# Power Management in IoT Systems

Hunter Adams, March 25 2022

# What I love about IoT projects

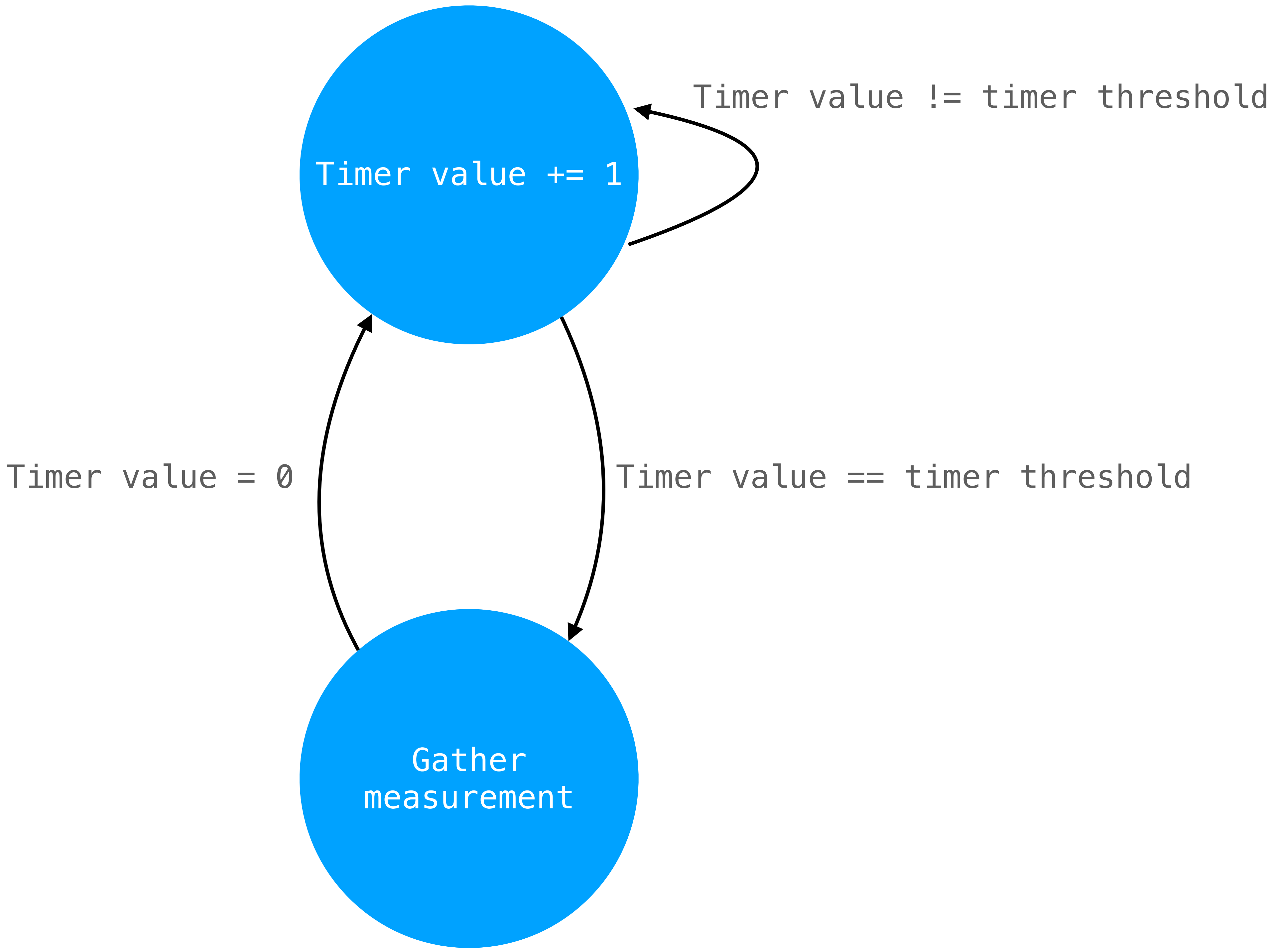
1. They are an avenue for learning about other things.
2. They offer constrained engineering.
3. The nature of debugging these systems, and the scope of their complexity

# Topics for today

1. Spending energy wisely
    - Sleep/wake cycles, and the coupling between hardware and software
    - Achieving ultra-low power modes
    - Methods for waking the system
    - Latching circuits for dead/alive cycling
    - Getting back to sleep as quickly/efficiently as possible

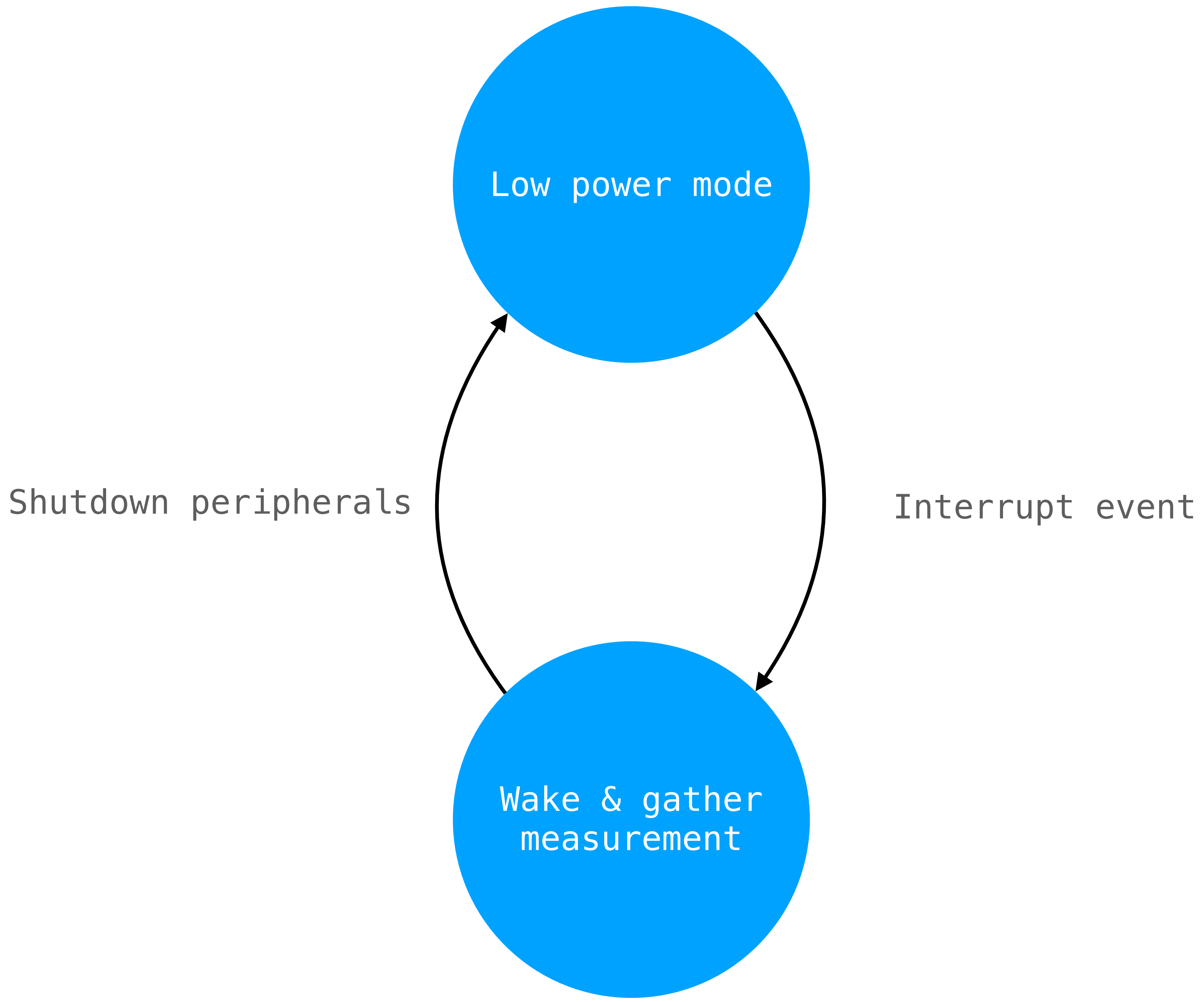
2. Gathering and storing energy
    - Solar cells/maximum power point tracking
    - Energy harvesting from gradients and energy distributors
    - Li-Ion charging curves

# Polling




Average power = $P_w$

# Sleep/wake cycles




Average power = $\alpha P_w + (1 - \alpha) P_s$

Low power mode

Fast wake cycle

Utilization of peripherals

# Achieving a low power mode

1. **Read the datasheet**
    - This is how you know when you're finished optimizing
    - CC1310 datasheet
    - RP2040 datasheet

2. Attempt to put the system in low-power mode
    - Turn off all peripherals that you aren't using (some are particularly expensive)
    - Decrease clock speed

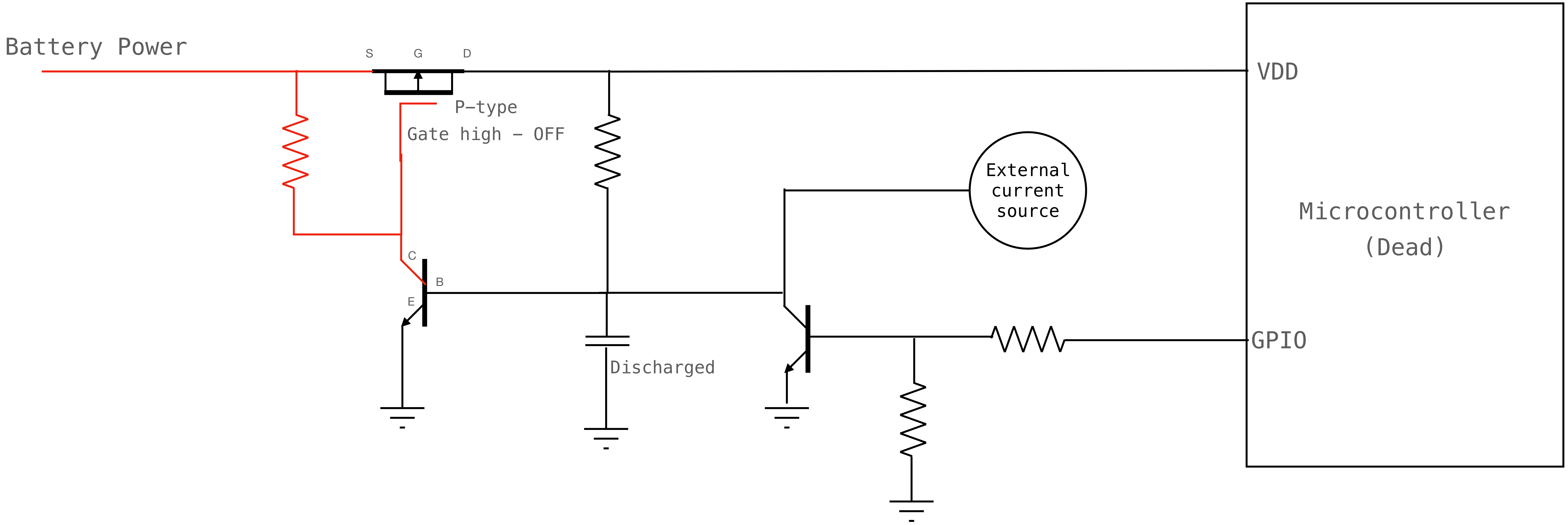
3. Measure current draw, compare to expected current draw
    - If you find excess power, check power draw from peripherals in datasheet
    - Still finding excess power? Start looking for leaks

# Waking up from a low power mode

1. Internal signals
    - Most often a timer interrupt, or an interrupt thrown by an internal RTC
    - Costs some power, but gives you regularity

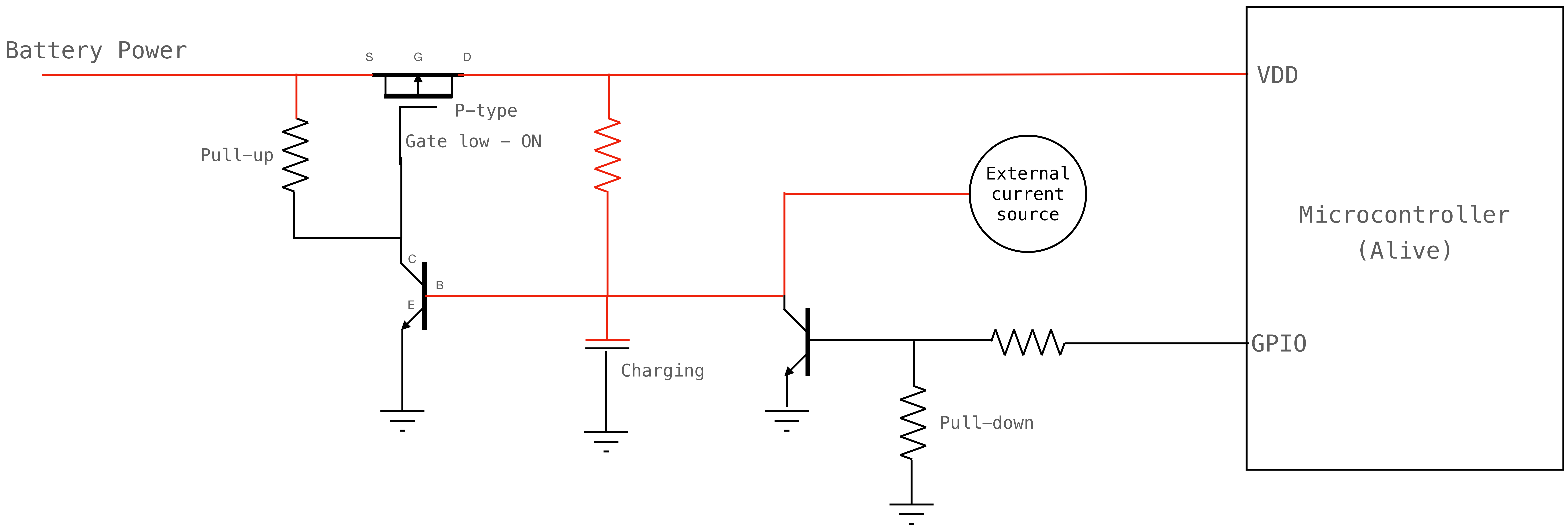
2. External signals
    - GPIO interrupt (piezo element, external RTC, etc)
    - Power being restored (see next slide)
    - Might cost you timing guarantees, but might also reduce power

# Latched off

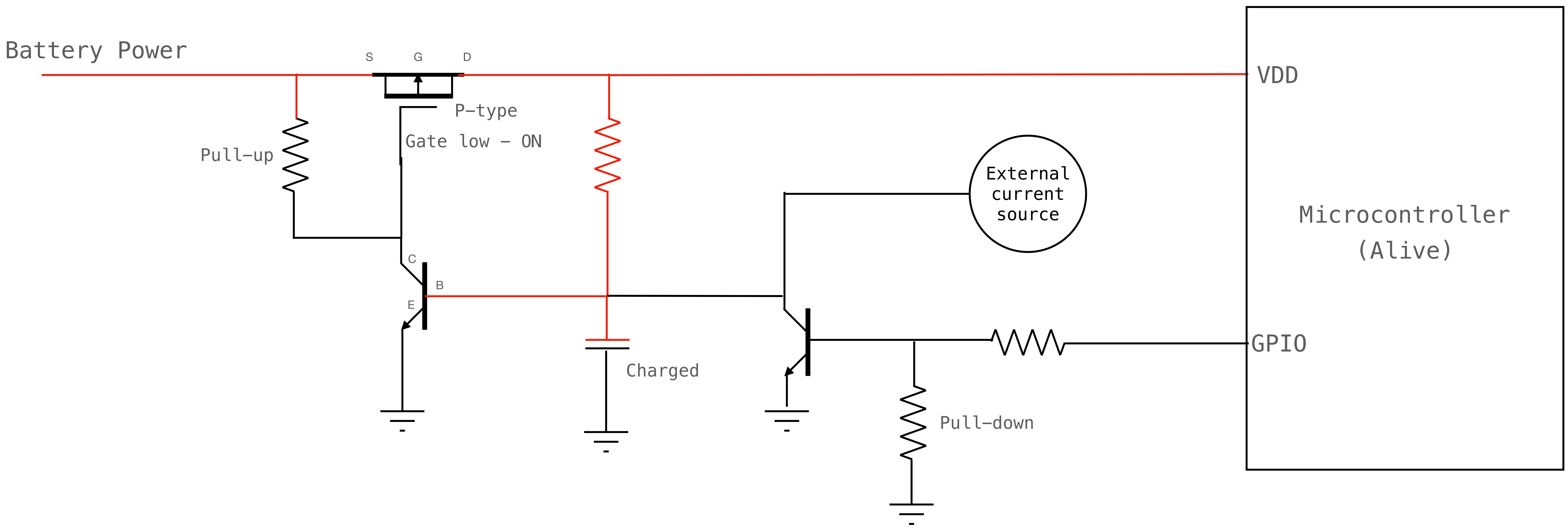


Battery Power
S
G
D
P-type
Gate high - OFF
VDD
External
current
source
Microcontroller
(Dead)
C
B
E
Discharged
GPIO

Toggled on by external current source (vibration sensor? Piezo element?)

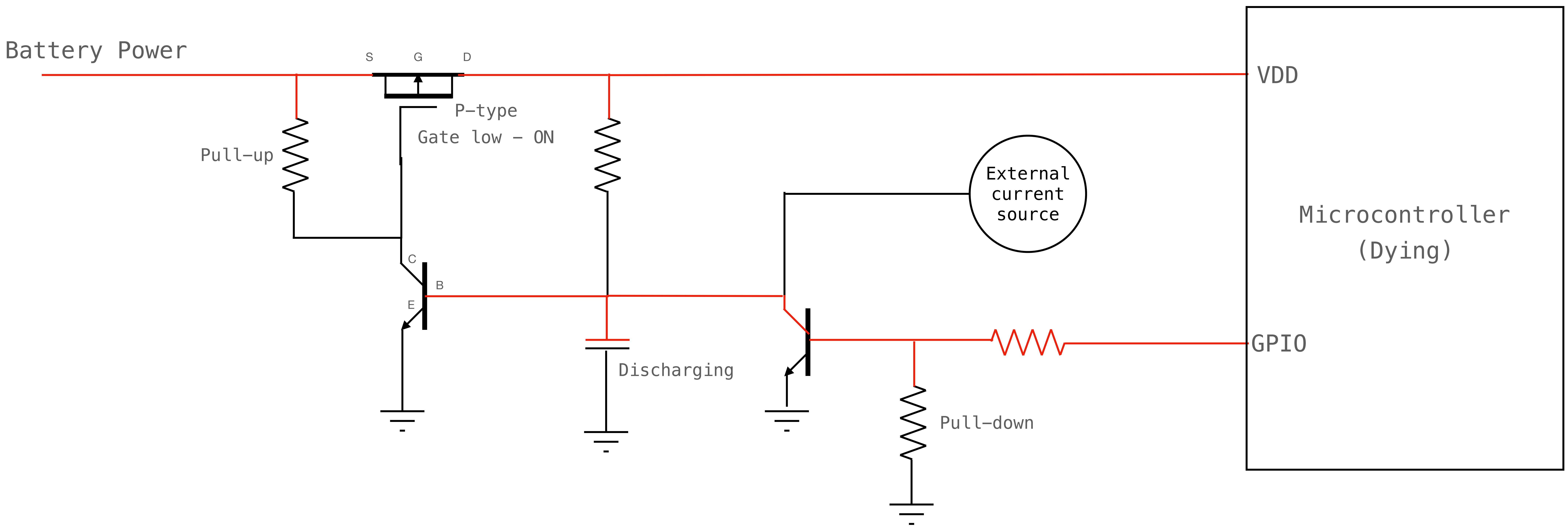


Latched on

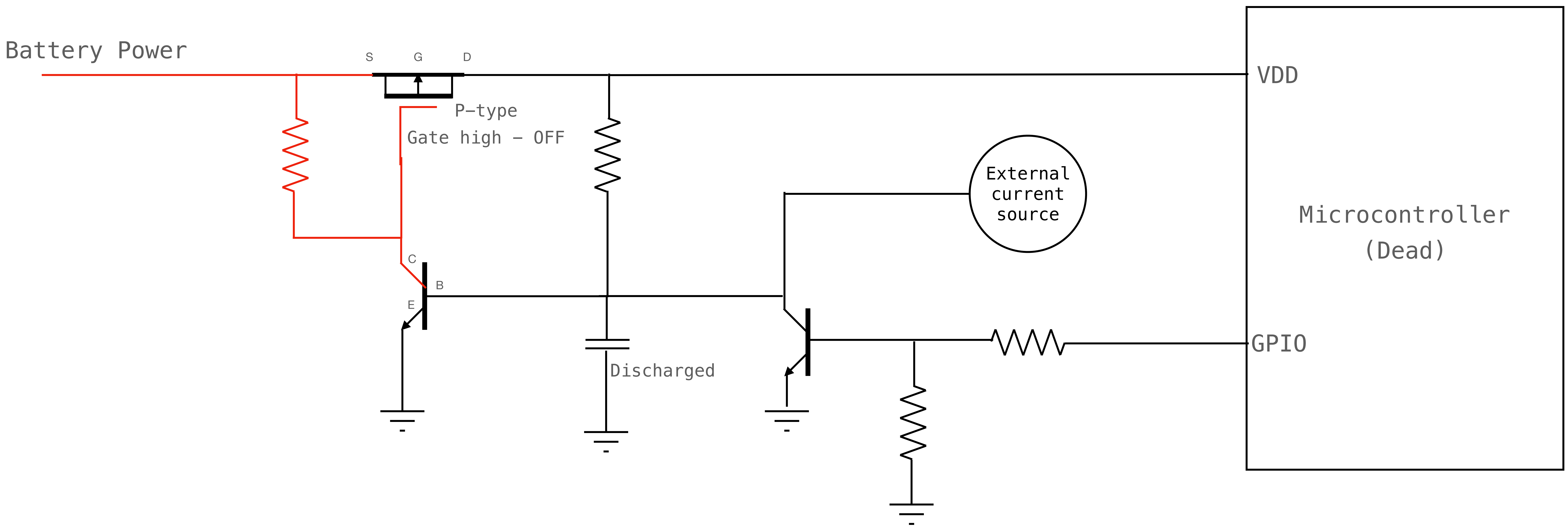


Battery Power
S
G
D
P–type
Gate low – ON
Pull–up
C
B
E
Charged
External current source
Pull–down
VDD
Microcontroller (Alive)
GPIO

# Toggled off by GPIO



Latched off



Battery Power
S
G
D
P-type
Gate high - OFF
C
B
E
Discharged
External current source
VDD
Microcontroller (Dead)
GPIO

Once your system wakes, you want to take your measurements and get back to sleep *as quickly and efficiently as possible*.

This means using hardware peripherals to accomplish tasks using the least possible power, and in parallel.

It also means performing necessary computation in the least possible number of CPU cycles.

Once your system wakes, you want to take your measurements and get back to sleep *as quickly and efficiently as possible*.

This means using hardware peripherals to accomplish tasks using the least possible power, and in parallel. → DMA

Fixed point ← It also means performing necessary computation in the least possible number of CPU cycles.

# Direct Memory Access (DMA)

Accomplish tasks in parallel and with less power

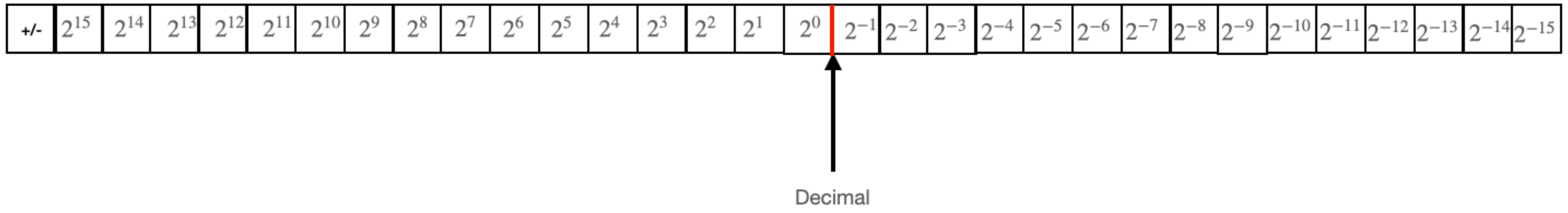
1. Available on nearly all microcontrollers (RP2040)
2. Simple co-processors that can only move data from one place in memory to another
3. Useful because peripherals are usually memory-mapped (including DMA control registers!!)
4. Lower power, and frees up CPU time for other tasks
5. Up to you to configure
    - Source address
    - Destination address
    - Start condition
    - Stop condition
    - Transfer block size
    - Transfer condition

# Fixed point arithmetic

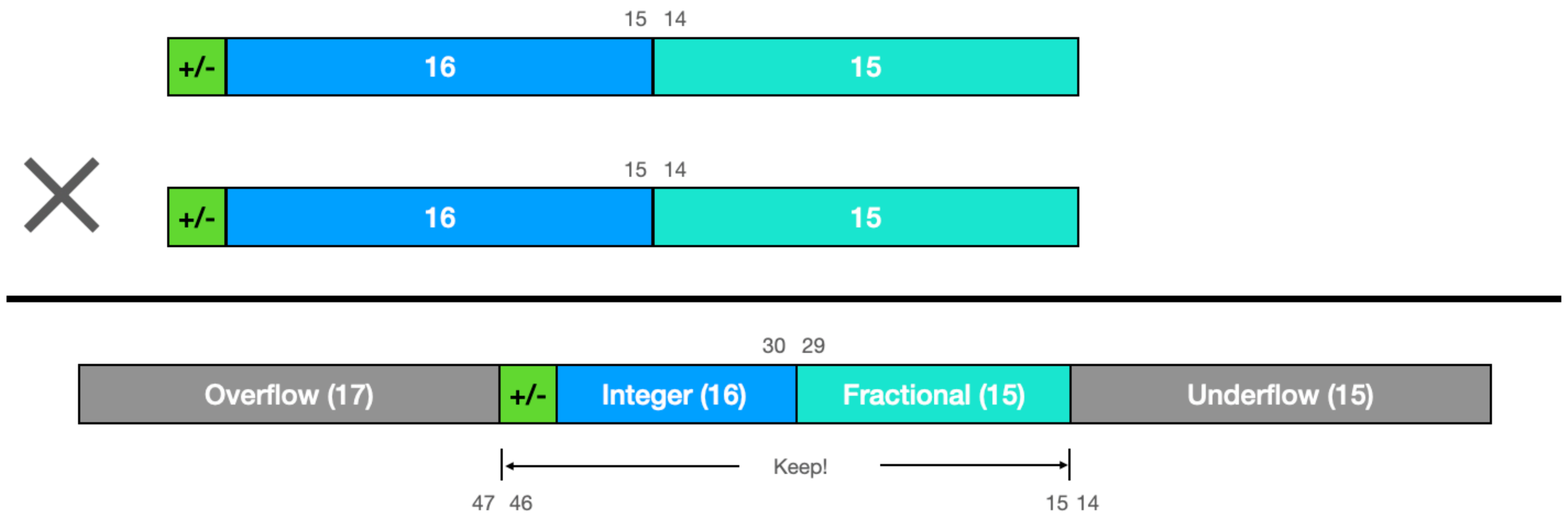
## Accomplish arithmetic faster

1. Mandatory knowledge for FPGA development, and tremendously helpful for accelerating arithmetic on processors without dedicated floating point hardware

2. Trades range for resolution in integer arithmetic




3. Requires that you write your own multiplication/division routines




[Video demo]

```
#define multfix(a,b) ((fix))(((( signed long long a))*((signed long long b)) >> 15))
```

# Topics for today

1. Spending energy wisely
    - Sleep/wake cycles, and the coupling between hardware and software
    - Achieving ultra-low power modes
    - Methods for waking the system
    - Latching circuits for dead/alive cycling
    - Getting back to sleep as quickly/efficiently as possible

2. **Gathering and storing energy**
    - Solar cells/maximum power point tracking
    - Energy harvesting from gradients and energy distributors
    - Li-Ion charging curves

# Solar Cells and Maximum Power Point Tracking

1. A solar cell is characterized by its IV curve
2. The maximum power point is a moving target, which varies based on temperature and environmental conditions
3. A MPPT is a closed-loop controller which measures panel voltages/currents, then updates the electrical characteristics of the load to maximize power transfer
4. Usually, this is accomplished by controlling the duty cycle to a buck converter

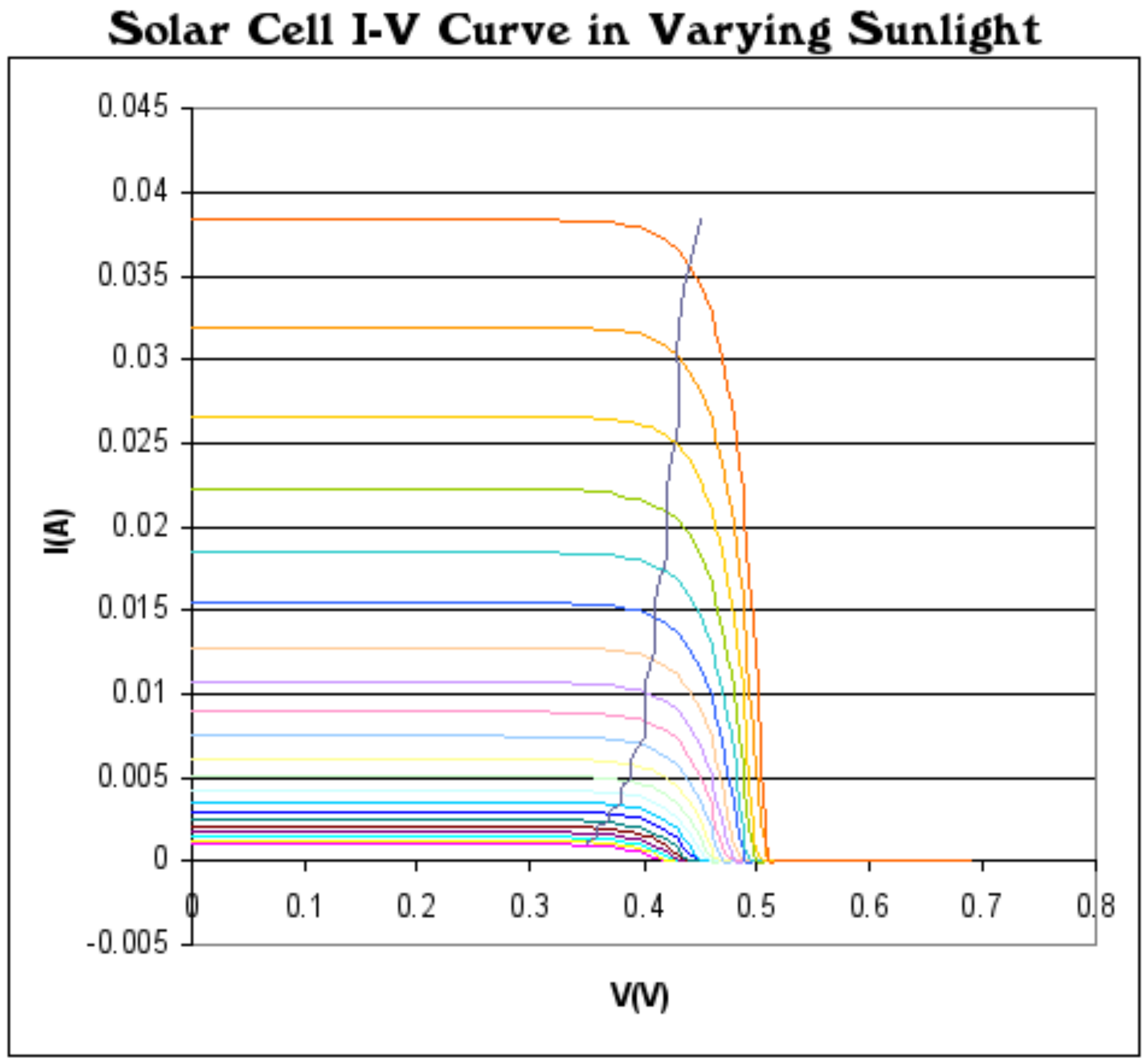



Image from Wikipedia

Often available as an integrated circuit

# Energy Harvesting

1. Look for gradients. Where there is a gradient, there is an opportunity to extract energy
   - Temperature, pressure (atmospheric, fluid, sound), magnetic field, gravitational potential, chemical, radioactivity, etc.

2. Look for energy distributers
   - Animals, people, vehicles

3. Black-swan energy events (maybe impossible, but fun to think about)
   - Hurricanes, landslides, volcanoes, tornadoes, lighting, Dyson-spheres

# Charging Li-Ion Batteries

**Be careful.**

1. Current-controlled phase: battery is charged by controlling the current into the battery to be ~0.5 - 1 C (2000 mAh battery —> C = 2000 mA)

2. Voltage across battery increases. When voltage reaches ~4.2 volts, switch to voltage-controlled charging. Maintain 4.2 V and measure current into the battery

3. When measured current falls to ~10% of rated current, stop charging. Overcharging the battery can lead to reduced capacity and fire. Undercharging can also lead to reduced capacity.

4. Sensitive to temperature! Can’t discharge too quickly!

**Don’t over engineer!**

Probably, all that matters is the data that your system gathers. Only make your system as complex as the application requires.

If you’re next to a power outlet, just plug it in! If you can support latency, log data. If changing batteries is easy, avoid a complicated solution!

<u>A case study</u>