Caroline Yao
Ali Tejani

Lab 4 Prep

C) Modify lines 97,98,99

```
#define SSID_NAME  "Caroline Phone"

#define SEC_TYPE   SL_SEC_TYPE_OPEN

#define PASSKEY    "bang133*"
```

D) Write a C function that extracts the Austin temperature from this buffer.

```
char* ParseBuffer(char* recvbuff){
    uint32_t j = 0;
    uint32_t k = TEMP_START_INDEX;
    for(uint32_t i = 0; i < MAX_RECV_BUFF_SIZE; ++i){
        if(recvbuff[i] == 't'){
            int result = CompareString(&recvbuff[i+1],"emp\":",5);
            if(result){
                for(j=i+6; j<TEMP_STRING_SIZE; ++j){
                    if(recvbuff[j] == ','){
                        recvbuff[j+1] = ' ';
                        recvbuff[j+2] = 'F';
                        ST7735_OutString(Temp);
                        return Temp;
                    }
                    Temp[k] = recvbuff[i+6];
                    ++k;
                }
            }
        }
    }
    return "0";
}
```

F) Write software to sample the ADC once using 64-point hardware averaging and calculate a physical parameter with units.

```
void ReadVoltage(void){
        uint32_t ADCVal = ADC0_InSeq3();
        uint32_t voltage = (ADCVal * 3300 + 2048)/4096;  //converts value to fixed point resolution .001
        ST7735_OutString("Voltage~");
        ST7735_OutUDec(voltage/1000);
        ST7735_OutChar('.');
        ST7735_OutUDec(voltage%1000);
}
```

ADC Sampling Functions

```
void ADC0_InitSWTriggerSeq3_Ch9(void){
  SYSCTL_RCGCADC_R |= 0x0001;   // 7) activate ADC0
                              // 1) activate clock for Port E
  SYSCTL_RCGCGPIO_R |= 0x10;
  while((SYSCTL_PRGPIO_R&0x10) != 0x10){};
  GPIO_PORTE_DIR_R &= ~0x10;      // 2) make PE4 input
  GPIO_PORTE_AFSEL_R |= 0x10;     // 3) enable alternate function on PE4
  GPIO_PORTE_DEN_R &= ~0x10;      // 4) disable digital I/O on PE4
  GPIO_PORTE_AMSEL_R |= 0x10;     // 5) enable analog functionality on PE4

//  while((SYSCTL_PRADC_R&0x0001) != 0x0001){};    // good code, but not yet implemented in simulator



  ADC0_PC_R &= ~0xF;              // 7) clear max sample rate field
  ADC0_PC_R |= 0x1;               //    configure for 125K samples/sec
  ADC0_SSPRI_R = 0x0123;          // 8) Sequencer 3 is highest priority
  ADC0_ACTSS_R &= ~0x0008;        // 9) disable sample sequencer 3
  ADC0_EMUX_R &= ~0xF000;         // 10) seq3 is software trigger

  ADC0_SSMUX3_R &= ~0x000F;       // 11) clear SS3 field

  ADC0_SSMUX3_R += 9;             //    set channel

  ADC0_SSCTL3_R = 0x0006;         // 12) no TS0 D0, yes IE0 END0

  ADC0_IM_R &= ~0x0008;           // 13) disable SS3 interrupts

        ADC0_SAC_R = 0x06;                                          //64x hardware
averaging

  ADC0_ACTSS_R |= 0x0008;         // 14) enable sample sequencer 3


}




//------------ADC0_InSeq3------------

// Busy-wait Analog to digital conversion

// Input: none

// Output: 12-bit result of ADC conversion

uint32_t ADC0_InSeq3(void){  uint32_t result;

  ADC0_PSSI_R = 0x0008;            // 1) initiate SS3

  while((ADC0_RIS_R&0x08)==0){};   // 2) wait for conversion done

    // if you have an A0-A3 revision number, you need to add an 8 usec wait here

  result = ADC0_SSFIFO3_R&0xFFF;   // 3) read result

  ADC0_ISC_R = 0x0008;             // 4) acknowledge completion

  return result;

}
```