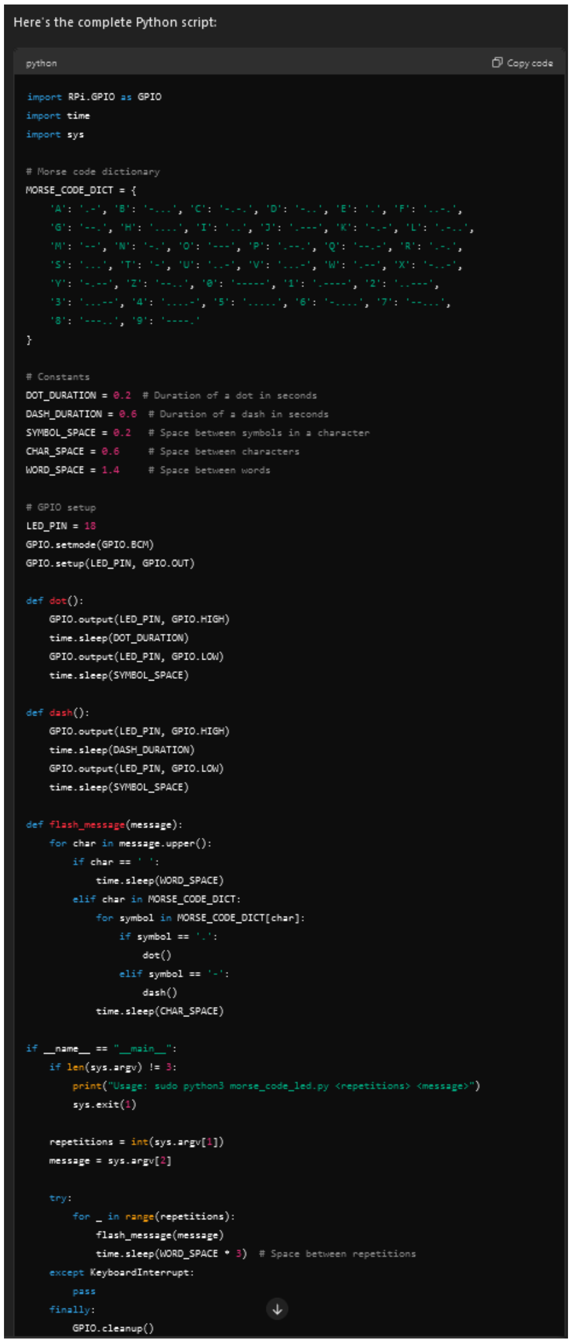
Lab 5 Report – Ryan King

Lab 4 Part 1:

Part 1 was done entirely by ChatGPT with minor adjustments from me.

https://chatgpt.com/share/d01ec6a4-4b10-4875-9c17-2cd909e838e3

“Write a program for a raspi-4 that takes two arguments, (the number of times to print a message and a message to print in morse code) and displays it on an LED”



Here's the complete Python script:

```python
import RPi.GPIO as GPIO
import time
import sys

# Morse code dictionary
MORSE_CODE_DICT = {
    'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.', 'F': '..-.',
    'G': '--.', 'H': '....', 'I': '..', 'J': '.---', 'K': '-.-', 'L': '.-..',
    'M': '--', 'N': '-.', 'O': '---', 'P': '.--.', 'Q': '--.-', 'R': '.-.',
    'S': '...', 'T': '-', 'U': '..-', 'V': '...-', 'W': '.--', 'X': '-..-',
    'Y': '-.--', 'Z': '--..', '0': '-----', '1': '.----', '2': '..---',
    '3': '...--', '4': '....-', '5': '.....', '6': '-....', '7': '--...',
    '8': '---..', '9': '----.'
}

# Constants
DOT_DURATION = 0.2  # Duration of a dot in seconds
DASH_DURATION = 0.6  # Duration of a dash in seconds
SYMBOL_SPACE = 0.2   # Space between symbols in a character
CHAR_SPACE = 0.6     # Space between characters
WORD_SPACE = 1.4     # Space between words

# GPIO setup
LED_PIN = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)

def dot():
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(DOT_DURATION)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(SYMBOL_SPACE)

def dash():
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(DASH_DURATION)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(SYMBOL_SPACE)

def flash_message(message):
    for char in message.upper():
        if char == ' ':
            time.sleep(WORD_SPACE)
        elif char in MORSE_CODE_DICT:
            for symbol in MORSE_CODE_DICT[char]:
                if symbol == '.':
                    dot()
                elif symbol == '-':
                    dash()
            time.sleep(CHAR_SPACE)

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: sudo python3 morse_code_led.py <repetitions> <message>")
        sys.exit(1)

    repetitions = int(sys.argv[1])
    message = sys.argv[2]

    try:
        for _ in range(repetitions):
            flash_message(message)
            time.sleep(WORD_SPACE * 3)  # Space between repetitions
    except KeyboardInterrupt:
        pass
    finally:
        GPIO.cleanup()
```

Lab 4 Part 2:

Part 2 I just used the lecture and copy-pasted the morse code list dictionary from part 1 with some find replaces commands to make it work in C.

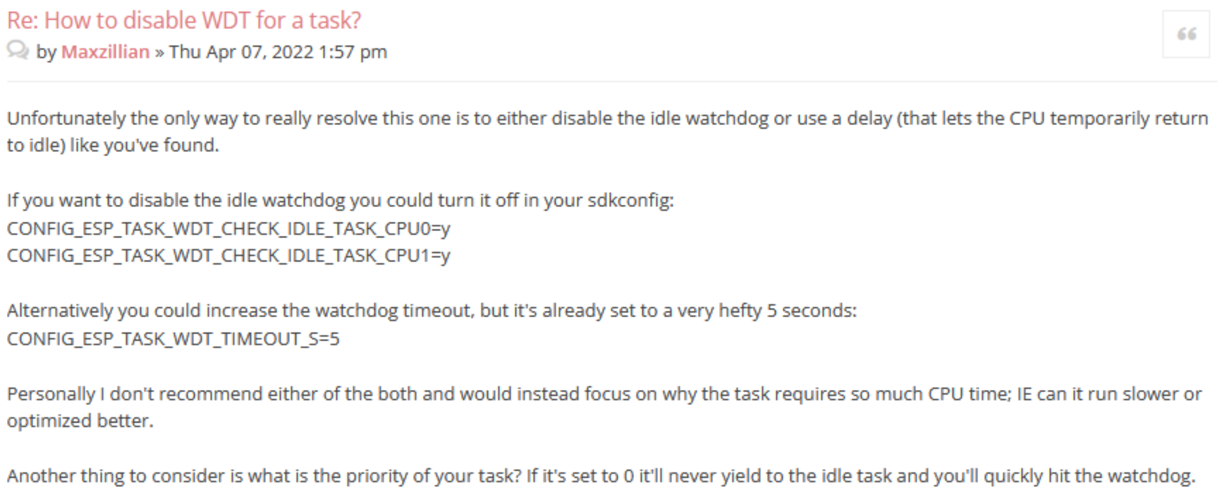
Lab 4 Part 3:

Part 3 I used my part 2 code and asked ChatGPT for help disabling the watchdog so I could run the adc check loop as fast as possible.

https://chatgpt.com/share/2281cb77-172d-428e-8389-ef2b7267a246

None of the code it gave worked so I didn't use any. I ended up using this forum online:

https://esp32.com/viewtopic.php?t=27170



Re: How to disable WDT for a task?
by Maxzillian » Thu Apr 07, 2022 1:57 pm

Unfortunately the only way to really resolve this one is to either disable the idle watchdog or use a delay (that lets the CPU temporarily return to idle) like you've found.

If you want to disable the idle watchdog you could turn it off in your sdkconfig:

```
CONFIG_ESP_TASK_WDT_CHECK_IDLE_TASK_CPU0=y
CONFIG_ESP_TASK_WDT_CHECK_IDLE_TASK_CPU1=y
```

Alternatively you could increase the watchdog timeout, but it's already set to a very hefty 5 seconds:

```
CONFIG_ESP_TASK_WDT_TIMEOUT_S=5
```

Personally I don't recommend either of the both and would instead focus on why the task requires so much CPU time; IE can it run slower or optimized better.

Another thing to consider is what is the priority of your task? If it's set to 0 it'll never yield to the idle task and you'll quickly hit the watchdog.

Part 3 Report:

Maximum transmission speed: ~800 characters per second

Aka: 11,111 dots per second, 3,703 dashes per second.

Testing phrase: “"we all make choices but in the end our choices make us”

Pass/Fail conditions. It works 100% of the time with a dot set to 0.1 milliseconds. With a dot at 0.09 milliseconds, its around 95%. At 0.08 milliseconds it only gets about 75% of characters correct.

Fail time: 65676 microseconds (822 characters per second)

Pass time: 67832 microseconds (796 characters per second)

(Max-min)/min = (822-796)/796 = 3%