Smartwatch

Embedded software project

Pedrolli Daniele
Scarano Davide
Valentini Cristian

FRI, JULY 4TH 2025

Team members

Pedrolli Daniele

Main FSM
Buttons + joystick

Scarano Davide

Temperature + I2C
Accelerometer

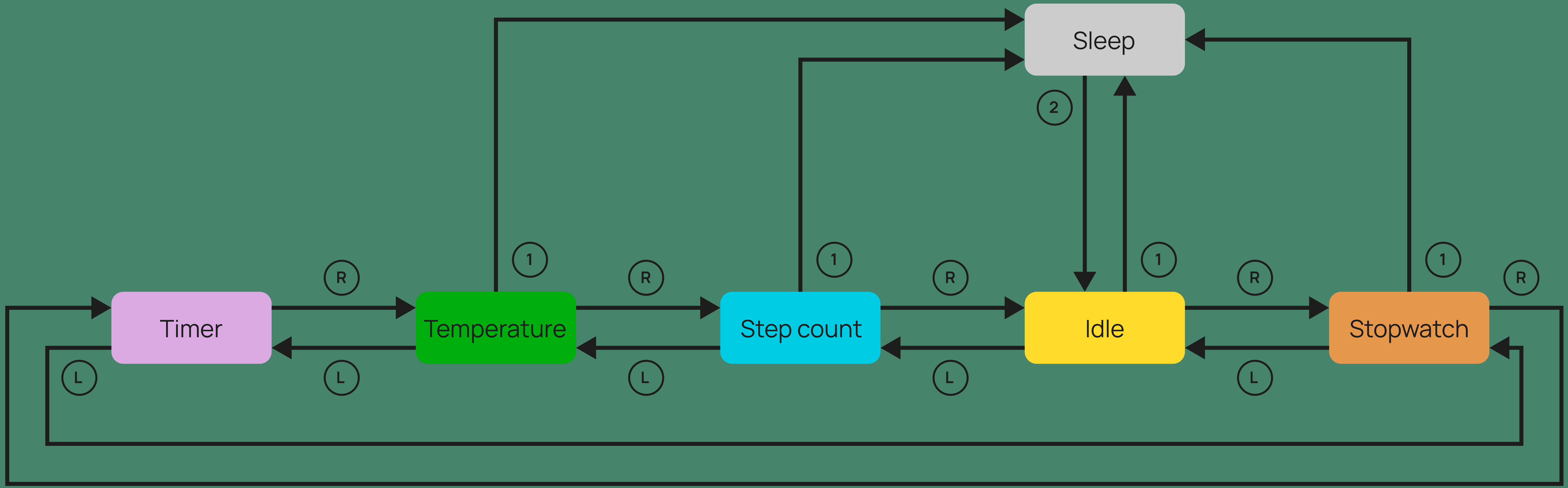
Valentini Cristian

Timer FSM impl.
Stopwatch

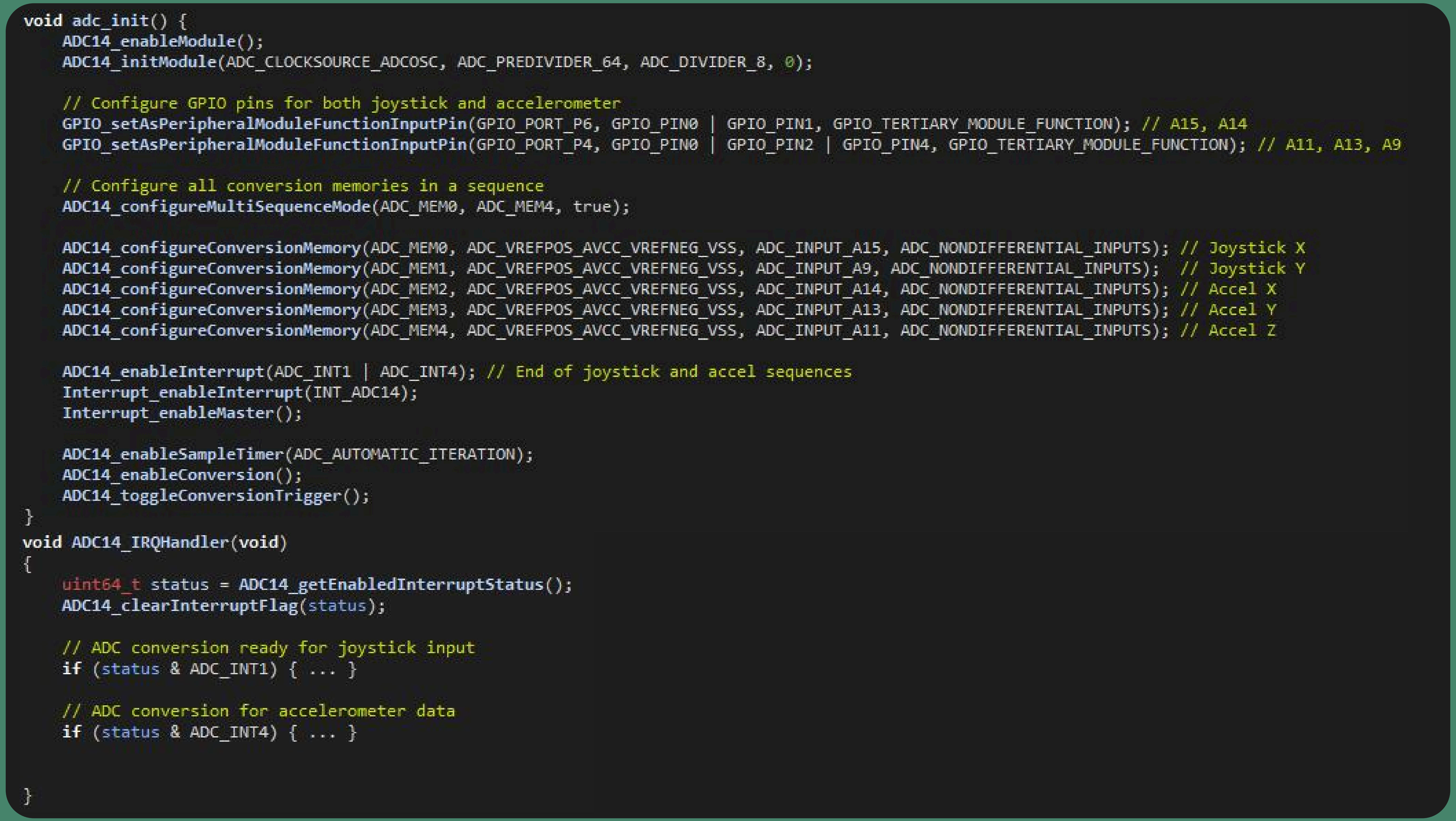
Features

- Step counting
- Temperature reading
- Wall clock
- Timer
- Stopwatch

Control flow



Representative codes

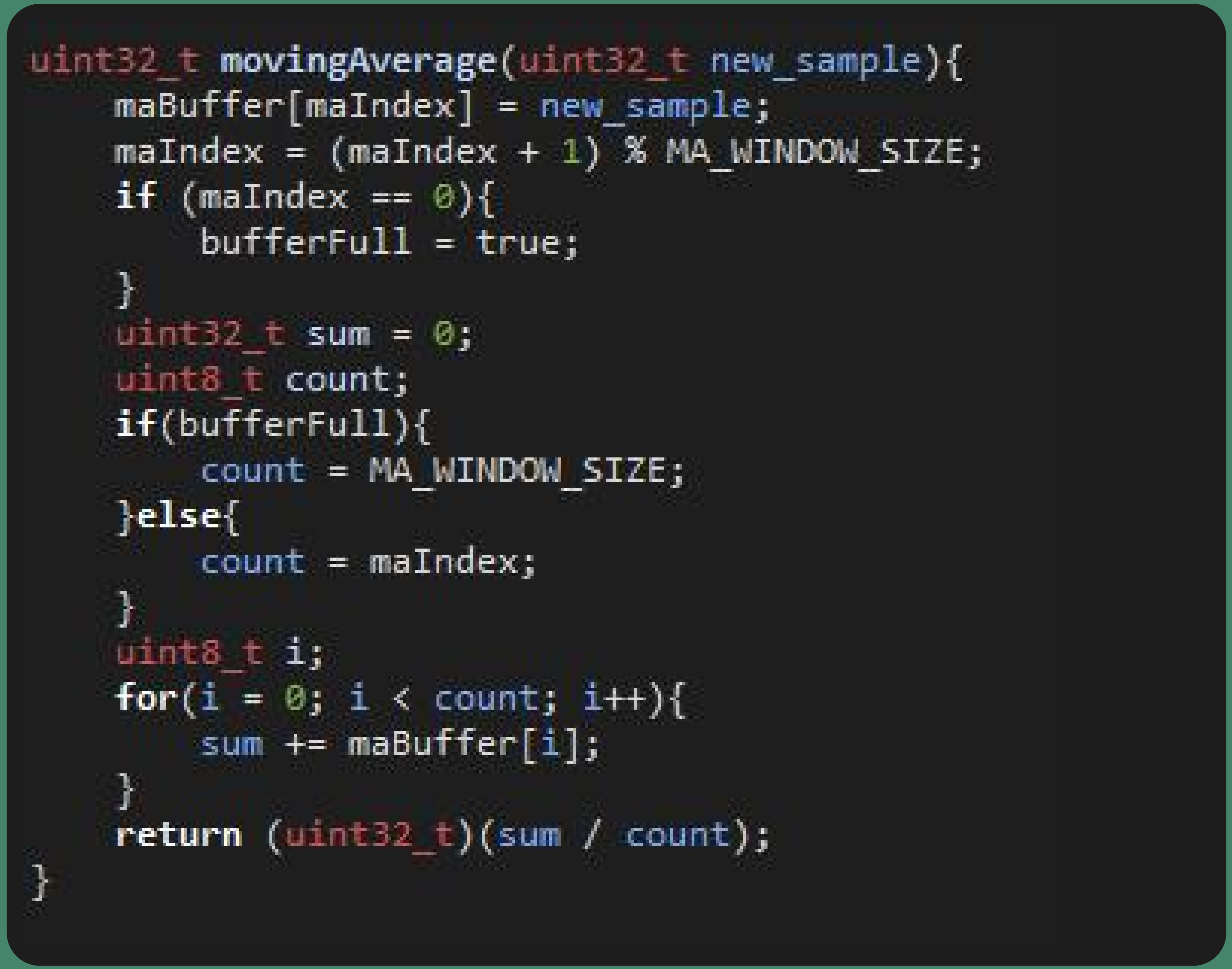


```
void adc_init() {
    ADC14_enableModule();
    ADC14_initModule(ADC_CLOCKSOURCE_ADCOSC, ADC_PREDIVIDER_64, ADC_DIVIDER_8, 0);

    // Configure GPIO pins for both joystick and accelerometer
    GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P6, GPIO_PIN0 | GPIO_PIN1, GPIO_TERTIARY_MODULE_FUNCTION); // A15, A14
    GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P4, GPIO_PIN0 | GPIO_PIN2 | GPIO_PIN4, GPIO_TERTIARY_MODULE_FUNCTION); // A11, A13, A9

    // Configure all conversion memories in a sequence
    ADC14_configureMultiSequenceMode(ADC_MEM0, ADC_MEM4, true);

    ADC14_configureConversionMemory(ADC_MEM0, ADC_VREFPOS_AVCC_VREFNEG_VSS, ADC_INPUT_A15, ADC_NONDIFFERENTIAL_INPUTS); // Joystick X
    ADC14_configureConversionMemory(ADC_MEM1, ADC_VREFPOS_AVCC_VREFNEG_VSS, ADC_INPUT_A9, ADC_NONDIFFERENTIAL_INPUTS);  // Joystick Y
    ADC14_configureConversionMemory(ADC_MEM2, ADC_VREFPOS_AVCC_VREFNEG_VSS, ADC_INPUT_A14, ADC_NONDIFFERENTIAL_INPUTS); // Accel X
    ADC14_configureConversionMemory(ADC_MEM3, ADC_VREFPOS_AVCC_VREFNEG_VSS, ADC_INPUT_A13, ADC_NONDIFFERENTIAL_INPUTS); // Accel Y
    ADC14_configureConversionMemory(ADC_MEM4, ADC_VREFPOS_AVCC_VREFNEG_VSS, ADC_INPUT_A11, ADC_NONDIFFERENTIAL_INPUTS); // Accel Z

    ADC14_enableInterrupt(ADC_INT1 | ADC_INT4); // End of joystick and accel sequences
    Interrupt_enableInterrupt(INT_ADC14);
    Interrupt_enableMaster();

    ADC14_enableSampleTimer(ADC_AUTOMATIC_ITERATION);
    ADC14_enableConversion();
    ADC14_toggleConversionTrigger();
}
void ADC14_IRQHandler(void)
{
    uint64_t status = ADC14_getEnabledInterruptStatus();
    ADC14_clearInterruptFlag(status);

    // ADC conversion ready for joystick input
    if (status & ADC_INT1) { ... }

    // ADC conversion for accelerometer data
    if (status & ADC_INT4) { ... }


}
```

ADC conversion

We use ADC14 to handle accelerometer and joystick data. They use different channels and are both interrupt-driven. In the interrupt handler the channel is checked to pass only the correct data to the main program.

Representative codes



```
uint32_t movingAverage(uint32_t new_sample){
    maBuffer[maIndex] = new_sample;
    maIndex = (maIndex + 1) % MA_WINDOW_SIZE;
    if (maIndex == 0){
        bufferFull = true;
    }
    uint32_t sum = 0;
    uint8_t count;
    if(bufferFull){
        count = MA_WINDOW_SIZE;
    }else{
        count = maIndex;
    }
    uint8_t i;
    for(i = 0; i < count; i++){
        sum += maBuffer[i];
    }
    return (uint32_t)(sum / count);
}
```

Moving average

To more accurately count steps, we apply a 100-sample moving average, so that outliers in accelerometer data aren't detected as steps. Considering peripherals work on a 48MHz clock, 100 samples should never mean we lose a step.

Issues

The LaunchPad does not have on-board RTC, so the board would need to be plugged into a computer to have an accurate timestamp, but once on external power it would be inaccurate again.