Sensor Device

System Parameters (defined by hardware)	from the datasheets		Profiles (usage of each component mode - defined by software and usage)		
			Sleep	Monitoring	Detecting
Processor					
XIAO ESP32-C3	om/IrisYzh/Sm				
Active	154	mW	0%	3%	20%
Idle	3	mW	0%	97%	80%
Sleep	1	mW	100%	0%	0%
LED					
NeoPixel - SK6812RG	om/IrisYzh/Sm				
Full bright	195	mW	0%	0%	30%
Dark(10%)	20	mW	0%	100%	70%
Sensor					
TAL221	om/IrisYzh/Sm				
On	5	mW	0%	3%	15%
Idle	3	mW	0%	97%	85%
Off	0	mW	100%	0%	0%
HX711	om/IrisYzh/Sm				
On	8	mW	0%	3%	15%
Idle	3	mW	0%	97%	85%
Off	0	mW	100%	0%	0%
Radio: BLE					
Data Rate	300	bps	0%	0%	0%
Standby Power	7	mW	100%	80%	60%
TX Power	240	mW	0%	10%	25%

RX Power	210	mW	0%	10%	15%	
			16	6.5	1.5	hours/day typical usage
Battery						
Capacity	600	mAh				
Nominal Voltage	4	V				
Regulator Efficiency	99%					

Reflection

How did you determine your "days of use" metric?

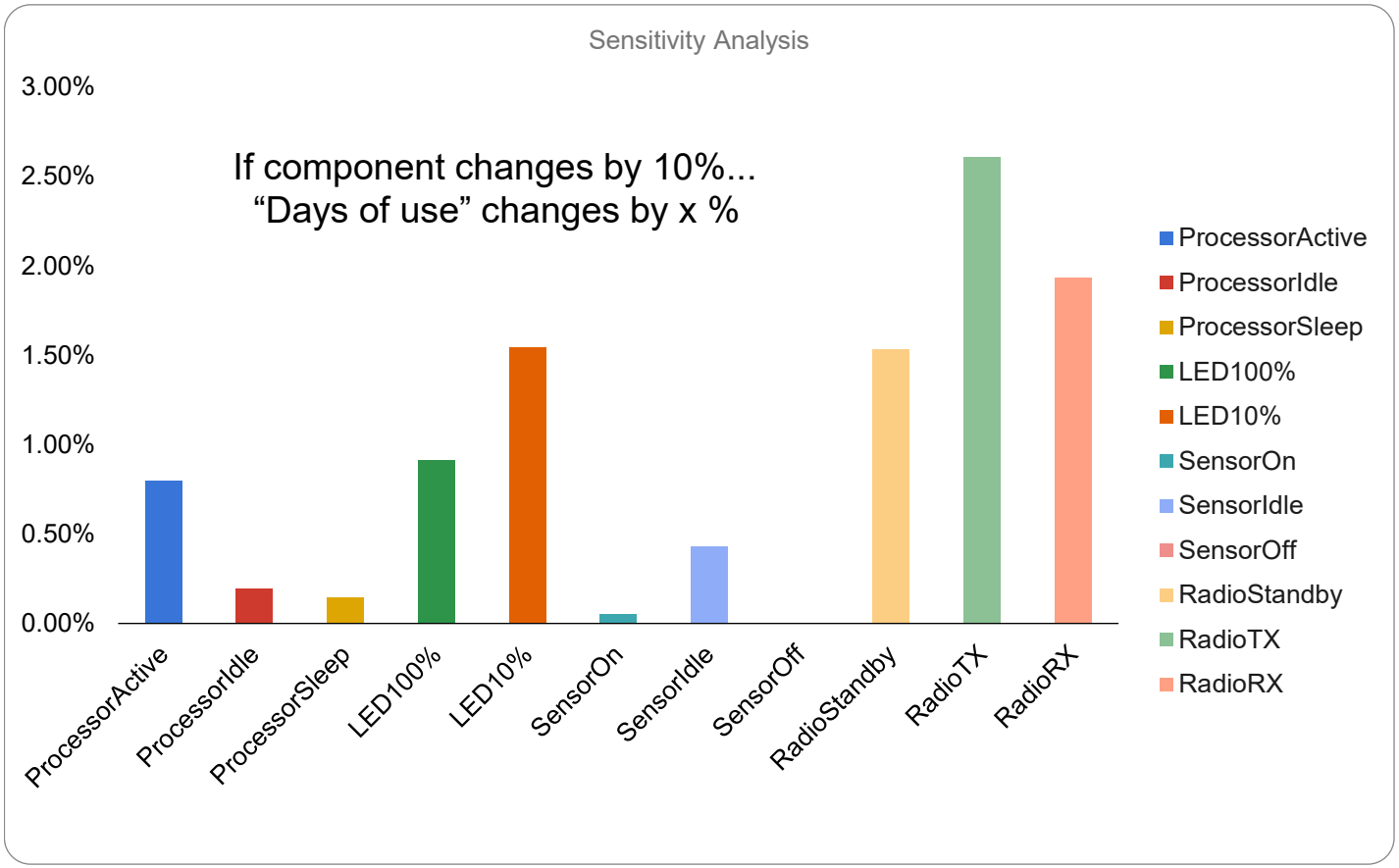
I measured power consumption in three operating modes: sleep at 7.5 mW for 16 hours, monitoring at 82.6 mW for 6.5 hours, and detecting at 207 mW for 1.5 hours. The total daily energy consumption is 967 mW·h. Dividing the battery capacity of 1099 mW·h by this daily consumption gives 1.14 days of use.

What do you think is the optimum size for the battery in your device?

A 600 mAh battery at 3.7V provides about 2.3 days of runtime, which is optimal for this device. This gives users flexibility to skip a day of charging without the device dying. Going smaller to 300-400 mAh would require daily charging, which is inconvenient if users forget to charge. Going larger to 800-1000 mAh would make the coaster thicker and heavier, which isn't ideal because the coaster needs to be thin and lightweight to fit under a cup.

What hardware/software/cost/effort tradeoffs could you make to improve the user experience?

The LED is the biggest power consumer at 55% of monitoring mode energy. I could turn it off completely when weight is stable and only flash briefly when detecting changes, which would cut LED power by 80% with just a simple code change. Also, the weight sensor checks every 3 seconds, but I could extend this to 5-10 seconds when no cup is detected, saving about 20% of sensor power without affecting user experience. I'm also thinking about reducing BLE transmission frequency from every weight change to only significant changes. This would save radio power but requires more intelligent filtering logic. The software optimizations are easiest to implement and could extend battery life to 2+ days without any hardware changes.



Total power in profile (mw)		Maximum Time	
Sleep	7.49 mW	293.4	hours
Monitoring	82.57645 mW	26.6	hours

Detecting	207.038 mW	10.6 hours

Effective Battery Capacity

2197.8 mW*h

Days of Use	**2.27 days**
Hours of Use	**54.54 hours**