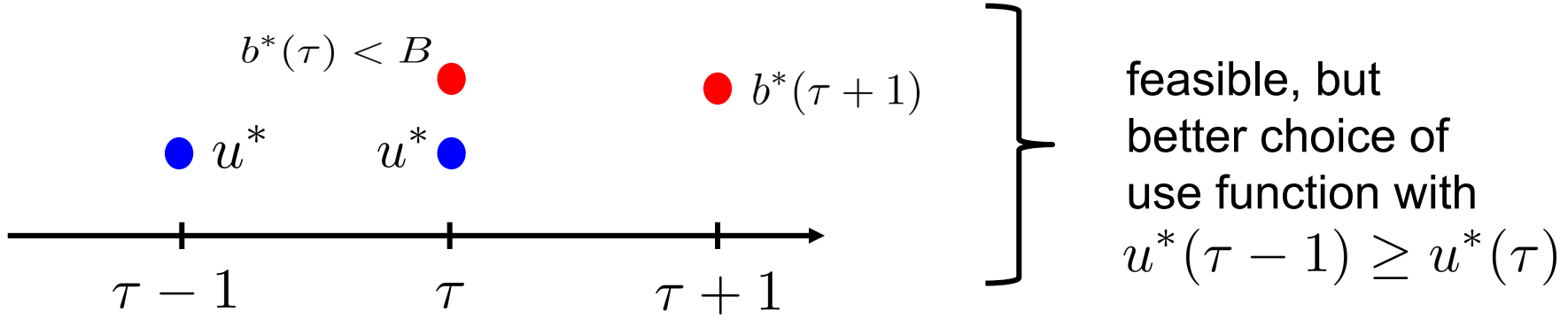
- Proof sketch cont.: Now we show that for all $\tau \in (t, T)$

$$u^*(\tau - 1) < u^*(\tau) \Longrightarrow b^*(\tau) = 0$$

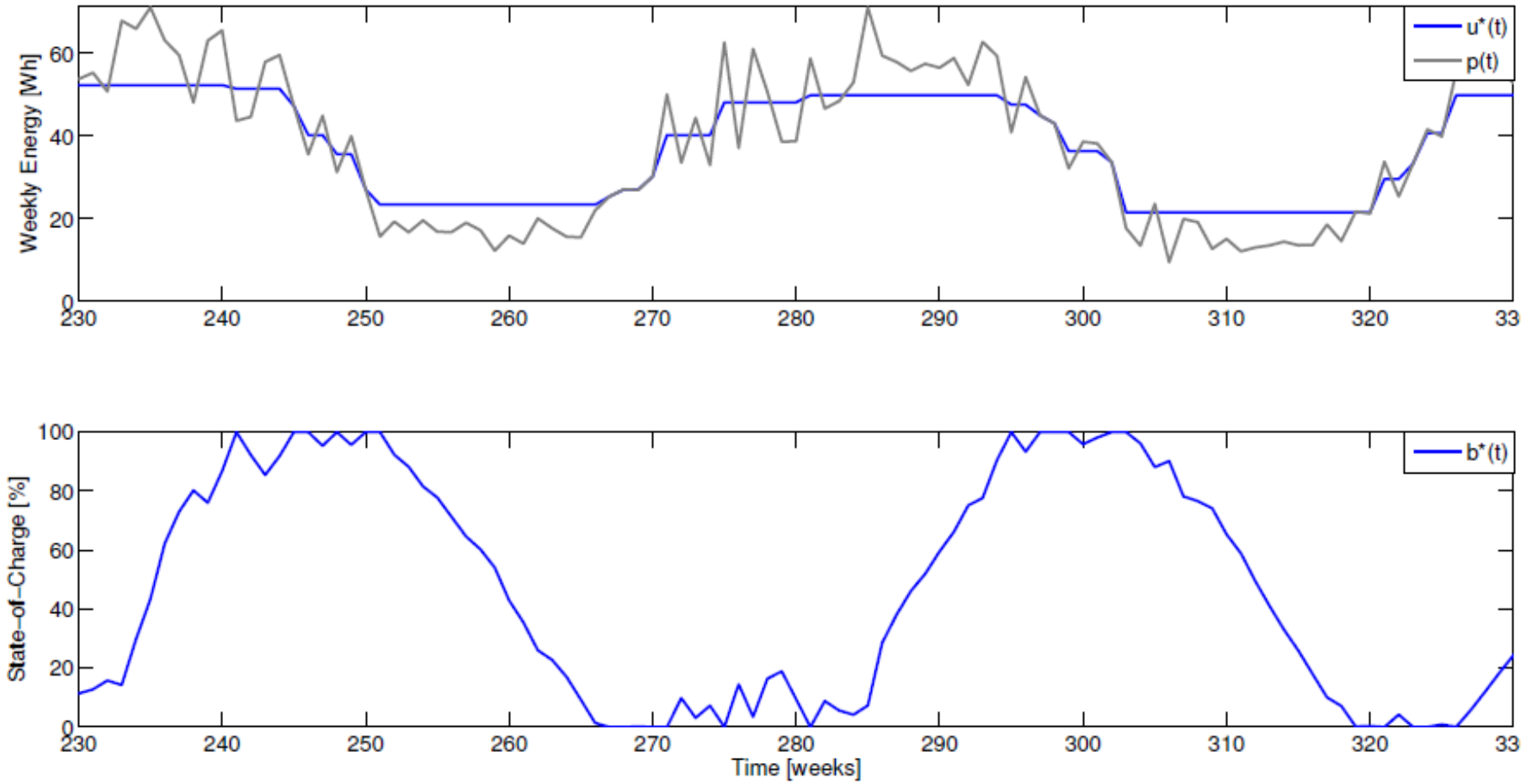
or equivalently

$$b^*(\tau) > 0 \Longrightarrow u^*(\tau - 1) \geq u^*(\tau)$$

We already have shown this for $0 < b^*(\tau) < B$. Therefore, we only need to show that $b^*(\tau) = B \Longrightarrow u^*(\tau - 1) \geq u^*(\tau)$. Suppose now that we have $u^*(\tau - 1) < u^*(\tau)$ if the battery is full at τ. Then we can increase the use at time $\tau - 1$ and decrease it at time τ by the same amount without changing the battery level at time $\tau + 1$. This again would increase the overall utility and potentially increase the minimal use function.



Application Control



(top) Example of an optimal use function $u^*(t)$ for a given harvest function $p(t)$ and (bottom) the corresponding stored energy $b^*(t)$.

Application Control

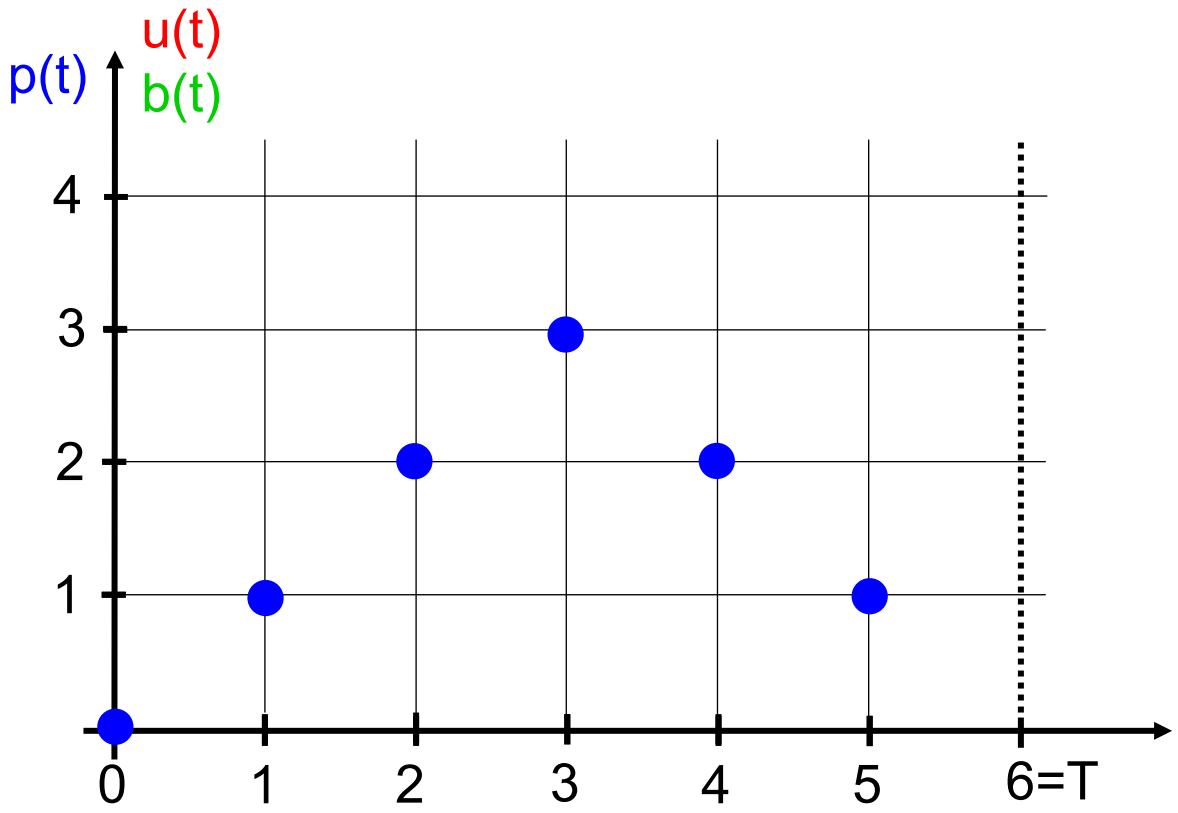
- How can we efficiently compute an optimal use function?
  - There are several options available as we just need to solve a convex optimization problem.
  - A simple but inefficient possibility is to convert the problem into a linear program. At first suppose that the utility is simply

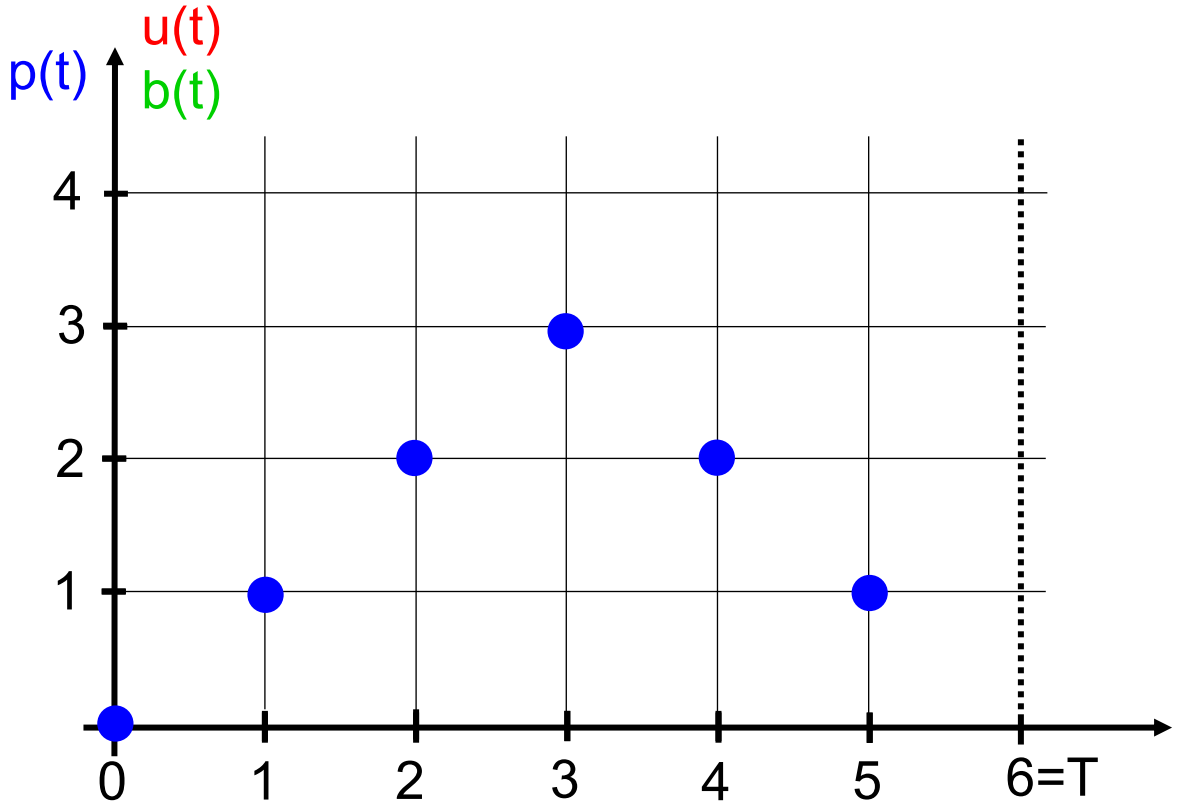
$$U(0,T) = \sum_{0 \le \tau < T} u(\tau)$$

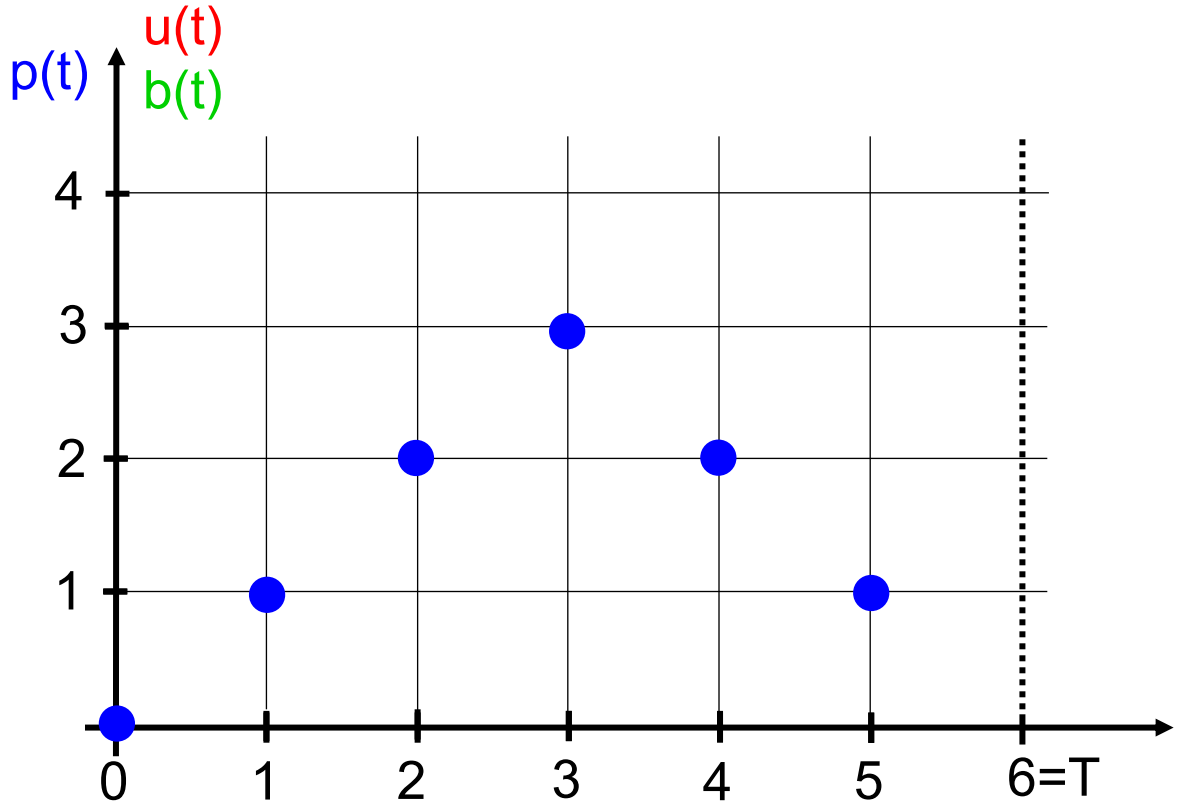
Then the linear program has the form:

$$\begin{aligned}
&\text{maximize} \quad \sum_{0 \le \tau < T} u(\tau) \\
&\forall \tau \in [0,T) \ : \ b(\tau+1) = b(\tau) - u(\tau) + p(\tau) \\
&\forall \tau \in [0,T) \ : \ 0 \le b(\tau+1) \le B \\
&\forall \tau \in [0,T) \ : \ u(\tau) \ge 0 \\
&b(T) = b(0) = b_0
\end{aligned}$$

[Concave functions μ could be piecewise linearly approximated. This is not shown here.]


u(t)
p(t)
b(t)
4
3
2
1
0
1
2
3
4
5
6=T


u(t)
b(t)
p(t)
4
3
2
1
0
1
2
3
4
5
6=T

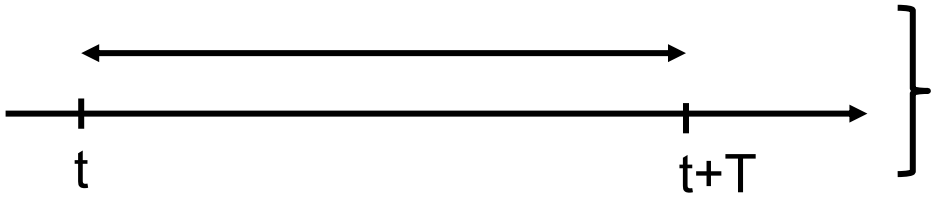

u(t)
b(t)
p(t)
4
3
2
1
0
1
2
3
4
5
6=T

Application Control

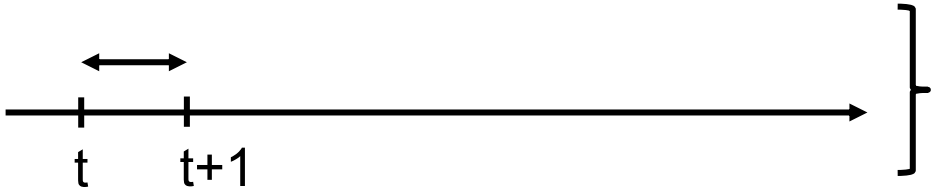
- But what happens if the estimation of the future incoming energy is not correct?
  - If it would be correct, then we would just compute the whole future application control now and would not change anything anymore.
  - This will not work as errors will accumulate and we will end up with many infeasible situations, i.e., the battery is completely empty and we are forced to stop the application.
  - **Possibility**: Finite horizon control
    - At time t, we compute the optimal control (see previous slides) using the currently available battery state b(t) with predictions $\tilde{p}(\tau)$ for all $t \leq \tau < t+T$ and $b(t+T) = b(t)$.
    - From the computed optimal use function $u(\tau)$ for all $t \leq \tau < t+T$ we just take the first use value *u(t)* in order to control the application.
    - At the next time step, we take as initial battery state the actual state; therefore, we take mispredictions into account. For the estimated future energy, we also take the new estimations.

Application Control

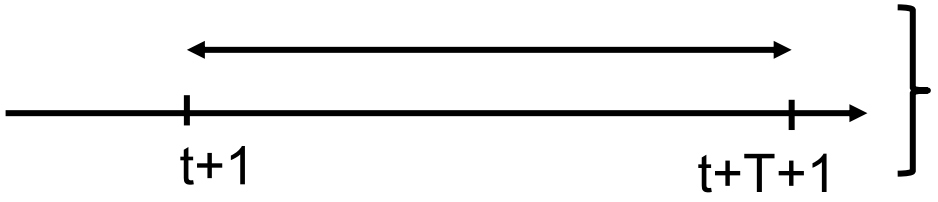
- **Finite horizon control**:



compute the optimal use function in [t, t+T)
using the actual battery state at time t

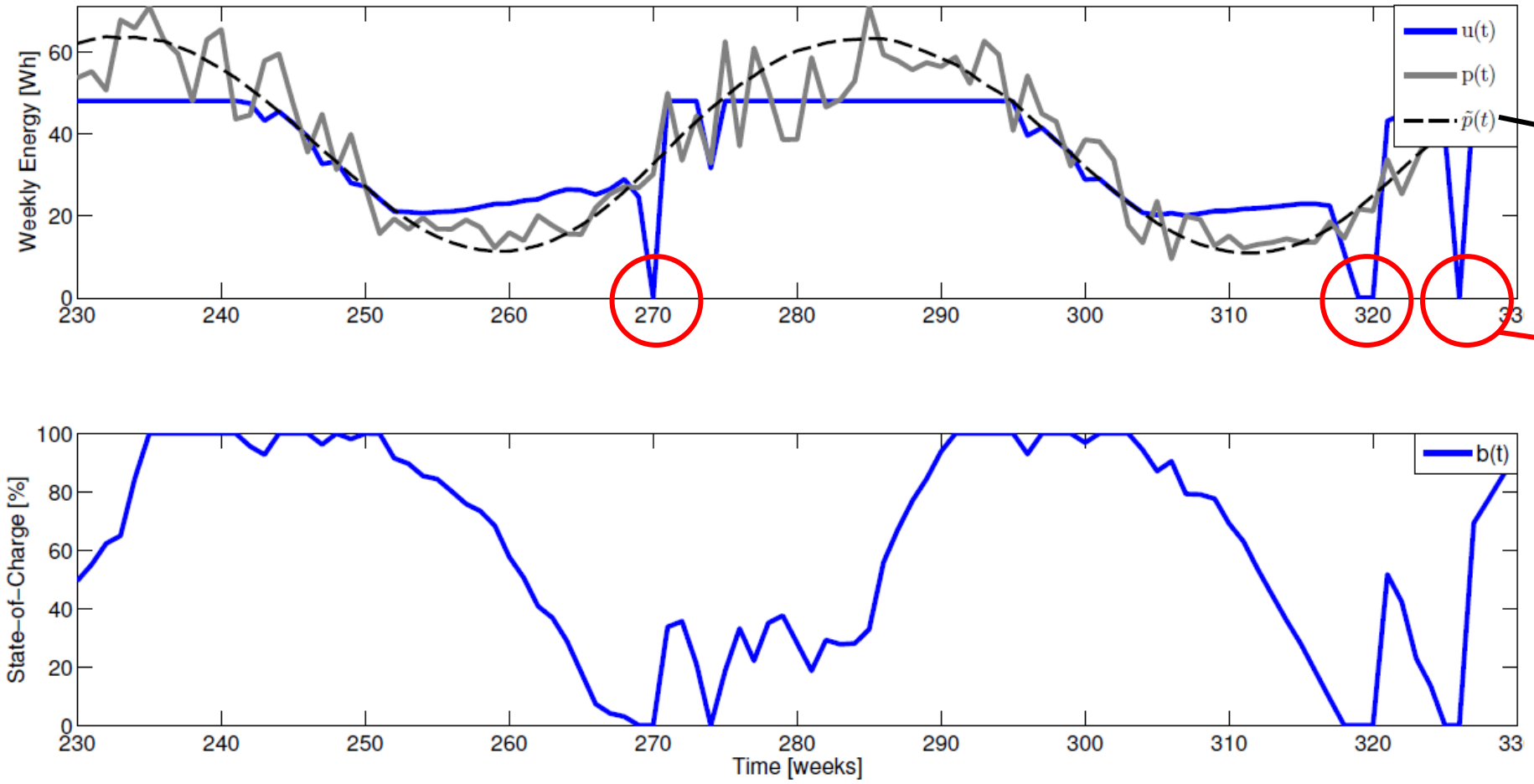


apply this use function in the interval [t, t+1).



compute the optimal use function in [t+1, t+T+1)
using the actual batter state at time t+1

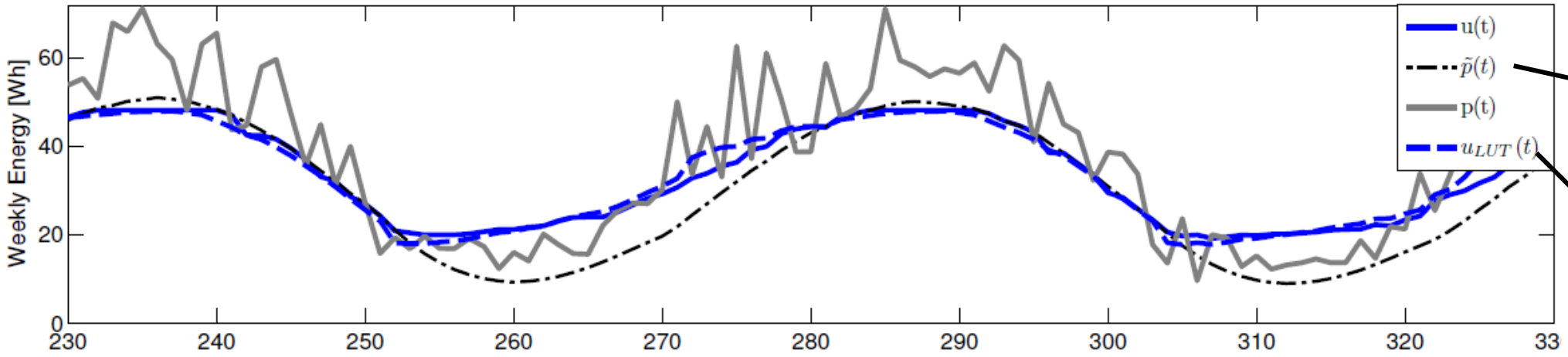
Application Control using Finite Horizon



estimated input energy

still energy breakdown due to misprediction

Application Control using Finite Horizon



more pessimistic prediction

simplified optimization using a look-up-table

[not covered]