# Timer-Based PC Wake-Up System

---

## 1. Project Title

**Timer-Based PC Wake-Up System**
Developed by: Ayushman Sahoo
February 2026
Version: v1.0

---

## 2. Abstract

This project, or sub-project, is to turn the server ON everyday at 6 AM using basic components and a cheap IKEA KUPONG digital alarm clock. A custom script checks the CPU usage for 2 mins at 12 AM everyday. If the Avg. CPU usage for 2 mins is less than 5%, meaning no background job is being performed, the script shuts down the server.

Then, to wake the PC up, I am made a simple circuit to check when the alarm beeps and then short the PWR Pins in the motherboard for few seconds untill the PC is on. Then it turns off the alarm by giving the SET button a HIGH signal.

So, overall, the PC shuts down at 12 AM everyday and wakes up at 6 AM to minimise load on fans and SMPS.

---

## 3. Problem Statement

I have made my own cloud storage server using NextCloud and Debian 12.6 for family use. It runs on a very old (11+ years) PC which doesn't have any modern BIOS features. As it is very old office desktop and not a server, it is not meant to run 24/7 for weeks at a time. This overuse has resulted in a seized CPU fan and a lot of headache. This also increases the load on SPMS. If the power supply’s fan gets seized, it may become a fire hazard.

---

## 4. Objective

- Check for average CPU load for 2 minutes, if it is less than 5%, proceed. Otherwise check for it again after 5 mins.
- Turn off the server at midnight when load is minimum.
- Start the server by shorting the PWR pins at 6 AM.
- Stop the alarm beeping after startup.
- Use as simple and passive components as required.

---

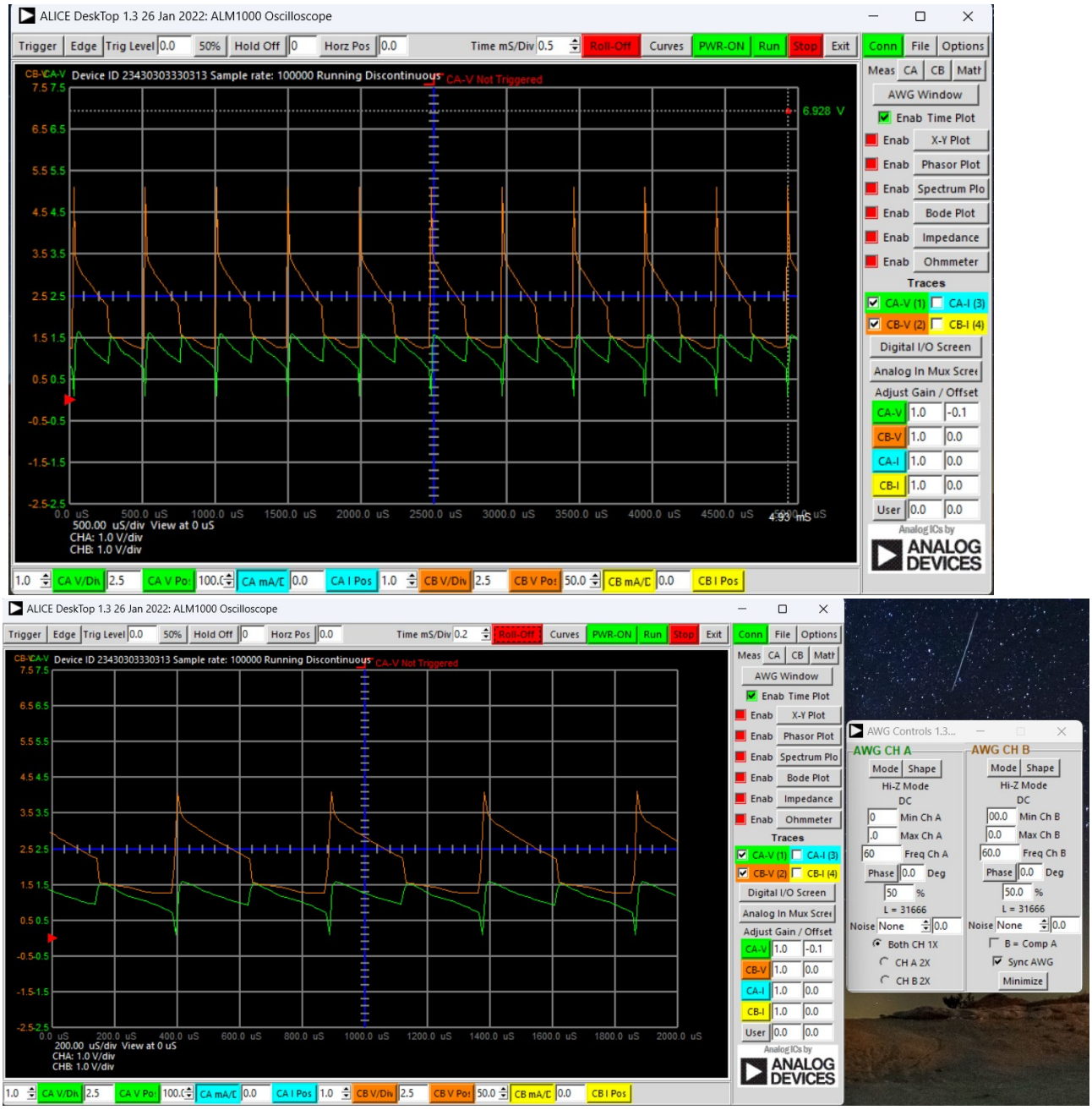
# 5. System Overview

## 5.1 Understanding the Clock

This would also be more like a report as I understand how this clock works.

At first I thought that the buzzer would have two pins, one of which would be negative and the other one connected to IC. But in reality this clock uses a positive driven buzzer, where one terminal is connected directly to +1.5V of battery and the other one is connected to IC. Interestingly when checked with multimeter, both the terminals seem to be in direct short.

Actually, this piezo buzzer is like a capacitor and when not in use, both the terminal show 1.5V but when it beeps, the IC produces an AC waveform which drives the buzzer. This waveform can be seen using an oscilloscope. Instead  I am using M1K ADALAM1000 device to see it with ALICE desktop software. I probed CH A to the negative terminal and CH B to the IC output (input to the buzzer).



# Waveform Analysis

**Screenshot 1**: CH-A (buzzer right pad) shows ~1.5V DC bias with 0.3-0.4Vpk-pk AC ripple during beep. CH-B (GND) flat 0V reference. Timebase ~2ms/div reveals ~800-1kHz frequency (cycle every ~1.25ms). Rising edges sharp (~200µs rise time).

**Screenshot 2**: Both channels stable ~1.5V DC during silence—no ripple, confirming beep-specific AC component. Vertical scale ±2V range captures full 1.1-1.9V signal swing.

**Trigger suitability**: 0.3Vpk edges provide >0.65V spikes through 1µF capacitor—sufficient for BC547 Vbe threshold

Apart from the buzzer, there are two buttons which are connected to positive terminal and to two IC pins.

---

### 5.2 Working Principle

At 12 AM, the linux script checks for average CPU usage for 2 minutes. If the average is less than 5%, it means that no background process is going on and its safe to shut down. If it is busy, it checks again after 5 minutes. And if everything is clear it shutsdown the pc.

At 6 AM, the alarm rings.

---

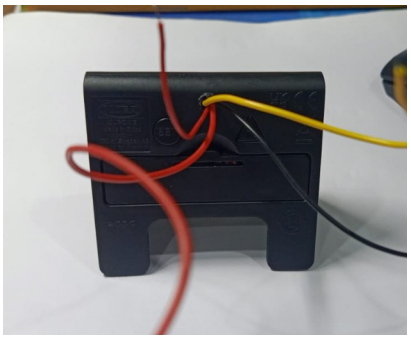
### 5.2 Block Diagram

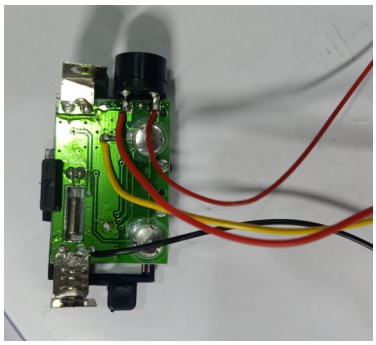
---

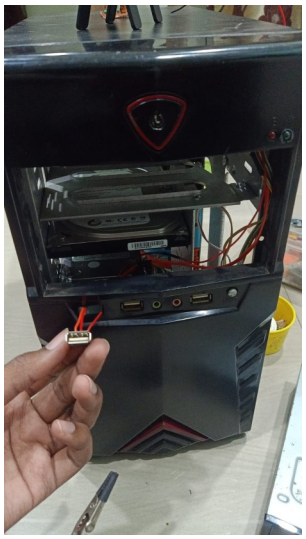
# 6. Components Used

---

# 7. Implementation





The above are the pictures of the circuit made on a perf board. Almost all the components are reused, it doesn’t look that great.

Explain step-by-step:

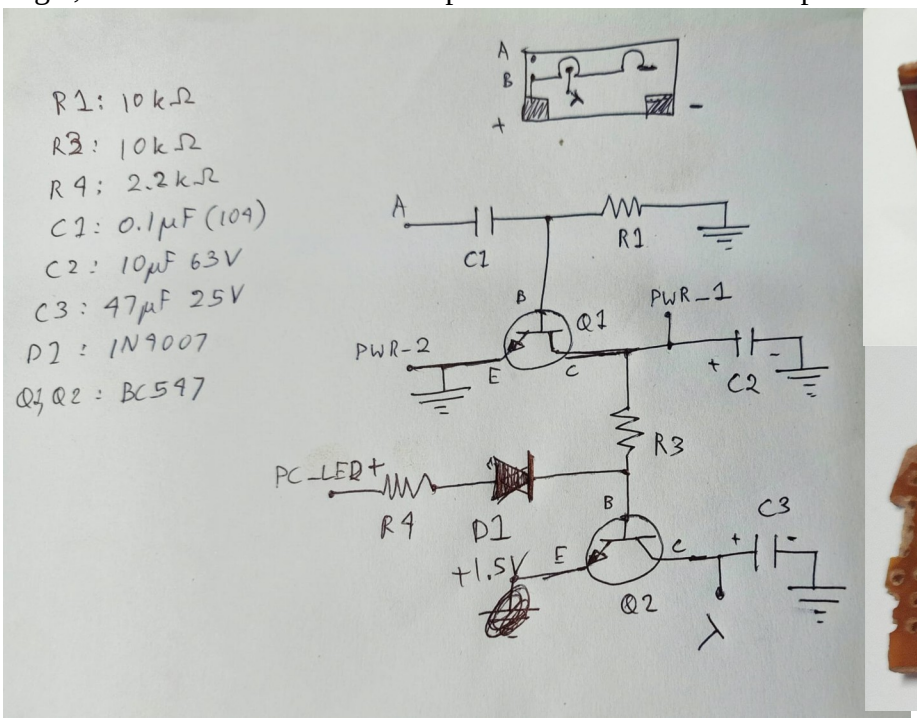
1. Setup timer module
2. Program time trigger
3. Connect relay across PC power switch pins
4. Test activation

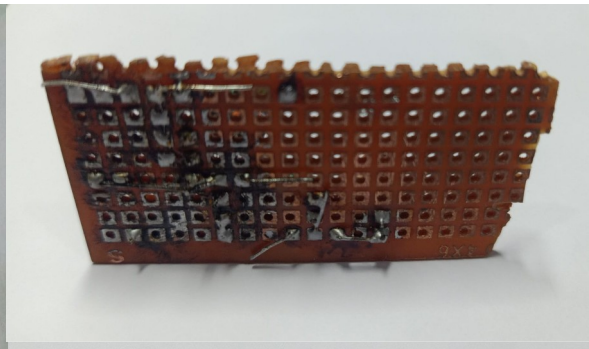
Keep it chronological.

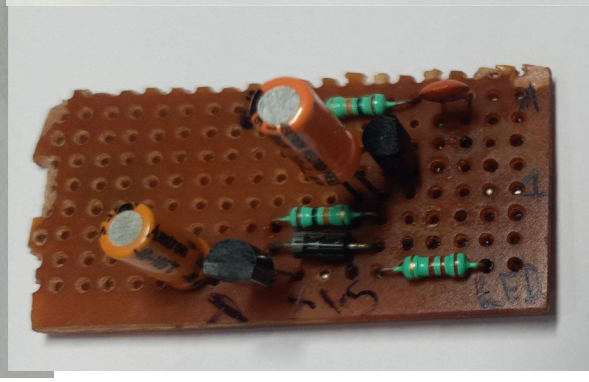
---

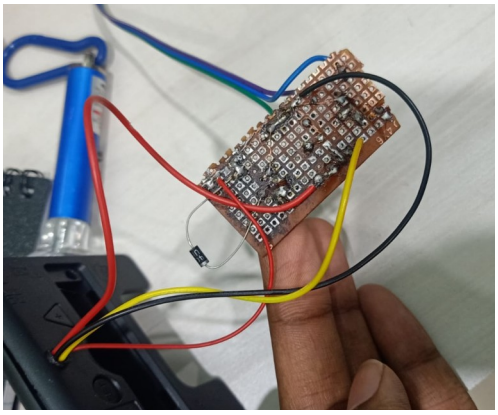
# 8. Failed attempts

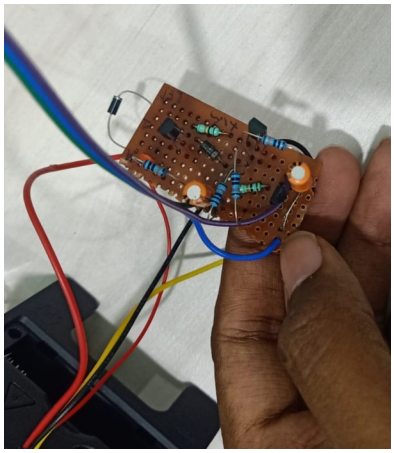
Initially I just thought of following ChatGPT blindly and trust it. To nobody's surprise, it didn't work, only the turning off the alarm part worked. Then I tried again with that but a bit of my own logic, it failed too. Here are some photos of those failed attempts.









## 9. Code (If Applicable)

Add:

- Key logic
- Important code snippet
- Explanation of algorithm

Do not paste 500 lines.
Only important part.

---

## 10. Testing & Results

- Was it successful?
- Delay accuracy?
- Reliability over multiple tests?
- Any failure cases?

Even simple results are fine.

---

## 11. Challenges Faced

- Incorrect timing?
- Relay bounce?
- BIOS restrictions?
- Wake failure?

Short but honest.

---

## 12. Limitations

Example:

- Requires physical access to motherboard
- Depends on power availability
- No battery backup

---

## 13. Future Improvements

- Add LCD display

- Add web interface
- Battery backup
- Multiple schedules

---

## 14. Conclusion

Summarize:

- Objective achieved?
- Cost effectiveness?
- Practical usability?

---

## 15. Appendix (Optional)

- Circuit diagram
- Full code
- Extra photos