Jacob Vosmaer's blog

My favourite computer ergonomics hack

2024-12-31

In this post I will talk about my favourite computer ergonomics hack, a DIY device I call "The Beeper".

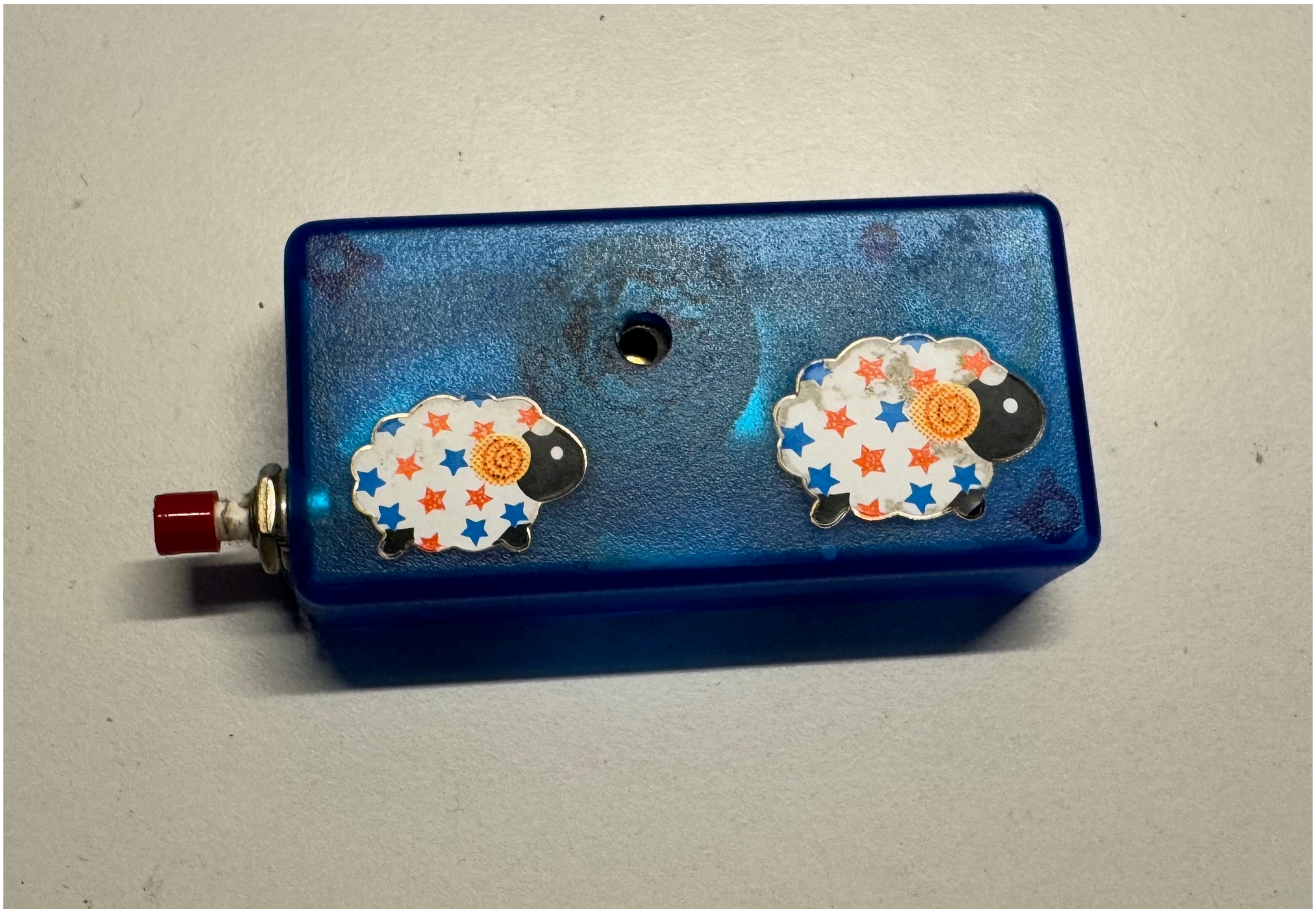
Background

I built the Beeper almost 8 years ago but I have never written a blog post about it and I thought it might be interesting. I do computer work sitting down at a desk at home. When I get focused on my work then I sit still for too long and my body starts hurting. The Beeper solves the sitting still problem.

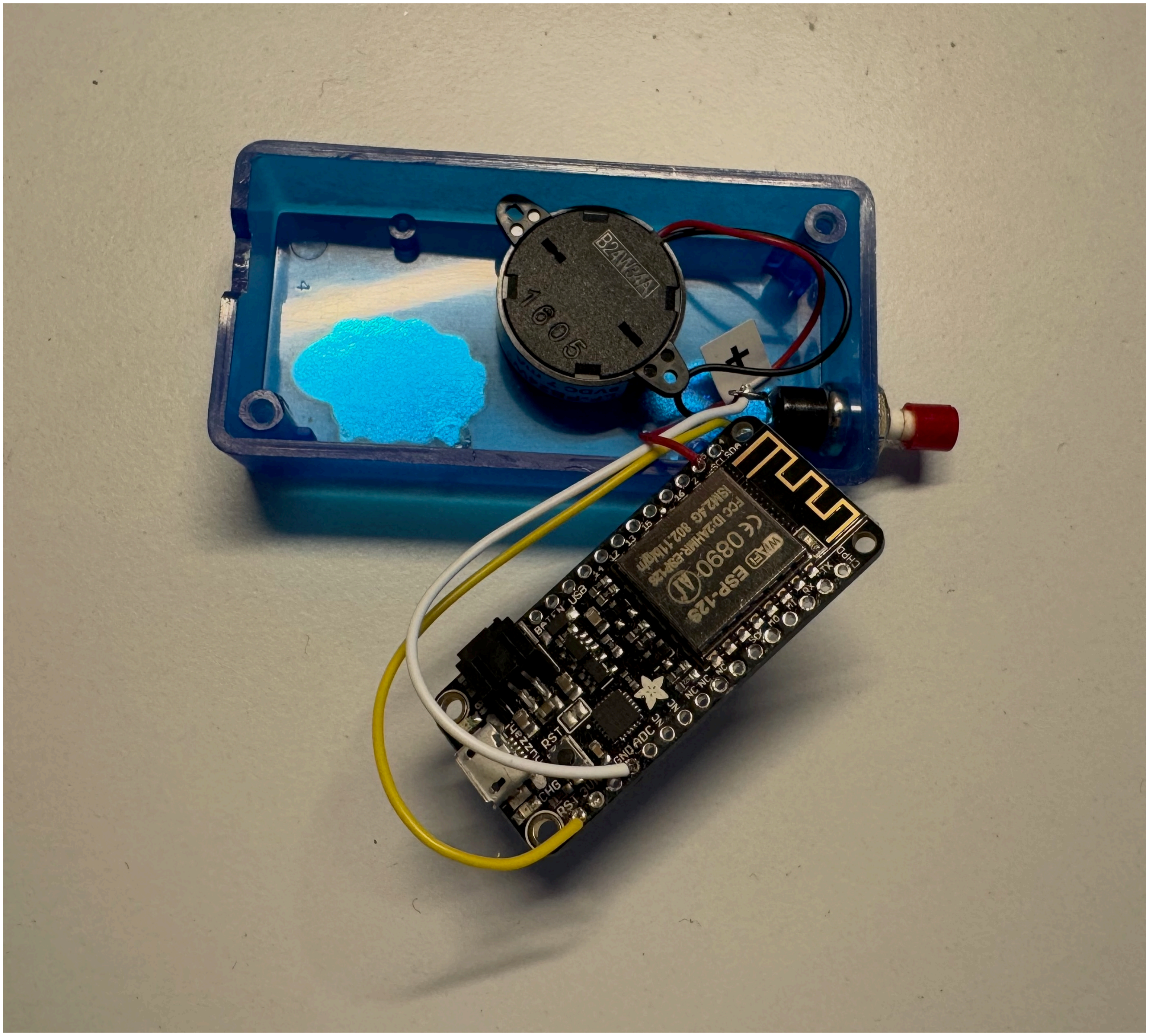
The Beeper consitst of three parts: the hardware, the firmware runnning on the hardware, and software running on my work computer. If my computer screen is unlocked for too long, the Beeper starts beeping and because it is situated away from my desk I must get up to silence it. Mission accomplished: I stopped sitting still.

Hardware

The Beeper is built into a small ABS Hammond 1551KTBU enclosure.



Inside the enclosure are an Adafruit Feather Huzzah ESP8266 development board, a momentary switch and a piezo buzzer. The buzzer is glued to the case.



The switch connects to the RST and GND pins of the Huzzah. Pushing the switch resets the microcontroller. The buzzer is connected to a PWM pin and GND.

The Huzzah is powered over its Micro-USB port. I cut a hole in the case that lets the USB plug come through. ABS plastic is easy to drill, cut and file.

Firmware

I'm a little fuzzy on how I got the firmware working because it's so long ago and I don't work with ESP8266 or Lua often. I had to upload a NodeMCU binary blob to the development board. I then somehow configured the ESP8266 to connect to our home WiFi network and when it starts up it runs the following Lua script.

```
-- init.lua: Beeper firmare

pin0 = 3
```

```
pin2 = 4
pin5 = 1

-- Periodically blink the LED connected to pin0
function blink()
  gpio.mode(pin0, gpio.OUTPUT)
  gpio.write(pin0, gpio.HIGH)
  tmr.create():alarm(10000, tmr.ALARM_AUTO, function()
    gpio.write(pin0, gpio.LOW)
    tmr.create():alarm(100, tmr.ALARM_SINGLE, function()
      gpio.write(pin0, gpio.HIGH)
    end)
  end)
end

-- TCP server that lets us remote-control the device over WiFi
function startserver()
  sv = net.createServer(net.TCP, 30)
  gpio.mode(pin2, gpio.OUTPUT)
  gpio.write(pin2, gpio.HIGH)
  gpio.mode(pin5, gpio.OUTPUT)
  if sv then
    sv:listen(5678, function(conn)
      conn:on("receive", function(sock)
        sock:close()
        gpio.write(pin2, gpio.LOW)
        beep()
      end)
    end)
  end
end

-- The beeping magic
function beep()
  freq = node.random(500) + 500
  delay = node.random(1000) + 1000
  pwm.setup(pin5, freq, 500)
  pwm.stop(pin5)
  tmr.create():alarm(delay/2, tmr.ALARM_SINGLE, function()
    pwm.start(pin5)
    tmr.create():alarm(delay, tmr.ALARM_SINGLE, beep)
  end)
end

blink()
startserver()
```

The high level description is that this script blinks an LED every 10s to show that the microcontroller hasn't halted and it starts a TCP server on port 5678. If anything connects to that port the connection handler calls a function called `beep()` which effectively loops forever because it keeps rescheduling itself with a timer. Every `beep()` invocation produces a shrill, annoying beep of random pitch and duration. I added the randomness because it makes it harder to mentally tune out the noise.

The only way to get this annoying beeping to stop is to reset the microcontroller with the switch.

Software

The software running on my laptop is the part I have had to tweak the most over the years. It has the biggest impact on the user experience and it is important to get this right, because if the Beeper is too annoying I will unplug it and then I no longer get the health benefits.

The current iteration of the laptop software is the following script.

```
#!/bin/sh
set -e
```

```
screen_locked() {
  ioreg -n Root -d1 -a | grep -q CGSSessionScreenIsLocked
}

external_monitor() {
  # The model identifier of my external monitor is EV2785
  system_profiler SPDisplaysDataType | grep -q EV2785
}

do_sleep() {
  sleep_time=1200
  expect_done=$(($(date +%s) + $sleep_time))
  sleep $sleep_time
  delta=$(($(date +%s) - $expect_done))
  if [ $(($delta * $delta)) -gt 100 ]
  then
    echo 'clock skew'
    exit 0
  fi
}

main() {
  do_sleep

  if ! screen_locked && external_monitor && ! pgrep -q zoom.us
  then
    echo hello | nc $IP $PORT
  fi
}

main
```

This script is started automatically by a macOS LaunchAgent. It sleeps for 20 minutes (which is the time I'm allowing myself not to move), checks if it is appropriate to beep and if so it activates the Beeper. If not then the script exits, macOS restarts it and we wait another 20 minutes.

If my screen is locked then I'm probably not sitting at my desk so we should not beep then. If I'm using my laptop somewhere else in the house away from my desk then the beeping would also be too annoying so we check if the external monitor that sits on my desk is connected to the computer. Finally, after many Zoom calls where I had to ask people to wait while I got up and turned off the beeper, I decided it would be better to not beep while I'm in a meeting.

It's certainly less awkward to not have the Beeper do its magic during a video call but in my experience video calls can be even worse than focused programming when it comes to sitting still too long. Arguably, I should not be suppressing the Beeper then. But I just got fed up with having to explain to the other participants in Zoom calls why I have to get up all the time.

Conclusion and acknowledgments

It's a silly device but it works. I am happy about how simple it is. It appears that the 10 seconds it takes for me to get up and push the button are enough to counteract the discomfort caused by sitting still too long.

As a closing thought I want to acknowledge and thank my wife for pointing out to me that I sit still too much and for putting up with the horrible screeching noises from the Beeper for the past 8 years and counting.

Tags: diy

Back