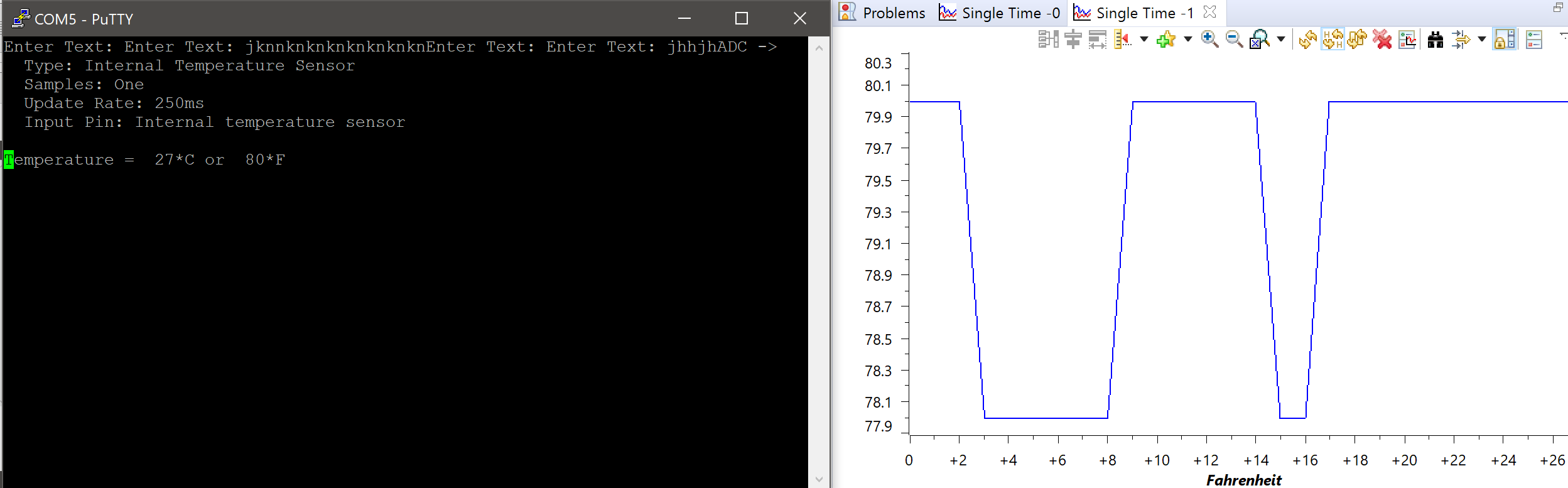
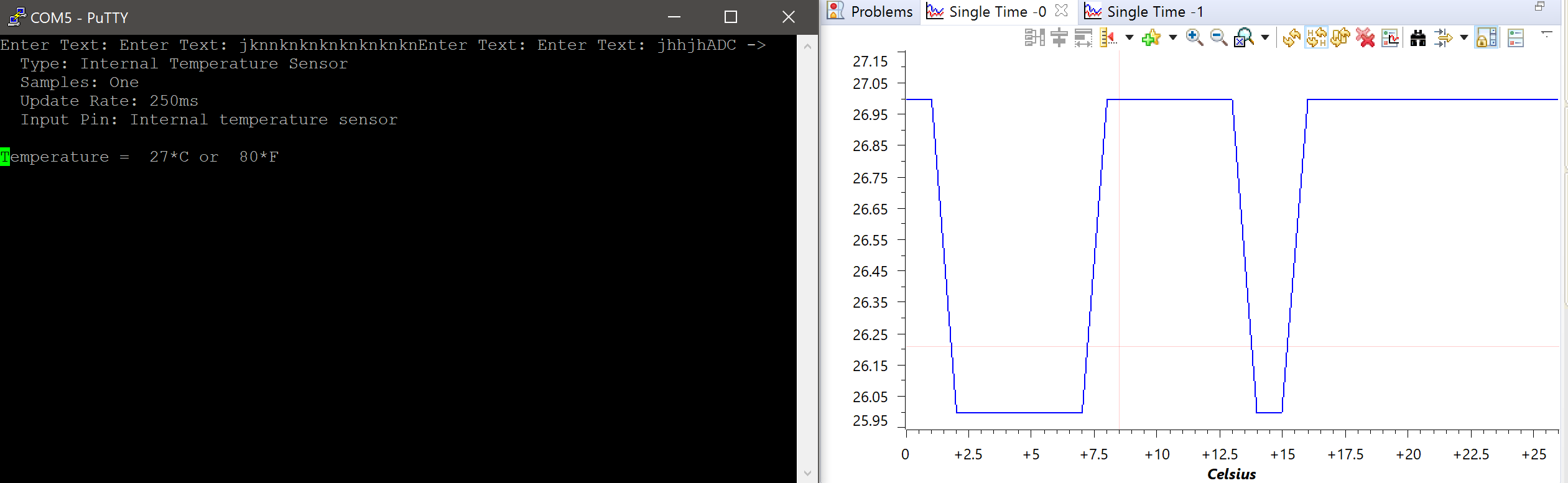
**Date Submitted: 10/11/2018 4:17 PM**

**Task 00: Execute provided code, display the temperatures in the built in Graph Tool.**



**The built-in graph measuring Fahrenheit**



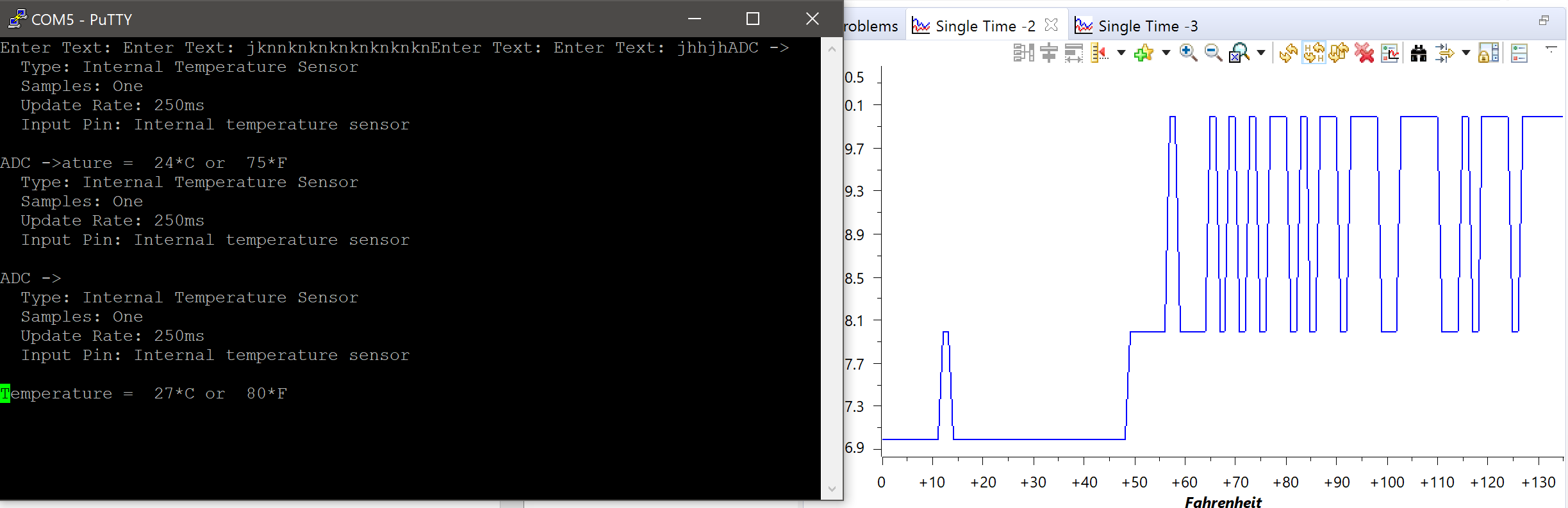
**The built-in graph measuring Celsius**

**Youtube Link: No Submission required for Task 00**

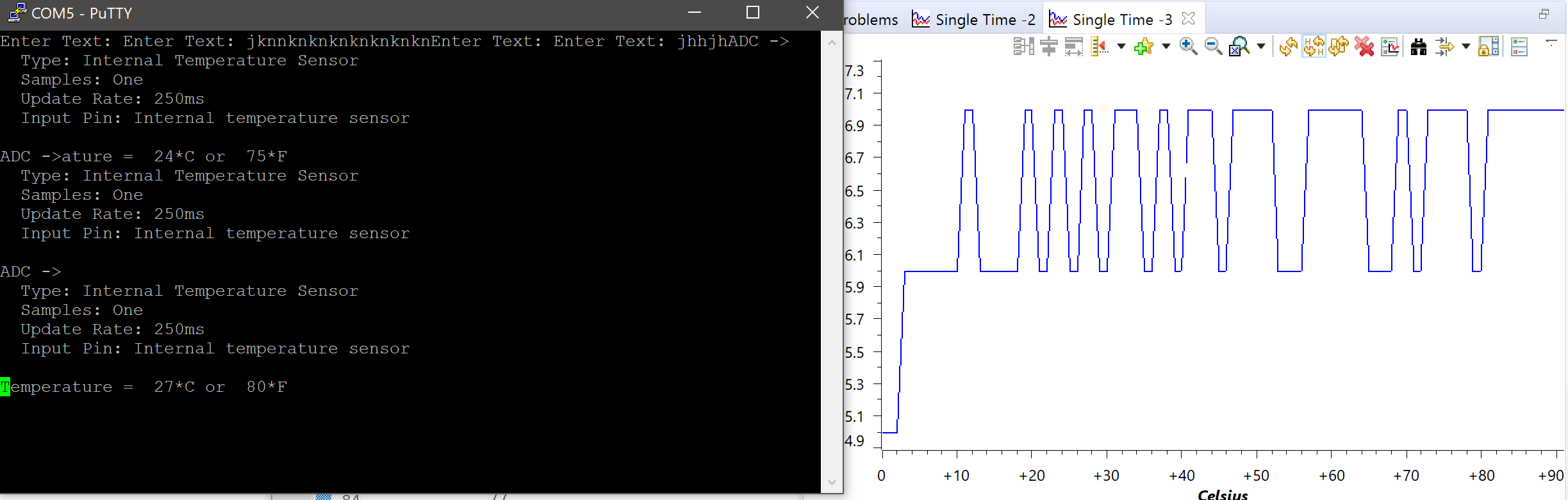
**------------------------------------------------------------------------------------**

**Task 01:** Continuously display the temperature of the device (internal temperature sensor) on the a) hyperterminal, and b) GUI Composer using a timer interrupt every 0.5s

Youtube Link: <https://www.youtube.com/watch?v=SCL_KCv4JKs>



Above is the GUI Composer and the hyperterminal (PuTTy) for Fahrenheit and below is for Celsius. It is responsible for refreshing using a timer interrupt every 0.5s.



**Modified Code:**

**#define** PART\_TM4C123GH6PM

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "stdlib.h"

**#include** "inc/hw\_ints.h"

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_uart.h"

**#include** "inc/hw\_gpio.h"

**#include** "inc/hw\_pwm.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/adc.h"

**#include** "driverlib/timer.h"

**#include** "driverlib/gpio.h"

**#include** "driverlib/interrupt.h"

**#include** "driverlib/pin\_map.h"

**#include** "driverlib/rom.h"

**#include** "driverlib/rom\_map.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/uart.h"

**#include** "driverlib/udma.h"

**#include** "driverlib/pwm.h"

**#include** "driverlib/ssi.h"

**#include** "driverlib/systick.h"

**#include** "driverlib/adc.h"

**#include** "utils/uartstdio.h"

**#include** "utils/uartstdio.c"

**#include** <string.h>

**#include** "inc/tm4c123gh6pm.h"

uint32\_t ui32Period;

uint32\_t ADCValues[1];

uint32\_t TempValueC ;

uint32\_t TempValueF ;

**void** **Timer1IntHandler**(**void**)

{

**int** clock;

clock = **SysCtlClockGet**();

SYSCTL\_RCGCTIMER\_R |= 2; //enable clock

TIMER1\_CTL\_R = 0; //disable Timer before initialization

TIMER1\_CFG\_R = 0x04; //16-bit option

TIMER1\_TAMR\_R = 0x02; //periodic mode and down-counter

TIMER1\_TAILR\_R = 80000-1; //TimerA interval load value reg

TIMER1\_TAPR\_R = 250 - 1; //TimerA prescaler 40MHz/250 = 16000Hz => 1/16000 = .0625ms => .5s/.0625ms = 80000

TIMER1\_ICR\_R = 0x01; //clear the TimerA timeout flag

TIMER1\_CTL\_R |= 0x01; //enable Timer A after initialization

// GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4) ;

**while**((TIMER1\_RIS\_R & 0x1) == 0)

; //turn off red LED

**ADCProcessorTrigger**(ADC0\_BASE, 3);

//

// Wait for conversion to be completed.

//

**while**(!**ADCIntStatus**(ADC0\_BASE, 3, false))

{

}

//

// Clear the ADC interrupt flag.

//

**ADCIntClear**(ADC0\_BASE, 3);

//

// Read ADC Value.

//

**ADCSequenceDataGet**(ADC0\_BASE, 3, ADCValues);

//

// Use non-calibrated conversion provided in the data sheet. I use floats in intermediate

// math but you could use intergers with multiplied by powers of 10 and divide on the end

// Make sure you divide last to avoid dropout.

//

TempValueC = (uint32\_t)(147.5 - ((75.0\*3.3 \*(**float**)ADCValues[0])) / 4096.0);

//

// Get Fahrenheit value. Make sure you divide last to avoid dropout.

//

TempValueF = ((TempValueC \* 9) + 160) / 5;

//

// Display the temperature value on the console.

//

**UARTprintf**("Temperature = %3d\*C or %3d\*F\r", TempValueC,

TempValueF);

//

// This function provides a means of generating a constant length

// delay. The function delay (in cycles) = 3 \* parameter. Delay

// 250ms arbitrarily.

//

**SysCtlDelay**(80000000 / 12);

TIMER1\_ICR\_R = 0x1; //clear flag

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// This function sets up UART0 to be used for a console to display information

// as the example is running.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void**

**InitConsole**(**void**)

{

//

// Enable GPIO port A which is used for UART0 pins.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA);

//

// Configure the pin muxing for UART0 functions on port A0 and A1.

**GPIOPinConfigure**(GPIO\_PA0\_U0RX);

**GPIOPinConfigure**(GPIO\_PA1\_U0TX);

// Enable UART0 so that we can configure the clock.

//

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0);

//

// Use the internal 16MHz oscillator as the UART clock source.

//

**UARTClockSourceSet**(UART0\_BASE, UART\_CLOCK\_PIOSC);

//

// Select the alternate (UART) function for these pins.

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

//

// Initialize the UART for console I/O.

//

**UARTStdioConfig**(0, 115200, 16000000);

}

**int** **main**(){

**SysCtlClockSet**(SYSCTL\_SYSDIV\_2\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

InitConsole();

//

// This array is used for storing the data read from the ADC FIFO. It

// must be as large as the FIFO for the sequencer in use. This example

// uses sequence 3 which has a FIFO depth of 1. If another sequence

// was used with a deeper FIFO, then the array size must be changed.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1);

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

ui32Period = (**SysCtlClockGet**() / 10) / 2;

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, ui32Period -1);

**IntEnable**(INT\_TIMER1A);

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

**IntMasterEnable**();

**TimerEnable**(TIMER1\_BASE, TIMER\_A);

//

// Display the setup on the console.

//

**UARTprintf**("ADC ->\n");

**UARTprintf**(" Type: Internal Temperature Sensor\n");

**UARTprintf**(" Samples: One\n");

**UARTprintf**(" Update Rate: 250ms\n");

**UARTprintf**(" Input Pin: Internal temperature sensor\n\n");

//

// The ADC0 peripheral must be enabled for use.

//

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0);

**SysCtlDelay**(3);

**ADCSequenceConfigure**(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0);

**ADCSequenceStepConfigure**(ADC0\_BASE, 3, 0, ADC\_CTL\_TS | ADC\_CTL\_IE |

ADC\_CTL\_END);

//

// Since sample sequence 3 is now configured, it must be enabled.

//

**ADCSequenceEnable**(ADC0\_BASE, 3);

// Clear the interrupt status flag.

//

**ADCIntClear**(ADC0\_BASE, 3);

// Sample the temperature sensor forever.

**while**(1)

{

//

// Trigger the ADC conversion.

//

}

}

**------------------------------------------------------------------------------------**

**Task 02:** Interaction/User interface: Develop a user interface using UART

Youtube Link: <https://www.youtube.com/watch?v=9wOHZZ9w924>

This program is responsible for creating a user interface for the components within the Tiva-C board. For example, when the ‘R’ button is pressed the RED LED will turn on, and when the ‘r’ button is pressed the RED LED will turn off.

**Modified Code:**

**#define** PART\_TM4C123GH6PM

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "stdlib.h"

**#include** "inc/hw\_ints.h"

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_uart.h"

**#include** "inc/hw\_gpio.h"

**#include** "inc/hw\_pwm.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/adc.h"

**#include** "driverlib/timer.h"

**#include** "driverlib/gpio.h"

**#include** "driverlib/interrupt.h"

**#include** "driverlib/pin\_map.h"

**#include** "driverlib/rom.h"

**#include** "driverlib/rom\_map.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/uart.h"

**#include** "driverlib/udma.h"

**#include** "driverlib/pwm.h"

**#include** "driverlib/ssi.h"

**#include** "driverlib/systick.h"

**#include** "driverlib/adc.h"

**#include** "utils/uartstdio.h"

**#include** "utils/uartstdio.c"

**#include** <string.h>

**#include** "inc/tm4c123gh6pm.h"

**void**

**InitConsole**(**void**)

{

// Enable GPIO port A which is used for UART0 pins.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA);

// Configure the pin muxing for UART0 functions on port A0 and A1.

**GPIOPinConfigure**(GPIO\_PA0\_U0RX);

**GPIOPinConfigure**(GPIO\_PA1\_U0TX);

// Enable UART0 so that we can configure the clock.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0);

// Use the internal 16MHz oscillator as the UART clock source.

**UARTClockSourceSet**(UART0\_BASE, UART\_CLOCK\_PIOSC);

// Select the alternate (UART) function for these pins.

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

// Initialize the UART for console I/O.

**UARTStdioConfig**(0, 115200, 16000000);

uint32\_t ADCValues[1];

uint32\_t TempValueC ;

uint32\_t TempValueF ;

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0);

**SysCtlDelay**(3);

**ADCSequenceConfigure**(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0);

**ADCSequenceStepConfigure**(ADC0\_BASE, 3, 0, ADC\_CTL\_TS | ADC\_CTL\_IE |

ADC\_CTL\_END);

**ADCSequenceEnable**(ADC0\_BASE, 3);

**ADCIntClear**(ADC0\_BASE, 3);

**while**(1)

{

**ADCProcessorTrigger**(ADC0\_BASE, 3);

**while**(!**ADCIntStatus**(ADC0\_BASE, 3, false))

{

}

**ADCIntClear**(ADC0\_BASE, 3);

**ADCSequenceDataGet**(ADC0\_BASE, 3, ADCValues);

TempValueC = (uint32\_t)(147.5 - ((75.0\*3.3 \*(**float**)ADCValues[0])) / 4096.0);

TempValueF = ((TempValueC \* 9) + 160) / 5;

**UARTprintf**("Temperature = %3d\*C or %3d\*F\n", TempValueC,

TempValueF); //print F and C for temperature

**SysCtlDelay**(800000000 / 12);

}

}

**void** **UARTIntHandler**(**void**)

{

uint32\_t ui32Status;

ui32Status = **UARTIntStatus**(UART0\_BASE, true); //get interrupt status

**UARTIntClear**(UART0\_BASE, ui32Status); //clear the asserted interrupts

**while**(**UARTCharsAvail**(UART0\_BASE)) //loop while there are chars

{

**if**(**UARTCharGet**(UART0\_BASE) == 'R'){

**UARTCharPut**(UART0\_BASE, 'R');

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, GPIO\_PIN\_1); //turn on red LED

}

**if**(**UARTCharGet**(UART0\_BASE) == 'r'){

**UARTCharPut**(UART0\_BASE, 'r');

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0); //turn off red

}

**if**(**UARTCharGet**(UART0\_BASE) == 'G'){

**UARTCharPut**(UART0\_BASE, 'G');

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, GPIO\_PIN\_3); //turn on green LED

}

**if**(**UARTCharGet**(UART0\_BASE) == 'g'){

**UARTCharPut**(UART0\_BASE, 'g'); //turn off green

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, 0);

}

**if**(**UARTCharGet**(UART0\_BASE) == 'B'){

**UARTCharPut**(UART0\_BASE, 'B');

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, GPIO\_PIN\_2); //turn on blue LED

}

**if**(**UARTCharGet**(UART0\_BASE) == 'b'){

**UARTCharPut**(UART0\_BASE, 'b'); //turn off blue

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0);

}

**if**(**UARTCharGet**(UART0\_BASE) == 'T'){

**UARTCharPut**(UART0\_BASE, 'T');

InitConsole(); //call function to run temperature

}

}

}

**int** **main**(**void**) {

**SysCtlClockSet**(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA);

**GPIOPinConfigure**(GPIO\_PA0\_U0RX);

**GPIOPinConfigure**(GPIO\_PA1\_U0TX);

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); //enable GPIO port for LED

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3); //enable pin for LED PF2

**UARTConfigSetExpClk**(UART0\_BASE, **SysCtlClockGet**(), 115200,

(UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE));

**IntMasterEnable**(); //enable processor interrupts

**IntEnable**(INT\_UART0); //enable the UART interrupt

**UARTIntEnable**(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RT); //only enable RX and TX interrupts

**UARTCharPut**(UART0\_BASE, 'E');

**UARTCharPut**(UART0\_BASE, 'n');

**UARTCharPut**(UART0\_BASE, 't');

**UARTCharPut**(UART0\_BASE, 'e');

**UARTCharPut**(UART0\_BASE, 'r');

**UARTCharPut**(UART0\_BASE, ' ');

**UARTCharPut**(UART0\_BASE, 'T');

**UARTCharPut**(UART0\_BASE, 'e');

**UARTCharPut**(UART0\_BASE, 'x');

**UARTCharPut**(UART0\_BASE, 't');

**UARTCharPut**(UART0\_BASE, ':');

**UARTCharPut**(UART0\_BASE, ' ');

**while** (1) //let interrupt handler do the UART echo function

{

// if (UARTCharsAvail(UART0\_BASE)) UARTCharPut(UART0\_BASE, UARTCharGet(UART0\_BASE));

}

}