CPE 325: Intro to Embedded Computer System

Lab 07

UART Communication

Submitted by: Hannah Kelley

Date of Experiment: October 14, 2025

Report Deadline: October 20, 2025

Demonstration Deadline: October 20, 2025

Theory

Topic 1: Serial Communication and UART

a) Serial communication is the means of transferring data between two devices over a single line of communication a single bit at a time. The MSP430 microcontroller used in this lab supports synchronous and a synchronous communication. This lab uses UART (Universal Asynchronous Receiver-Transmitter), an asynchronous protocol that does not require the sender and receiver to have a shared clock. Instead, both devices must be configured to operate at the same baud rate—the rate at which bits are transmitted per second—to ensure proper timing and interpretation of data.

Topic 2: UAH Serial App

a) The UAH Serial App is a communication app used to visualize numerical data over time. Data is received by the app one packet at a time with a header byte showing the start of each packet. Properly formatting these packets ensures that the app interprets and displays the incoming data correctly.

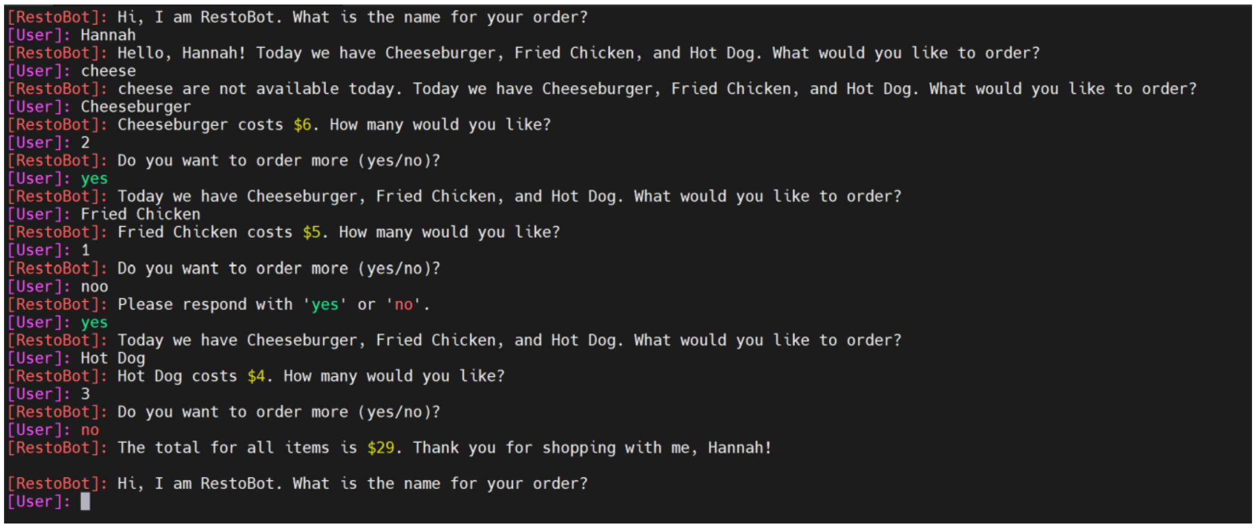
Results & Observation

Program 1:

Program Description:

In this lab, I implemented and tested a UART-based communication interface in C. I wrote functions to initialize the UART module at 115,200 baud (UART_SETUP), send and receive individual characters (UART_sendChar, UART_getChar), transmit strings (UART_sendString), and safely read user input lines with null termination and buffer limits (UART_getLine). After verifying that each function handled input/output correctly without accessing invalid memory, I used them to build RestoBot, an interactive restaurant chatbot. RestoBot greets the user, asks for their name, lists a menu (Cheeseburger, Fried Chicken, Hot Dog), validates orders, collects quantities, allows multiple items, calculates the total, and finally thanks the user before restarting for the next customer—all through UART communication in a terminal emulator. Additionally, the user and RestoBot have colored labels at the start of their respective messages and backspace handling has been implemented.

Program Output:

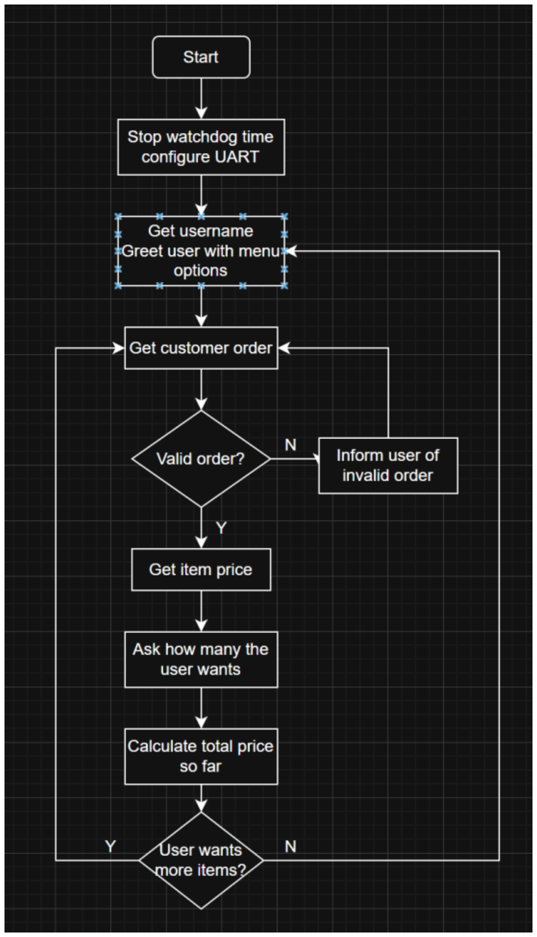


```
[RestoBot]: Hi, I am RestoBot. What is the name for your order?
[User]: Hannah
[RestoBot]: Hello, Hannah! Today we have Cheeseburger, Fried Chicken, and Hot Dog. What would you like to order?
[User]: cheese
[RestoBot]: cheese are not available today. Today we have Cheeseburger, Fried Chicken, and Hot Dog. What would you like to order?
[User]: Cheeseburger
[RestoBot]: Cheeseburger costs $6. How many would you like?
[User]: 2
[RestoBot]: Do you want to order more (yes/no)?
[User]: yes
[RestoBot]: Today we have Cheeseburger, Fried Chicken, and Hot Dog. What would you like to order?
[User]: Fried Chicken
[RestoBot]: Fried Chicken costs $5. How many would you like?
[User]: 1
[RestoBot]: Do you want to order more (yes/no)?
[User]: noo
[RestoBot]: Please respond with 'yes' or 'no'.
[User]: yes
[RestoBot]: Today we have Cheeseburger, Fried Chicken, and Hot Dog. What would you like to order?
[User]: Hot Dog
[RestoBot]: Hot Dog costs $4. How many would you like?
[User]: 3
[RestoBot]: Do you want to order more (yes/no)?
[User]: no
[RestoBot]: The total for all items is $29. Thank you for shopping with me, Hannah!

[RestoBot]: Hi, I am RestoBot. What is the name for your order?
[User]: █
```

Figure 01: RestoBot Output

Program Flowchart:



Start
Stop watchdog time configure UART
Get username Greet user with menu options
Get customer order
Valid order?
N
Inform user of invalid order
Y
Get item price
Ask how many the user wants
Calculate total price so far
Y
User wants more items?
N

Figure 02: Program 1 Flowchart

Program 2:

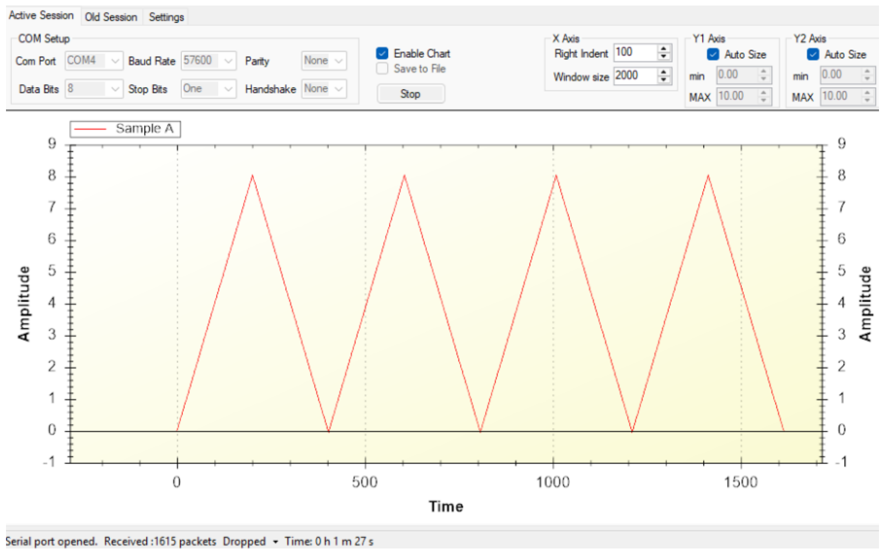
Program Description:

In this lab, I wrote a C program to generate a triangular wave and transmit it to the UAH Serial App using UART at 57,600 baud. The waveform parameters were:

- Amplitude: 8 units
- Frequency: 2.5 Hz
- Duration: 4 full periods

The program continuously calculated sample values to form the triangular waveform and sent them sequentially over UART so that the serial app could display the waveform in real time, matching the expected example output.

Program Output:



COM Setup: Com Port COM4, Baud Rate 57600, Parity None, Data Bits 8, Stop Bits One, Handshake None

✓ Enable Chart
Save to File

X Axis: Right Indent 100, Window size 2000

Y1 Axis: ✓ Auto Size, min 0.00, MAX 10.00

Y2 Axis: ✓ Auto Size, min 0.00, MAX 10.00

Serial port opened. Received :1615 packets Dropped • Time: 0 h 1 m 27 s

Figure 03: Program 2 Output in UAH Serial App

Report Questions:

1. **What is the duration of the signal?**
   Since the period of the signal is T=1/f we know the period of one signal will be T=1/2.5 Hz = 0.4 seconds. Since the program transmits the signal for four full periods the signal will last a total of 1.6 seconds.

2. **What is the frequency of the signal?**
   To achieve the 2.5 Hz frequency while smoothly changing the waveform we need to increase from 0 to 8 over half the period of the waveform. To find the value to increment by we divide the total amplitude by the number of samples in each rising or falling period (8/200) to get an increment value of 0.04 units per sample sent.

Appendix

Table 01: Program 1 Source Code

```
/*--------------------------------------------------------------------------
 * File:       Lab07_P1.c
 * Description: interactive chatbot (RestoBot) implemented uisng UART
 *            115200 baud rate
 *
 * Board:      5529
 * Input:      user choices
 * Output:     chatbot responses
 * Author:     Hannah Kelley
 * Date:       October 15, 2025
 *--------------------------------------------------------------------------*/
#include <msp430.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define colorReset     "\x1b[0m"
#define colorBot       "\x1b[31m" // red like example
#define colorUser      "\x1b[35m" // purple cause why not
// function prototypes
void UART_SETUP();
void UART_sendChar(char bip);
char UART_getChar();
void UART_sendString(char* str);
void UART_getLine(char* buf, int limit);
int main() {
   WDTCTL = WDTPW + WDTHOLD; // stop WDT
   UART_SETUP();
   // arrays for printing
   char userName[50];  // allow username to be up to 50 chars long
   char item[15];      // max item name is 13 chars long
   char response[10];
   char message[120];
   int totalPrice;
   while (1) {
      UART_sendString(colorBot "[RestoBot]: " colorReset);
      UART_sendString("Hi, I am RestoBot. What is the name for your order?\r\n");
      UART_sendString(colorUser "[User]: " colorReset);
      UART_getLine(userName, 50); // get user's name
      // greeting after receiving name
      snprintf(message, sizeof(message), "Hello, %s! Today we have Cheeseburger, Fried Chicken, and
Hot Dog. What would you like to order?\r\n", userName);
      UART_sendString(colorBot "[RestoBot]: " colorReset);
      UART_sendString(message);
      totalPrice = 0;
      do {
         // have user select item
         UART_sendString(colorUser "[User]: " colorReset);
         UART_getLine(item, 15);
         while(strcmp(item, "Cheeseburger") != 0 && strcmp(item, "Fried Chicken") != 0 &&
strcmp(item, "Hot Dog") != 0) {
            snprintf(message, sizeof(message), "%s are not available today. Today we have
Cheeseburger, Fried Chicken, and Hot Dog. What would you like to order?", item);
            UART_sendString(colorBot "[RestoBot]: " colorReset);
            UART_sendString(message);
            UART_sendString(colorUser "[User]: " colorReset);
            UART_getLine(item, 15);
         }
         // item price and quantity stuff
         int PRICE = (strcmp(item, "Cheeseburger") == 0) ? 6:
               (strcmp(item, "Fried Chicken") == 0) ? 5: 4;
         snprintf(message, sizeof(message), "%s costs $%d. How many would you like? \r\n", item,
PRICE);
         UART_sendString(colorBot "[RestoBot]: " colorReset);
         UART_sendString(message);
         UART_sendString(colorUser "[User]: " colorReset);
         UART_getLine(response, 10);
         int QUANTITY = atoi(response); // convert str to int
         totalPrice += PRICE * QUANTITY; // update total price
         // ask if they want more
         UART_sendString(colorBot "[RestoBot]: " colorReset);
         UART_sendString("Do you want to order more (yes/no)?\r\n");
         UART_sendString(colorUser "[User]: " colorReset);
         UART_getLine(response, 10);
         // Validate yes/no response
         while (strcmp(response, "yes") != 0 && strcmp(response, "no") != 0) {
            UART_sendString(colorBot "[RestoBot]: " colorReset);
            UART_sendString("Please respond with 'yes' or 'no'.\r\n");
            UART_sendString(colorUser "[User]: " colorReset);
            UART_getLine(response, 10);
         }
         // If "yes", re-prompt for the next item
         if (strcmp(response, "yes") == 0) {
            UART_sendString(colorBot "[RestoBot]: " colorReset);
            UART_sendString("Today we have Cheeseburger, Fried Chicken, and Hot Dog. What would
you like to order?\r\n");
         }
      } while (strcmp(response, "yes") == 0);
      // total price and goodbye
      snprintf(message, sizeof(message), "The total for all items is $%d. Thank you for shopping with
me, %s!\r\n\n", totalPrice, userName);
```

```
        UART_sendString(colorBot "[RestoBot]: " colorReset);
        UART_sendString(message);
    }
}
// UART Setup()
void UART_SETUP() {
    P3SEL |= BIT3 + BIT4;   // set RXD/TXD pins
    UCA0CTL1 |= UCSWRST;    // sw reset
    UCA0CTL0 = 0;           // ctrl register
    UCA0CTL1 |= UCSSEL_2;   // use SMCLK
    UCA0BR0 = 0x09; // baud rate = 115200
    UCA0BR1 = 0x00;
    UCA0MCTL = 0x02;
    UCA0CTL1 &= ~UCSWRST; // init UART state machine
}
// send a char via UART
void UART_sendChar(char bip) {
    while (!(UCA0IFG & UCTXIFG)); // wait for TX buffer to be ready
    UCA0TXBUF = bip;
}
// retrieve a char via UART
char UART_getChar() {
    while (!(UCA0IFG & UCRXIFG));  // Wait for RX buffer to be ready
    return UCA0RXBUF;
}
// send a string via UART
void UART_sendString(char* str) {
    while(*str) {
        UART_sendChar(*str++);
    }
}
// receive a line of user input via UART (w/ backspace handling)
void UART_getLine(char* buf, int limit) {
    int i = 0;
    char ch;
    while (1) {
        ch = UART_getChar();
        if ((ch == '\r') || (ch == '\n')) { // enter key pressed
            buf[i] = '\0'; // null-terminate str
            UART_sendString("\r\n");
            break;
        } else if (ch == '\b' || ch == 127) { // backspace pressed
            if (i > 0) {
                i--; // move buffer ptr back
                UART_sendString("\b \b"); // clear char on screen
            }
        } else if ( i < limit - 1) { // normal char input
            buf[i++] = ch;
        UART_sendChar(ch); // echo print char
      }
    }
}
```

Table 02: Program 2 Source Code

```
/*----------------------------------------------------------------------------
 * File:        Lab07_P1.c
 * Description: displays a triangular wave form in UAH Serial App
                constructed from packets sent over UART
 *
 * Board:       5529
 * Input:       UART packets from WDT interrupt
 * Output:      triangular wave in UAH Serial App packets
 * Author:      Hannah Kelley
 * Date:        October 15, 2025
 *--------------------------------------------------------------------------*/
#include <msp430.h>
#include <stdint.h>
#include <stdbool.h>

volatile float myData = 0.0;
int i = 0; // counter for periods
bool direction = true; // for rising/falling, rising = true

void UART_setup(void) {
    P3SEL |= BIT3 | BIT4;    // Set USCI_A0 RXD/TXD to receive/transmit data
    UCA0CTL1 |= UCSWRST;        // Set software reset during initialization
    UCA0CTL1 |= UCSSEL_2;       // use smclk as clock for UART
    UCA0BR0 = 18;        // set lower byte of baud rate divider (1MHz/57600)
    UCA0BR1 = 0;                // set upper byte of bauud rate divider to 0
    UCA0MCTL= UCBRF_0 | UCBRS_2;// config modulation for accurate baud rate
    UCA0CTL1 &= ~UCSWRST;    // release USCI_A0 from reset, begin operation
}

void sendChar(char c) {
    while (!(UCA0IFG&UCTXIFG)); // Wait for previous character to transmit
    UCA0TXBUF = c;               // Put character into tx buffer
}
int main() {
    WDTCTL = WDT_MDLY_32;       // config WDT for 32ms interval interrupts
    UART_setup();               // Initialize USCI_A0 module in UART mode
    SFRIE1 |= WDTIE;            // Enable watchdog interrupts

    myData = 0.0;
    __bis_SR_register(LPM0_bits + GIE); // enter LPM 0 with GIE enabled
}
// Sends a ramp signal; amplitude of one period ranges from 0.0 to 9.9
#pragma vector = WDT_VECTOR
__interrupt void watchdog_timer(void) {
    char *dataPtr = (char*)&myData; // ptr to bytes of float waveform
    sendChar(0x55);      // send a start byte to indicate start of packet
    // send each byte over UART
    int j;
    for (j = 0; j < 4; j++) {
        sendChar(dataPtr[j]);
    }
    // update value and phase if needed
    if (direction) { // if rising
        myData += 0.04;
        if (myData >= 8.0) {
            direction = false;
        }
    }
    else { // if falling
        myData -= 0.04;
        if (myData <= 0.0) {
            direction = true;
            i++; // one period done
        }
    }
    if (i == 4) {
        SFRIE1 &= ~WDTIE; // disable WDT interrupts if 4 periods sent
    }
}
```