Multi-Sensor Smart Sleep Dock: Engineering Design Notebook

1. Problem Definition & Motivation

In modern society, ‘sleep procrastination’ is a common phenomenon. Exposure to blue light from smartphone screens before bed suppresses the secretion of melatonin. Meanwhile, the feedback loops of social media render traditional ‘software-based solutions’ ineffective, as users can easily disable them with a simple swipe of their fingers. This product aims to fill this gap by using a hardware-based physical interlock mechanism. It forces users to physically disconnect from their digital devices. Only when the phone is physically separated from the user and certain environmental conditions are met can the sleep-enhancing mode be activated. This isn’t just a monitoring device; it’s a system designed to help users change their behaviors.

2. Design Ideation & Novelt

2.1 Rejected Concept: Smart Water Reminder

I first think about a device that can count your drinking time and remind you to drink water more often. It can also change the frequency if the temperature of the environment change. For example, when the temperature rises very high, the device will increase the frequency to remind people to drink more water. However, I give up this idea because it is too derivative. There are already too many similar mobile apps and smart water bottles on the market. Additionally, it’s not very useful because users can easily get used to them and ignore the alerts, making them ineffective in helping people change their habits.

2.2 Final Concept & Novelty: The Physical Interlock

To address the lack of constraints in previous ideas, I decided to incorporate the concept of an "interlock" from industrial circuit design. I extended the capacitor pins of the CPX to serve as a "phone holder." By running an alligator clip from a capacitive touch pin to a custom-made conductive phone cradle on my nightstand, the CPX can detect the precise physical presence of the phone.

The core novelty lies in shifting the microcontroller from a passive sensor to an active behavioral enforcement gate. The sleeping mode cannot initiate unless the phone is surrendered, and any relapse (picking up the phone) triggers an immediate state violation.

3. System Architecture & Code Logic

3.1 Variable Dictionary

Variable Name	Type	Description / Functionality
currentState	Integer	Tracks the active phase of the FSM (0=OFF, 1=ENV_CHECK, 2=WAIT_PHONE, 3=READY, 4=SLEEPING, 5=ALARM).
isPhoneDocked	Boolean	True if the capacitive touch pin (A1) detects the phone; False if lifted.
LIGHT_THRESHOLD	Integer	Maximum acceptable room brightness (set to 20 units).
SOUND_THRESHOLD	Integer	Maximum acceptable ambient noise (set to 60 units).
TEMP_MIN / TEMP_MAX	Integer	The safe temperature boundary (18°C to 25°C).
alarmFlashCounter	Integer	A non-blocking incremental counter used in the Alarm state to manage LED flashes without freezing the CPU.

3.2 Core Logic Control Flow

System Configuration and Initialization:

The system begins by defining the environmental safety thresholds, where light must be less than or equal to 20, sound less than or equal to 60, and temperature must remain within the 18°C to 25°C range. Upon boot, the initial system state is strictly set to State 0 (STATE_OFF).

Data Acquisition & Hardware Interrupt:

During the continuous execution loop, the system polls the light, sound, and temperature sensors while updating isPhoneDocked. The Global Hardware Interrupt evaluates if Button B is pressed. If triggered, it forces an instant transition to State 0, clears all LEDs, plays a power-down tone, and pauses for 0.5 seconds for software debouncing.

Mode Control Logic:

State 0 (OFF): All LEDs remain deactivated. Waits until Button A is pressed to transition to State 1.

State 1 (ENV_CHECK): Evaluates room conditions. Illuminates Orange LEDs if noise is too high, Yellow if light is too high, or Purple if the temperature is unsafe. If clear, displays Green LEDs and transitions to State 2.

State 2 (WAIT_PHONE): Maintains Green LEDs. Falls back to State 1 if the environment degrades, or transitions to State 3 if the phone is successfully detected.

State 3 (READY): Displays Blue LEDs, plays a lock-in tone, pauses for 3 seconds, clears all LEDs for total darkness, and moves to State 4.

State 4 (SLEEPING): Maintains strict physical interlocking. Jumps to State 5 if the phone is removed. Passively monitors the environment, silently illuminating warning colors if conditions degrade and clearing them when recovered.

State 5 (ALARM): Rapidly flashes Red LEDs and emits a high-frequency buzzer. Remains in this loop until the phone is returned, which falls back to State 1.

Current State	Trigger Condition	System Action	Next State
0 - STATE_OFF	Button A is Pressed	Flash Cyan LEDs; Play high-pitch boot tone	1 - STATE_ENV_CHECK
1 - STATE_ENV_CHECK	Environment is Optimal (All conditions met)	Turn LEDs Solid Green	2 - STATE_WAIT_PHONE
1 - STATE_ENV_CHECK	Environment degrades (Noise / Light / Temp out of bounds)	Display reactive warning colors (Orange / Yellow / Purple)	Remain in 1 - STATE_ENV_CHECK
2 - STATE_WAIT_PHONE	Environment suddenly degrades	Display corresponding warning color	1 - STATE_ENV_CHECK
2 - STATE_WAIT_PHONE	Phone is detected (isPhoneDocked == True)	Prepare system lock-in	3 - STATE_READY
3 - STATE_READY	Phone is lifted prematurely (!isPhoneDocked)	None	5 - STATE_ALARM
3 - STATE_READY	Phone stays docked for 3 seconds	Turn LEDs Solid Blue for 3s, then extinguish all lights	4 - STATE_SLEEPING
4 - STATE_SLEEPING	Phone is lifted without authorization	None	5 - STATE_ALARM
4 - STATE_SLEEPING	Room environment degrades during the night	Silently illuminate warning LEDs (No buzzer to protect sleep)	Remain in 4 - STATE_SLEEPING
5 - STATE_ALARM	Phone is surrendered back to dock	Extinguish Red lights; Stop alarm tone	1 - STATE_ENV_CHECK
[GLOBAL OVERRIDE]	Button B is Pressed (Any active state)	Force system shutdown; Extinguish all lights; Play power-down tone	0 - STATE_OFF

4. Iteration History (Attempts)

1. https://makecode.com/_8MoT699K3Lrz
    a) In the first version, lots of the codes are not working at all. Also, the function of this device is not well-developed.
2. https://makecode.com/_DPgDK6HpE71D
    a) In the second version, the code bugs are repaired but the detector of the phone depends on the capacity change which requires extra-device and it's hard to simulate. Therefore, I turn the detector of the phone into a button. The bottom should be continuously pressing to avoid the alarm.
3. https://makecode.com/_DHmAt3Y3DJas
    a) The basic function of the device is completed. However, there's condition like temperature that can be added as one of the conditions that should be met before sleeping.
4. https://makecode.com/_U55i2v9hxJpy
    a) The condition requirement of temperature is added and functioning well. However, there is an obvious bug for this code. If the simulating button for light never turn to zero, the simulating button for the sound will never appear.
5. https://makecode.com/_DumL5vKyMcAK
    a) The problem of the simulating buttons is solved. Now, there are three individual buttons, one is for temperature, one is for light, one is for sound. However, whenever the device turns into alarming mode, it won't stop anyways. Therefore, I need to develop a bottom that can control the start and the end of the device or the program.
6. https://makecode.com/_22t8tAMxJ08r
    a) The button A is added as the starting button for the program. It works well for the starting process. However, when the alarm starts to ring, it's hard to stop the program. It requires users to press it for many times.
7. https://makecode.com/_4k5hCk9bRMDp
    a) Finally, the problem is fixed. The button A is still for starting the program, but the button B is now for ending the program. And the program can end very quickly at any time.

5. Technical User Guide

5.1 System Specifications & Ideal Conditions

To guarantee optimal sleep quality, the device evaluates your room against scientifically backed environmental thresholds. The system defines a "Perfect Sleep Environment" when all the following criteria are met simultaneously:

- Ambient Light: $\leq$ 20 units (Pitch black or extremely dim ambient light)
- Ambient Noise: $\leq$ 60 units (Quiet room, ignoring natural breathing or faint fan hums)
- Room Temperature: 18°C to 25°C (64°F to 77°F)

5.2 Step-by-Step Operation

Step 1: Initialization. Prepare your bedroom for sleep. Press Button A to initiate the program. The device will flash Cyan and play a boot tone.
Step 2: Environmental Diagnostic. The device scans your room. If any factor prevents healthy sleep, a warning light displays: Solid Orange (Too noisy), Solid Yellow (Too bright), or Solid Purple (Temperature out of bounds). Correct the issue until the LEDs turn Solid Green.
Step 3: Docking the Phone. When Green, place your phone flat onto the docking platform. The lights will switch to Solid Blue, marking a 3-second secure lock-in grace period.
Step 4: Active Monitoring. After 3 seconds, all LEDs extinguish for total darkness. The device strictly monitors your phone. If room conditions degrade during the night, the device will silently glow the corresponding warning color to alert you without waking you up.
Step 5: Waking Up. In the morning, BEFORE lifting your phone, press Button B. The device will play a power-down tone and transition to OFF. You can now safely remove your phone.

5.3 Behavioral Penalty Routine (The Alarm)

If the phone is lifted while the system is locked-in without pressing Button B first, the LEDs will flash Red rapidly and a piercing 2000Hz tone will play.
To Silence: Return the phone flat onto the pad. The system will route you back to the Environmental Check. Alternatively, press Button B to force a master shutdown.

6. Troubleshooting Guide

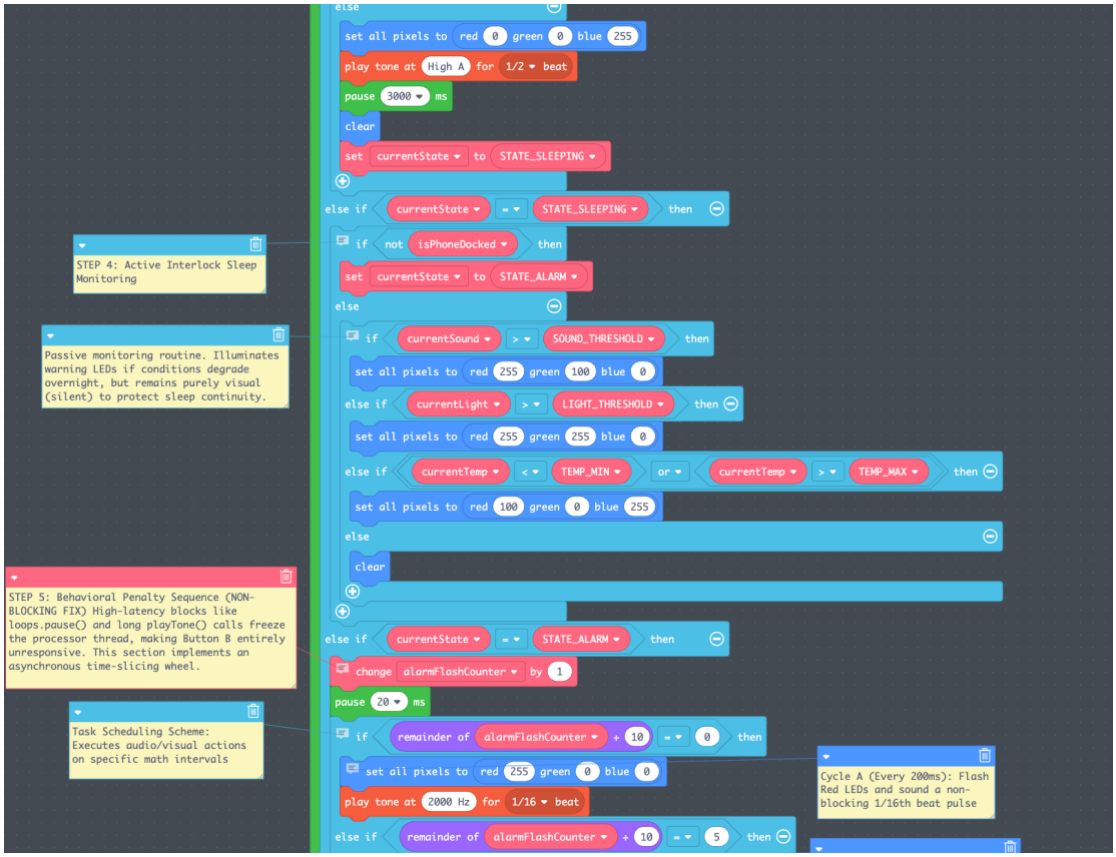
- System Responsiveness: If the system stays dead when pressing Button A, it lacks power or the loop is frozen. Check the USB connection, unplug it for 5 seconds to clear the memory, replug it to force a hard reset, and try again.
- Sensor Calibration: If stuck on a purple light when the room feels fine, the temperature is hovering just outside the strict 18°C–25°C window. Tweak your

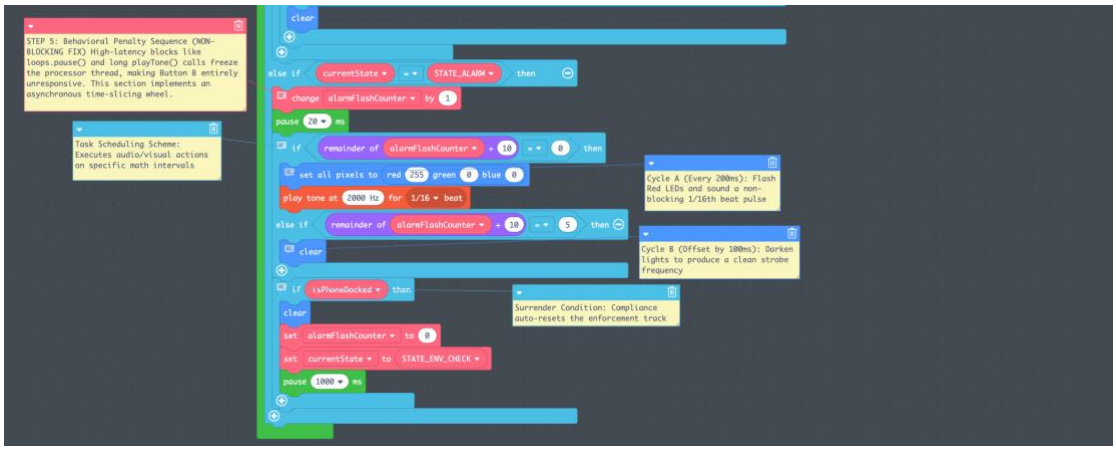
AC or heater. If the sensor is trapped in a warm spot on the casing, press Button B to shut down, let it cool, and restart with Button A.

- Docking Issues: If the Green light stays on but won't turn Blue when you put your phone down, Pin A1 isn't detecting capacitive touch. Make sure the phone rests flat against the metal contact. Thick plastic or rubber phone cases act as insulators; remove the case if the connection fails.
- False Alarms: If the alarm triggers randomly at night while the phone is flat, micro-movements are breaking the capacitive connection. Ensure the phone sits heavy and flush on the pad and move the dock away from unstable surfaces prone to vibrations.
- Missing Simulator Sliders: If the sound, light, or temperature sliders disappear in MakeCode, the compiler is hiding idle sensor blocks. To fix this and force the sliders to stay visible, always cache all sensor readings into independent variables (currentLight, currentSound, currentTemp) at the absolute top of the loop.

P.S. If the comments on the code blocks do not go well, here is the screenshots of them.

Environment satisfies requirements. Advance the user.
STEP 2: Interlock Onboarding Stage
Continuous validation: If room quality drops before docking, force re-diagnostic loop
STEP 3: Lock-In Countdown Phase
STEP 1: Environmental Diagnostics Block user progression unless bedroom parameters conform to healthy sleep boundaries.


STEP 4: Active Interlock Sleep Monitoring
Passive monitoring routine. Illuminates warning LEDs if conditions degrade overnight, but remains purely visual (silent) to protect sleep continuity.
STEP 5: Behavioral Penalty Sequence (NON-BLOCKING FIX) High-latency blocks like loops.pause() and long playTone() calls freeze the processor thread, making Button B entirely unresponsive. This section implements an asynchronous time-slicing wheel.
Task Scheduling Scheme: Executes audio/visual actions on specific math intervals
Cycle A (Every 200ms): Flash Red LEDs and sound a non-blocking 1/16th beat pulse


STEP 5: Behavioral Penalty Sequence (NON-BLOCKING FIX) High-latency blocks like loops.pause() and long playTone() calls freeze the processor thread, making Button B entirely unresponsive. This section implements an asynchronous time-slicing wheel.
Task Scheduling Scheme: Executes audio/visual actions on specific math intervals
clear
else if currentState = STATE_ALARM then
change alarmFlashCounter by 1
pause 20 ms
if remainder of alarmFlashCounter + 10 = 0 then
set all pixels to red 255 green 0 blue 0
play tone at 2000 Hz for 1/16 beat
Cycle A (Every 200ms): Flash Red LEDs and sound a non-blocking 1/16th beat pulse
else if remainder of alarmFlashCounter + 10 = 5 then
clear
Cycle B (Offset by 100ms): Darken lights to produce a clean strobe frequency
if isPhoneDocked then
clear
set alarmFlashCounter to 0
set currentState to STATE_ENV_CHECK
pause 1000 ms
Surrender Condition: Compliance auto-resets the enforcement track