Flashing an “LED” (Part 2)
A Better Busy Wait Timer

Computer Systems Lecture 8.5

Busy Wait v1

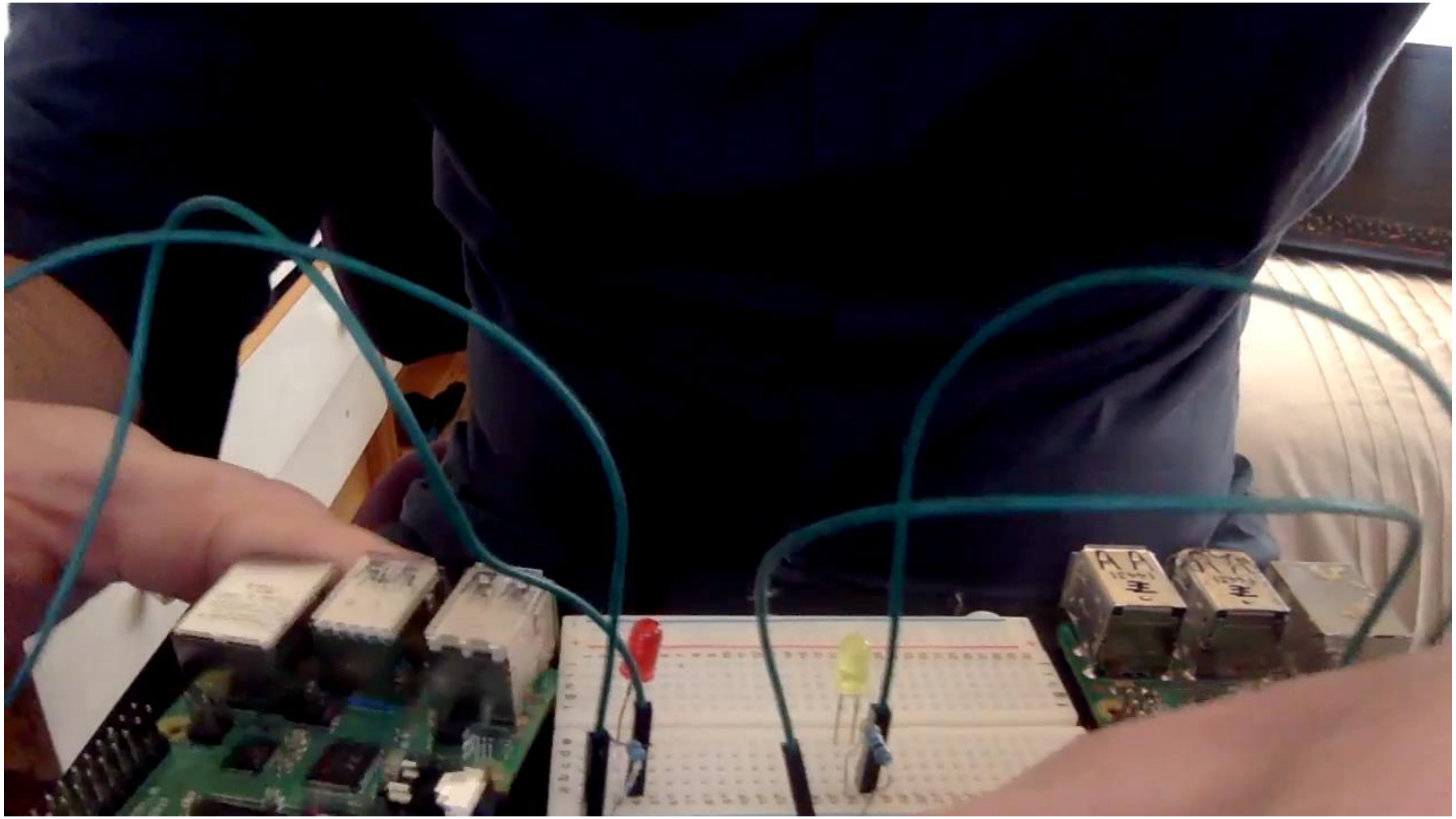
- Last video we applied the “dumb” busy wait timer:

```
MOV R2, #5000000          // init number of iterations for timer
MOV R2,R3                 // MOV iterations into working register R3
timer1:
      SUB R3,R3,#1        // subtract 1 from R3
      CMP R3, #0          // compare R3 with #0
      BNE timer1          // keep looping until R3 reaches zero
```

Any issues ?

- Yes !
- Consider executing this assembly code on real hardware and ask yourself “how long will the delay be?”
- Imagine you executed the code on two different CPUs ?
  - EG
    - RPi 2B (900MHz quad-core ARM Cortex-A7 processor)
    - RPi 4B (1.5GHz quad core ARM Cortex-A72 processor)
- Will the delay be the same on both ?
  - Lets see !

Busy Wait Flash on RPi4 (left) vs RPi2 (right)



A Better Busy Wait

- Most CPUs have access to a register that maintains a real time count.
- On the Raspberry Pi for example, a Timer register counts in 1 microsecond intervals (10^6 per second)
- In ARMlite, a register maintains a 1 second counter
  - accessible via the pre-defined .Time label
  - Eg `LDR R0,.Time` loads the current time in seconds (since Jan 1 2000) into R0
- We can utilise this to produce a hardware independent real-time delay

A Better Busy Wait - psuedocode

```
Set desired delay time in seconds
start_time = current time in seconds
loop:
   now = current time in seconds
   remaining_time = (now - start_time)
   compare remaining_time, delay
   loop if remaining_time <= delay
```

A Better Busy Wait

- Let’s allocate some variables to registers
  - R2 - desired delay
  - R3 – now (where the current time will be stored)
  - R4 - start time
  - R5 - elapsed time (R3-R4)

A Better Busy Wait

- Let’s allocate some variables to registers
  - R2 - desired delay
  - R3 – now (where the current time will be stored)
  - R4 - start time
  - R5 - elapsed time (R3-R4)

Pause the video and have a go!

You might want to edit the code from the last video, and replace Busy Wait v1 with new improved Busy Wait

A Better Busy Wait - solution

```
        mov r0, #.green
        mov r1, #.white
        mov r2, #1          ; 1sec delay time
flash:
        str r0,.Pixel367    ; flash on
        LDR r3,.Time        ; start time
timer1:
        LDR r4, .Time       ; current time
        sub r5,r4,r3        ; elapsed time = current time - start time
        CMP r5,r2           ; compare elapsed to delay time
        BLT timer1
        str r1,.Pixel367    ; flash off
        LDR r3,.Time        ; start time
timer2:
        LDR r4,.Time        ; current time
        sub r5,r4,r3        ; elapsed time = current time - start time
        CMP r5,r2           ; compare elapsed to delay time
        BLT timer2
        B flash
        halt
```

A Better Busy Wait

- Provides an absolute timer (no longer dependent on processor speed!)
- However, our approach is still a form of busy wait !
  - CPU still occupied for duration of delay
  - Essentially this is polling the timer
- Is there a potentially even better approach ?
  - Well yeah! We could implement an interrupt-based timer
  - We will come back to this later in semester!