Jiajian Chen
Github root directory: https://github.com/IMSB007/2019Fall

Date Submitted:

Task 01:

Youtube Link: https://www.youtube.com/watch?v=kIPKyqwq9kY

Modified Schematic (if applicable):

Modified Code:

```
// Insert code here
#ifdef DEBUG
void__error__(char *pcFilename, uint32_t ui32Line)
{
}
#endif
    uint32_t ui32ADC0Value[4];
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;
    char buffer[2];

void Timer1AHandler(void)
{
    TimerIntClear(TIMER1_BASE,TIMER_TIMA_TIMEOUT);
    ADCIntClear(ADC0_BASE, 2);
    ADCProcessorTrigger(ADC0_BASE, 2);
   while(!ADCIntStatus(ADC0_BASE, 2, false))
   {
   }

    ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);
    ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
    ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
    ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
    ltoa(ui32TempValueF, buffer);
    UARTCharPut(UART0_BASE,'T');
    UARTCharPut(UART0_BASE,'e');
    UARTCharPut(UART0_BASE,'m');
    UARTCharPut(UART0_BASE,'p');
    UARTCharPut(UART0_BASE,'e');
    UARTCharPut(UART0_BASE,'r');
    UARTCharPut(UART0_BASE,'a');
    UARTCharPut(UART0_BASE,'t');
    UARTCharPut(UART0_BASE,'u');
    UARTCharPut(UART0_BASE,'r');
    UARTCharPut(UART0_BASE,'e');
    UARTCharPut(UART0_BASE,':');
    UARTCharPut(UART0_BASE, buffer[0]);
    UARTCharPut(UART0_BASE, buffer[1]);
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
    UARTCharPut(UART0_BASE,'F');
    UARTCharPut(UART0_BASE,'\n');
    UARTCharPut(UART0_BASE,'\r');
    SysCtlDelay(20000000);
}

int main(void) {

        SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN |
SYSCTL_XTAL_16MHZ);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCHardwareOversampleConfigure(ADC0_BASE, 32);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);

    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE,2,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 2);

    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    TimerConfigure(TIMER1_BASE,TIMER_CFG_PERIODIC);

    TimerLoadSet(TIMER1_BASE, TIMER_A, (SysCtlClockGet()/2)-1);

    IntEnable(INT_TIMER1A);
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,
        (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
    IntMasterEnable();
    TimerEnable(TIMER1_BASE, TIMER_A);

    ADCIntEnable(ADC0_BASE,2);


    while (1)
    {
    }

}
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Jiajian Chen
Github root directory: https://github.com/IMSB007/2019Fall

Task 02:

Youtube Link: https://www.youtube.com/watch?v=xtV_jm2ti24

Modified Schematic (if applicable):

Modified Code:

```
// Insert code here

#ifdef DEBUG
void__error__(char *pcFilename, uint32_t ui32Line)
{
}
#endif
    uint32_t ui32ADC0Value[4];
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;
    char command;
    char F[2];
    char C[2];

void UARTIntHandler(void)
{
    uint32_t ui32Status;
    ui32Status = UARTIntStatus(UART0_BASE, true);
    UARTIntClear(UART0_BASE, ui32Status);

    while(1)
    {
        ADCIntClear(ADC0_BASE, 2);
        ADCProcessorTrigger(ADC0_BASE, 2);
        command = UARTCharGet(UART0_BASE);
        UARTCharPut(UART0_BASE, command);

        switch(command)
        {
        case 'R':
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,2);
            SysCtlDelay(SysCtlClockGet()/(1000*3));
            break;
        case 'r':
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1,0);
            SysCtlDelay(SysCtlClockGet()/(1000*3));
            break;
        case 'B':
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,4);
            SysCtlDelay(SysCtlClockGet()/(1000*3));
            break;
        case 'b':
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2,0);
```

```
            SysCtlDelay(SysCtlClockGet()/(1000*3));
            break;
        case 'G':
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,8);
            SysCtlDelay(SysCtlClockGet()/(1000*3));
            break;
        case 'g':
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3,0);
            SysCtlDelay(SysCtlClockGet()/(1000*3));
            break;
        case 'T':
            while(!ADCIntStatus(ADC0_BASE, 2, false))
                      {
                      }

            ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);
            ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
            ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
            ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;

            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,0);
            UARTCharPut(UART0_BASE,'\n');
            UARTCharPut(UART0_BASE,'\r');
            ltoa(ui32TempValueC, C);
            UARTCharPut(UART0_BASE,'T');
            UARTCharPut(UART0_BASE,'e');
            UARTCharPut(UART0_BASE,'m');
            UARTCharPut(UART0_BASE,'p');
            UARTCharPut(UART0_BASE,'e');
            UARTCharPut(UART0_BASE,'r');
            UARTCharPut(UART0_BASE,'a');
            UARTCharPut(UART0_BASE,'t');
            UARTCharPut(UART0_BASE,'u');
            UARTCharPut(UART0_BASE,'r');
            UARTCharPut(UART0_BASE,'e');
            UARTCharPut(UART0_BASE,':');
            UARTCharPut(UART0_BASE, C[0]);
            UARTCharPut(UART0_BASE, C[1]);
            UARTCharPut(UART0_BASE, 'C');
            UARTCharPut(UART0_BASE, ' ');
            UARTCharPut(UART0_BASE, '=');
            UARTCharPut(UART0_BASE, ' ');
            ltoa(ui32TempValueF,F);
            UARTCharPut(UART0_BASE, F[0]);
            UARTCharPut(UART0_BASE, F[1]);
            UARTCharPut(UART0_BASE,'F');
            UARTCharPut(UART0_BASE,'\n');
            UARTCharPut(UART0_BASE,'\r');
            break;


        }
    }
}
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
int main(void) {

        SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN |
SYSCTL_XTAL_16MHZ);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCHardwareOversampleConfigure(ADC0_BASE, 32);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE,2,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 2);

    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,
        (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
    IntMasterEnable();

    ADCIntEnable(ADC0_BASE,2);

    IntEnable(INT_UART0);
    UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT);

    UARTCharPut(UART0_BASE,'U');
    UARTCharPut(UART0_BASE,'A');
    UARTCharPut(UART0_BASE,'R');
    UARTCharPut(UART0_BASE,'T');
    UARTCharPut(UART0_BASE,' ');
    UARTCharPut(UART0_BASE,'a');
    UARTCharPut(UART0_BASE,'n');
    UARTCharPut(UART0_BASE,'d');
    UARTCharPut(UART0_BASE,' ');
    UARTCharPut(UART0_BASE,'L');
    UARTCharPut(UART0_BASE,'E');
    UARTCharPut(UART0_BASE,'D');
    UARTCharPut(UART0_BASE,'\n');
    UARTCharPut(UART0_BASE,'\r');
    UARTCharPut(UART0_BASE,'R');
    UARTCharPut(UART0_BASE,':');
    UARTCharPut(UART0_BASE,' ');
    UARTCharPut(UART0_BASE,'r');
    UARTCharPut(UART0_BASE,'e');
    UARTCharPut(UART0_BASE,'d');
    UARTCharPut(UART0_BASE,',');
```

```
    UARTCharPut(UART0_BASE,' ');
    UARTCharPut(UART0_BASE,'G');
    UARTCharPut(UART0_BASE,':');
    UARTCharPut(UART0_BASE,' ');
    UARTCharPut(UART0_BASE,'g');
    UARTCharPut(UART0_BASE,'r');
    UARTCharPut(UART0_BASE,'e');
    UARTCharPut(UART0_BASE,'e');
    UARTCharPut(UART0_BASE,'n');
    UARTCharPut(UART0_BASE,',');
    UARTCharPut(UART0_BASE,' ');
    UARTCharPut(UART0_BASE,'B');
    UARTCharPut(UART0_BASE,':');
    UARTCharPut(UART0_BASE,' ');
    UARTCharPut(UART0_BASE,'b');
    UARTCharPut(UART0_BASE,'l');
    UARTCharPut(UART0_BASE,'u');
    UARTCharPut(UART0_BASE,'e');
    UARTCharPut(UART0_BASE,',');
    UARTCharPut(UART0_BASE,'T');
    UARTCharPut(UART0_BASE,':');
    UARTCharPut(UART0_BASE,' ');
    UARTCharPut(UART0_BASE,'t');
    UARTCharPut(UART0_BASE,'e');
    UARTCharPut(UART0_BASE,'m');
    UARTCharPut(UART0_BASE,'p');
    UARTCharPut(UART0_BASE,'e');
    UARTCharPut(UART0_BASE,'r');
    UARTCharPut(UART0_BASE,'a');
    UARTCharPut(UART0_BASE,'t');
    UARTCharPut(UART0_BASE,'u');
    UARTCharPut(UART0_BASE,'r');
    UARTCharPut(UART0_BASE,'e');
    UARTCharPut(UART0_BASE,'.');
    UARTCharPut(UART0_BASE,'\n');
    UARTCharPut(UART0_BASE,'\r');


    while (1)
    {

    }

}----------------------------------------------------------------------------
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.