EE 193 Homework 2

First steps

Set up your development board to blink an LED at 1Hz.

For the ESP32x chips using Espressif IDF (IoT development framework), you can use the following code:

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h" // Contains thread delay function
#include "driver/gpio.h" // GPIO pin controls

#define BLINK_GPIO 33 // Change this to whatever GPIO pin you're using

void app_main() {
    gpio_reset_pin(BLINK_GPIO);
    gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);

    while(1){
        gpio_set_level(BLINK_GPIO, 1);
        vTaskDelay(500 / portTICK_PERIOD_MS);
        gpio_set_level(BLINK_GPIO, 0);
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}
```

Timing the loop

Remove the timing delays from the code, so that the "blink" code runs as fast as possible. Use an oscilloscope or logic analyzer to measure the frequency at which the pin is being toggled. (If you've not used one before, I'm happy to show you!) How long does it take to set the GPIO pin?

Running faster

We saw in class that function calls often get between us and maximum performance. Replace the `gpio_set_level()` calls with direct writes to the appropriate registers in your chip.

- How fast can you set a GPIO pin now? Is this surprising?
- The ESP32x processor core(s) run at hundreds of MHz (240MHz for the ESP32). Why can't you achieve toggling at this frequency?

Note: The ESP32 has special "set" and "clear" registers, so you should be able to toggle an individual GPIO pin without having to read/write the whole register.

ADC

Write code to sample an analog voltage on one of the pins. For the ESP32x chips, you should look at the examples here: https://github.com/espressif/esp-idf/tree/master/examples/peripherals/adc You'll only need a handful of key function calls from the code, so your code should be much shorter than the example.

- How long does it take to read a single sample? Describe your process for measuring this.
- Does the read time depend on the resolution of the sample (number of bits in the final result)? By how much?
- How fast can you read from the ADC if you read it continuously? The ESP32x chips have a DMA (direct memory access) mode where the ADC can continuously write its results into memory without CPU intervention, which is much faster than capturing individual samples.