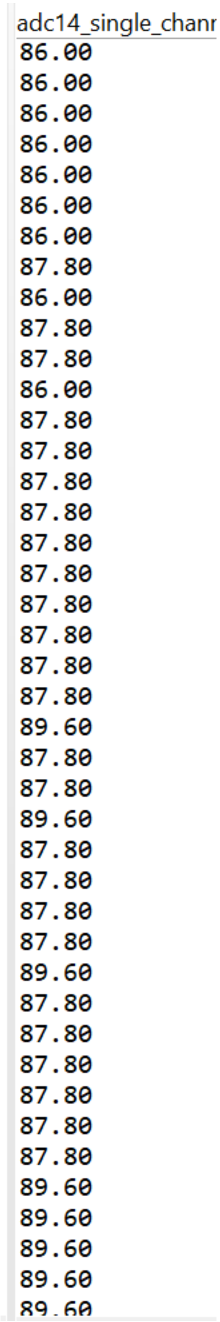
Rajbir Kahlon

Lab 6 Assignment

Exercise 1

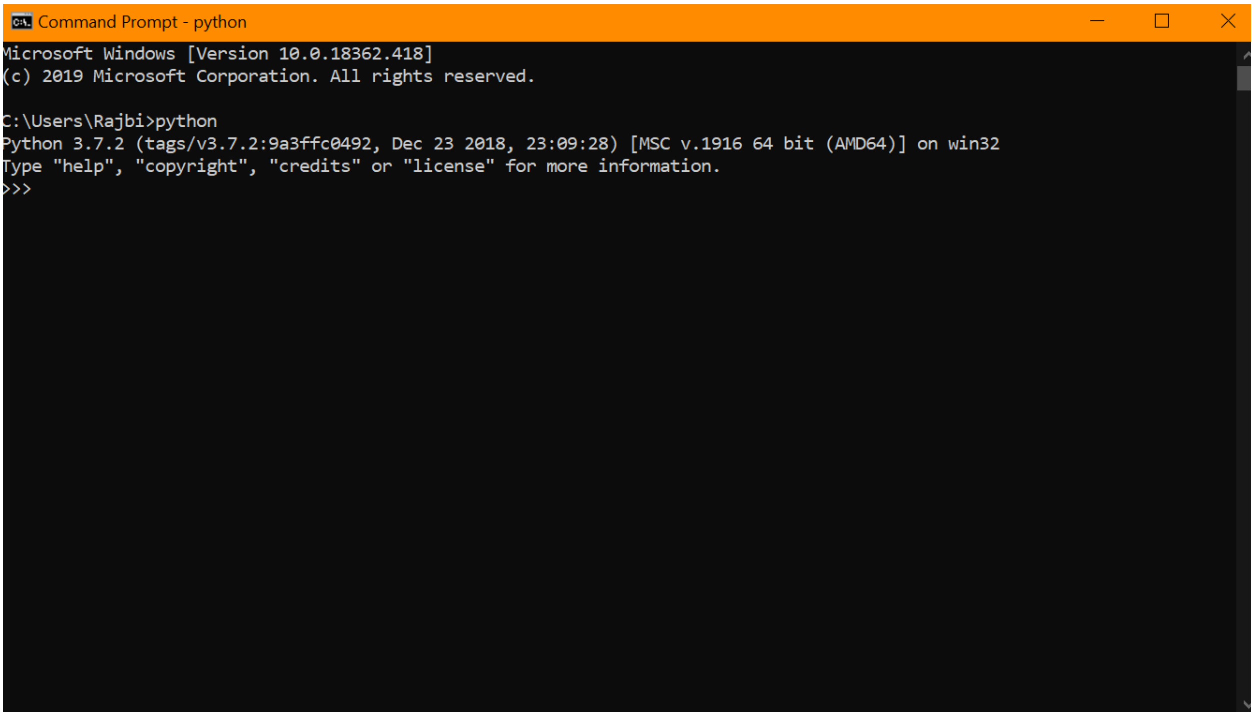
The adc14_single_channel_temperature_sensor_MSP_EXP432P401R_nortos_ccs example project provided by TI was used as a baseline. The calDifference, tempC, and tempF global floater variables were changed to int variables and the equation to calculate tempF was modified by removing the f's and multiplying the whole thing by 100. This was done to print the int numbers as a number with 2 decimal places as requested. The delay_ms function was brought over from my Lab4. delay_ms(1000) was used to add a one second delay between each printing of temperature. The average room temperature was about 86.0 degrees Fahrenheit, when the MCU was touched, it became 87.80 degrees Fahrenheit and if continued to be touched went up to 89.6 degrees Fahrenheit as seen in screenshot below. Code is in appendix.



```
adc14_single_chanr
86.00
86.00
86.00
86.00
86.00
86.00
86.00
87.80
86.00
87.80
87.80
86.00
87.80
87.80
87.80
87.80
87.80
87.80
87.80
87.80
87.80
87.80
89.60
87.80
87.80
89.60
87.80
87.80
87.80
87.80
89.60
87.80
87.80
87.80
87.80
87.80
87.80
89.60
89.60
89.60
89.60
89.60
```

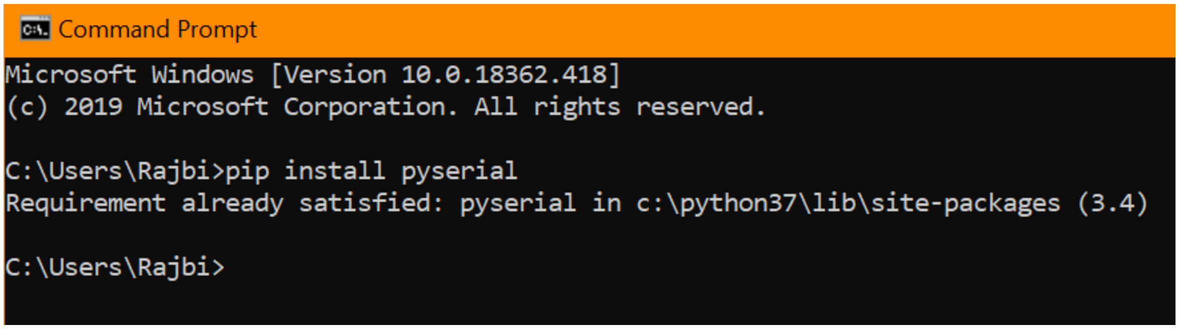
Exercise 2.1

Python was successfully installed as well as pyserial.



```
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Rajbi>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



```
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Rajbi>pip install pyserial
Requirement already satisfied: pyserial in c:\python37\lib\site-packages (3.4)

C:\Users\Rajbi>
```

Exercise 2.2

Not able to send to UART correctly

Appendix

Exercise 1

```
/* --COPYRIGHT--,BSD
 * Copyright (c) 2017, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/
/*******************************************************************************
 * MSP432 ADC14 - Single Channel Repeat Temperature Sensor
 *
 * Description: This example shows the use of the internal temperature sensor.
 * A simple continuous ADC sample/conversion is set up with a software trigger.
 * The sample time is set to TBD as speced by the User's Guide. All calculations
 * take place in the ISR which take advantage of the Stacking Mode of the FPU.
 * The temperature is calculated in both Celsius and Fahrenheit.
 *
 *                MSP432P401
 *             ------------------
 *         /|\|                  |
 *          | |                  |
 *          --|RST         P5.5  |
 *            |                  |
 *            |                  |
 *            |                  |
 *
 ******************************************************************************/
/* DriverLib Includes */
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

/* Standard Includes */
#include <stdint.h>
#include <string.h>
#include <stdbool.h>
#include <stdlib.h>
#include <stdio.h>

uint32_t cal30;
uint32_t cal85;
int calDifference; //changed from floater to int
int tempC; //changed from floater to int
int tempF; //changed from floater to int

void delay_ms(uint32_t count); //function created in lab4
bool tick;
```

```c
volatile uint32_t val;

int main(void)
{
    /* Halting WDT  */
    WDT_A_holdTimer();
    Interrupt_enableSleepOnIsrExit();

    /* Enabling the FPU with stacking enabled (for use within ISR) */
    FPU_enableModule();
    FPU_enableLazyStacking();

    /* Initializing ADC (MCLK/1/1) with temperature sensor routed */
    ADC14_enableModule();
    ADC14_initModule(ADC_CLOCKSOURCE_MCLK, ADC_PREDIVIDER_1, ADC_DIVIDER_1,
            ADC_TEMPSENSEMAP);

    /* Configuring ADC Memory (ADC_MEM0 A22 (Temperature Sensor) in repeat
     * mode).
     */
    ADC14_configureSingleSampleMode(ADC_MEM0, true);
    ADC14_configureConversionMemory(ADC_MEM0, ADC_VREFPOS_INTBUF_VREFNEG_VSS,
            ADC_INPUT_A22, false);

    /* Configuring the sample/hold time for 192 */
    ADC14_setSampleHoldTime(ADC_PULSE_WIDTH_192,ADC_PULSE_WIDTH_192);

    /* Enabling sample timer in auto iteration mode and interrupts*/
    ADC14_enableSampleTimer(ADC_AUTOMATIC_ITERATION);
    ADC14_enableInterrupt(ADC_INT0);

    /* Setting reference voltage to 2.5 and enabling temperature sensor */
    REF_A_enableTempSensor();
    REF_A_setReferenceVoltage(REF_A_VREF2_5V);
    REF_A_enableReferenceVoltage();

    cal30 = SysCtl_getTempCalibrationConstant(SYSCTL_2_5V_REF,
            SYSCTL_30_DEGREES_C);
    cal85 = SysCtl_getTempCalibrationConstant(SYSCTL_2_5V_REF,
            SYSCTL_85_DEGREES_C);
    calDifference = cal85 - cal30;

    /* Enabling Interrupts */
    Interrupt_enableInterrupt(INT_ADC14);
    Interrupt_enableMaster();

    /* Triggering the start of the sample */
    ADC14_enableConversion();
    ADC14_toggleConversionTrigger();

    /* Going to sleep */
    while (1)
    {
        delay_ms(1000); //delay 1 second
        PCM_gotoLPM0();
    }
}

/* This interrupt happens every time a conversion has completed. Since the FPU
 * is enabled in stacking mode, we are able to use the FPU safely to perform
 * efficient floating point arithmetic.*/
void ADC14_IRQHandler(void)
{
    uint64_t status;
    int16_t conRes;

    status = ADC14_getEnabledInterruptStatus();
    ADC14_clearInterruptFlag(status);

    if(status & ADC_INT0)
    {
        conRes = ((ADC14_getResult(ADC_MEM0) - cal30) * 55);
        tempC = (conRes / calDifference) + 30.0;
        tempF = (tempC * 9.0 / 5.0 + 32.0)*100; //got rid of floating point
        printf("%d.%02d\n", tempF/100, tempF%100); //print value in Farenheit
    }
}

void delay_ms(uint32_t count) {
    Timer32_startTimer(TIMER32_0_BASE,true);
    Timer32_setCount(TIMER32_BASE,UINT32_MAX);
    tick = true;
    while(tick) {
        val = UINT32_MAX-Timer32_getValue(TIMER32_BASE);
        if(val >= (count*3000))
            tick = false;
    }
}
```