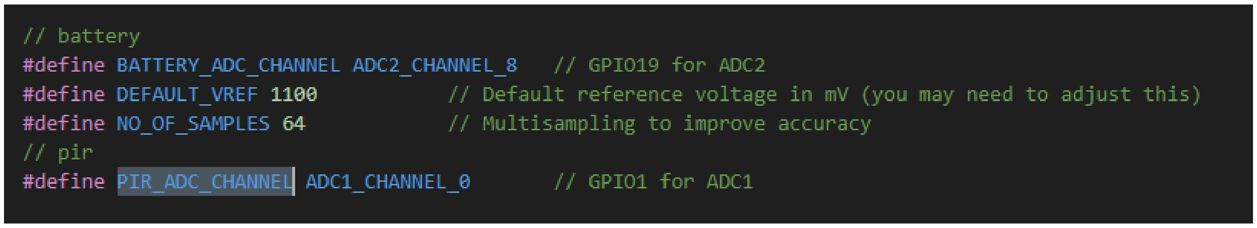
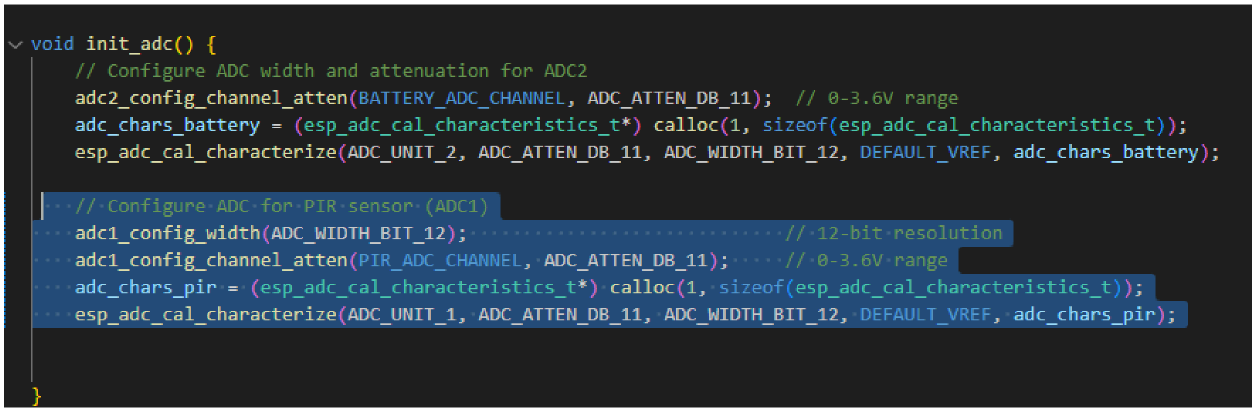
Sozib Al Mamun

Embedded Software Engineer

ADC Int:



```
// battery
#define BATTERY_ADC_CHANNEL ADC2_CHANNEL_8   // GPIO19 for ADC2
#define DEFAULT_VREF 1100           // Default reference voltage in mV (you may need to adjust this)
#define NO_OF_SAMPLES 64            // Multisampling to improve accuracy
// pir
#define PIR_ADC_CHANNEL ADC1_CHANNEL_0       // GPIO1 for ADC1
```

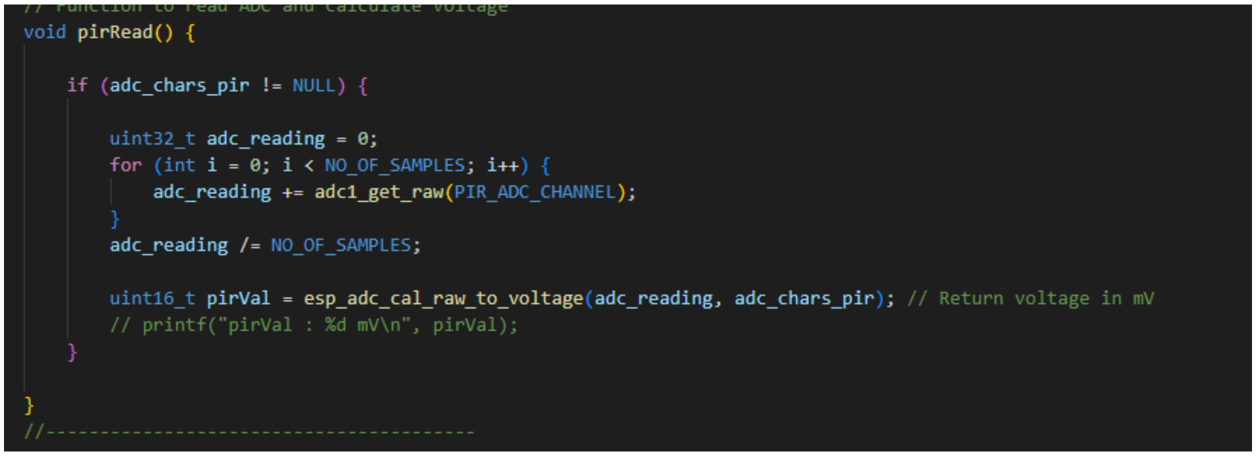


```
void init_adc() {
    // Configure ADC width and attenuation for ADC2
    adc2_config_channel_atten(BATTERY_ADC_CHANNEL, ADC_ATTEN_DB_11);  // 0-3.6V range
    adc_chars_battery = (esp_adc_cal_characteristics_t*) calloc(1, sizeof(esp_adc_cal_characteristics_t));
    esp_adc_cal_characterize(ADC_UNIT_2, ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12, DEFAULT_VREF, adc_chars_battery);

    // Configure ADC for PIR sensor (ADC1)
    adc1_config_width(ADC_WIDTH_BIT_12);                             // 12-bit resolution
    adc1_config_channel_atten(PIR_ADC_CHANNEL, ADC_ATTEN_DB_11);     // 0-3.6V range
    adc_chars_pir = (esp_adc_cal_characteristics_t*) calloc(1, sizeof(esp_adc_cal_characteristics_t));
    esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12, DEFAULT_VREF, adc_chars_pir);


}
```

Get PIR ADC:



```
// Function to read ADC and calculate voltage
void pirRead() {

    if (adc_chars_pir != NULL) {

        uint32_t adc_reading = 0;
        for (int i = 0; i < NO_OF_SAMPLES; i++) {
            adc_reading += adc1_get_raw(PIR_ADC_CHANNEL);
        }
        adc_reading /= NO_OF_SAMPLES;

        uint16_t pirVal = esp_adc_cal_raw_to_voltage(adc_reading, adc_chars_pir); // Return voltage in mV
        // printf("pirVal : %d mV\n", pirVal);
    }

}
//------------------------------------------
```