Muhammad Sharjeel and Naafiul Hossain

115185427

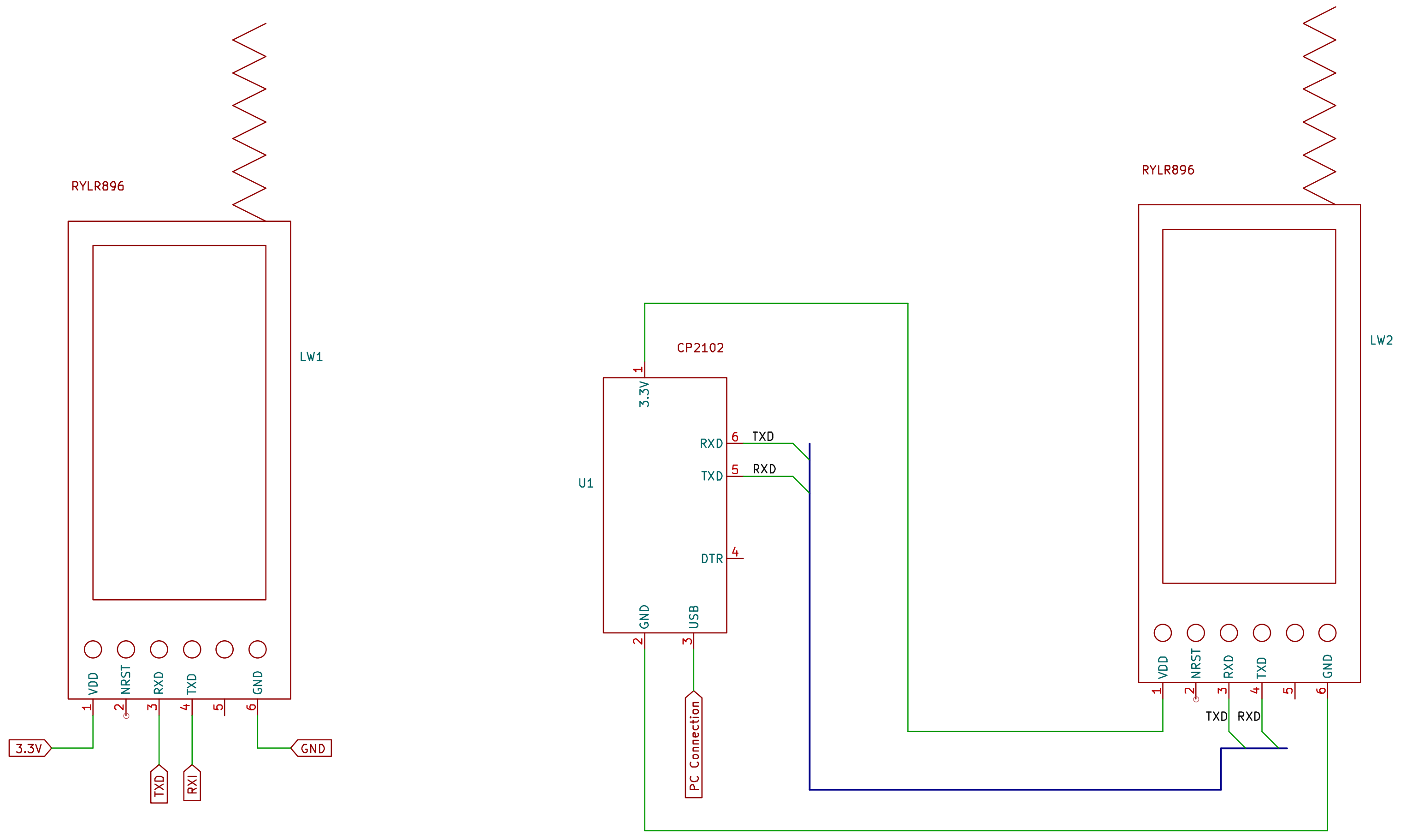
115107623

Pre-Lab 9: Asynchronous Serial (RS232) Communication Over a LoraWAN Channel

ESE 381 Section L02

Bench 7

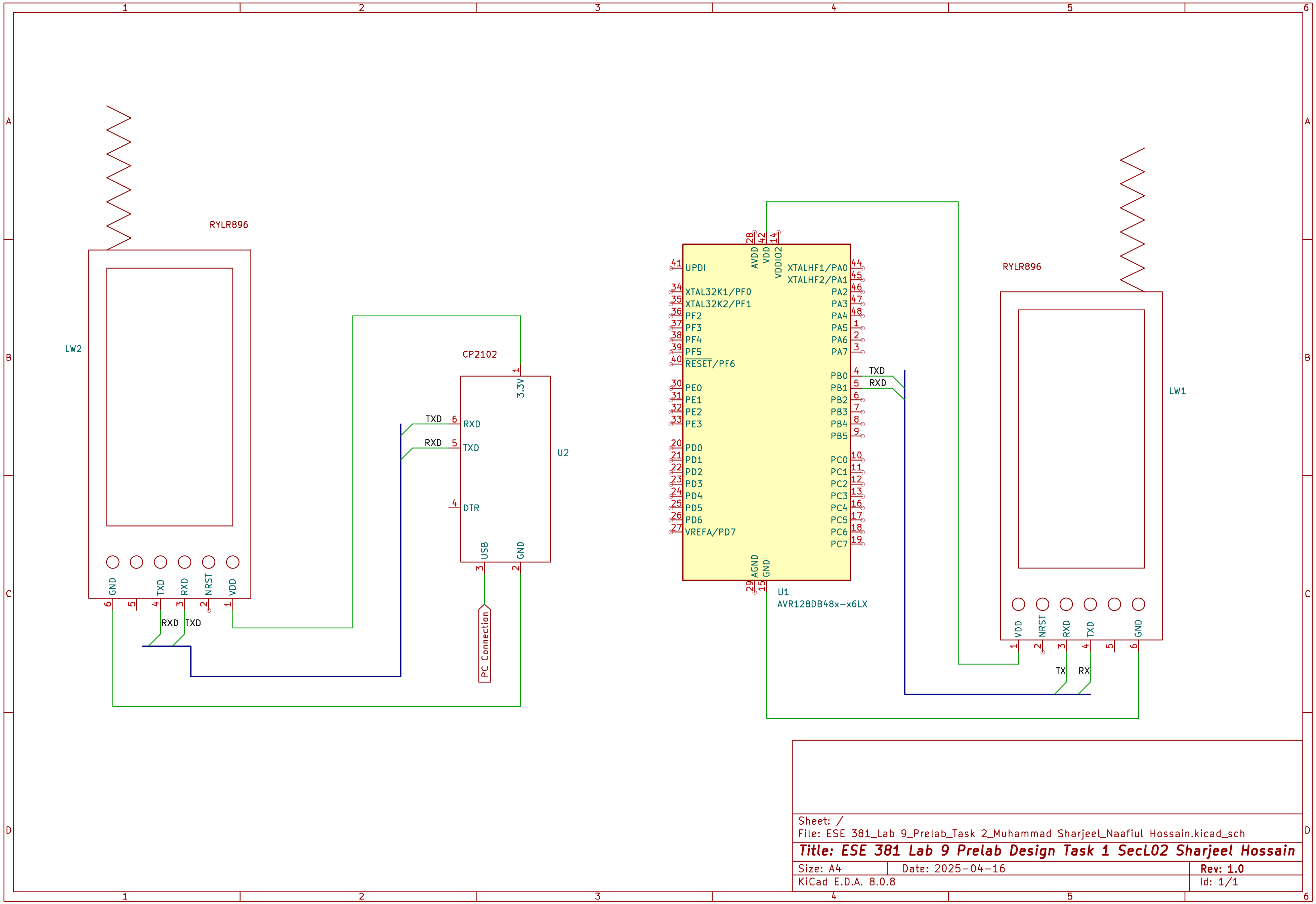
Breadboard: K2



Sheet: /
File: ESE 381_Lab 9_Prelab_Task 1_Muhammad Sharjeel_Naafiul Hossain.kicad_sch

Title: ESE 381 Lab 9 Prelab Design Task 1 SecL02 Sharjeel Hossain

Size: A4	Date: 2025-04-15	**Rev: 1.0**
KiCad E.D.A. 8.0.8		Id: 1/1



Sheet: /
File: ESE 381_Lab 9_Prelab_Task 2_Muhammad Sharjeel_Naafiul Hossain.kicad_sch

Title: ESE 381 Lab 9 Prelab Design Task 1 SecL02 Sharjeel Hossain

Size: A4	Date: 2025-04-16	**Rev: 1.0**
KiCad E.D.A. 8.0.8		Id: 1/1

```c
/*
 * receive_payload.c
 * Task 2 - Receive message via LoRa (USART1), extract payload, send to PC via USART3
 * Author: Naafiul Hossain
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <string.h>
#include <stdio.h>

#define F_CPU 4000000UL
#define BAUD_RATE 115200
#define USART_BAUD ((uint16_t)((float)(F_CPU * 64) / (16.0f * BAUD_RATE) + 0.5f))

#define RX_BUFFER_SIZE 80
#define TX_BUFFER_SIZE 40

// === RX and TX Buffers ===
char rxU1_buff[RX_BUFFER_SIZE] = {0};     // Incoming LoRa message
char txU3_buff[TX_BUFFER_SIZE] = {0};     // Payload to PC via USART3
uint8_t rxU1_index = 0;
uint8_t txU3_index = 0;
uint8_t rxU1_buff_dav = 0;
uint8_t txU3_buff_dav = 0;

// === Parsed Fields ===
char RCV_preamble[10];
char payload[40];
uint16_t txmtr_addr;
uint16_t rcv_data_len;
int16_t RSSI;
int16_t SNR;

void USART1_init(void);  // RX from LoRa
void USART3_init(void);  // TX to PC
void parse_rxU1_buff(void);

int main(void) {
    USART1_init();
    USART3_init();
    sei();  // Enable global interrupts

    while (1) {
        if (rxU1_buff_dav) {
            parse_rxU1_buff();           // Extract payload from received LoRa message
            USART3.CTRLA |= USART_DREIE_bm;  // Enable TX ISR to send to CoolTerm
        }
    }
}

// === Initialize USART1 (RX from LoRa) ===
void USART1_init(void) {
    PORTC.DIRCLR = PIN0_bm;  // PC0 = RX1
    USART1.CTRLB = USART_RXEN_bm;
    USART1.CTRLC = USART_CHSIZE_8BIT_gc;
    USART1.BAUD = USART_BAUD;
    USART1.CTRLA = USART_RXCIE_bm;
}

// === Initialize USART3 (TX to PC) ===
void USART3_init(void) {
    PORTB.DIRSET = PIN0_bm;  // PB0 = TX3
    USART3.CTRLB = USART_TXEN_bm;
    USART3.CTRLC = USART_CHSIZE_8BIT_gc;
    USART3.BAUD = USART_BAUD;
}

// === ISR: USART1 RX Complete - Receive LoRa message ===
ISR(USART1_RXC_vect) {
    char c = USART1.RXDATAL;

    if (rxU1_index < RX_BUFFER_SIZE - 1) {
        rxU1_buff[rxU1_index++] = c;

        if (c == '\n') {
            rxU1_buff[rxU1_index] = '\0';
            rxU1_index = 0;
            rxU1_buff_dav = 1;
            USART1.CTRLA &= ~USART_RXCIE_bm;  // Disable RX interrupt until parsed
        }
    } else {
        rxU1_index = 0;  // Prevent overflow
    }
}

// === ISR: USART3 DRE - Send payload one char at a time ===
ISR(USART3_DRE_vect) {
    if (txU3_buff_dav && txU3_buff[txU3_index] != '\0') {
        USART3.TXDATAL = txU3_buff[txU3_index++];
    } else {
        USART3.CTRLA &= ~USART_DREIE_bm;
        txU3_buff_dav = 0;
    }
}

// === Parse LoRa message and copy payload to txU3_buff ===
void parse_rxU1_buff(void) {
    sscanf(rxU1_buff, "%[^=]=%u,%u,%[^,],%d,%d",
           RCV_preamble, &txmtr_addr, &rcv_data_len, payload, &RSSI, &SNR);

    snprintf(txU3_buff, sizeof(txU3_buff), "%s\r\n", payload);  // Add CRLF 
      for terminal display

    rxU1_buff_dav = 0;
    txU3_buff_dav = 1;
    txU3_index = 0;

    memset(rxU1_buff, 0, sizeof(rxU1_buff));
    USART1.CTRLA |= USART_RXCIE_bm;  // Re-enable RX
}

```

```c
/*
 * LoRaWAN_reply.c
 * Task 3 - AVR receives LoRa message and echoes payload back via AT+SEND
 * Author: Naafiul Hossain
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <string.h>
#include <stdbool.h>
#include <stdio.h>

#define F_CPU 4000000UL
#define BAUD_RATE 115200
#define USART1_BAUD ((uint16_t)((float)(F_CPU * 64) / (16.0f * BAUD_RATE) + 
  0.5f))

#define BENCH_NUM 7
#define PC_ADDRESS (BENCH_NUM + 30)  // Destination LW2 address (PC side)

#define RX_BUFFER_SIZE 80
#define TX_BUFFER_SIZE 64

// === RX and TX Buffers ===
char rxU1_buff[RX_BUFFER_SIZE] = {0};  // Receive from LW1
char *rxU1_ptr = rxU1_buff;
uint8_t rxU1_index = 0;
uint8_t rxU1_buff_dav = 0;             // Data Available Flag

char txU1_buff[TX_BUFFER_SIZE] = {0};  // Transmit to LW1
char *txU1_ptr = txU1_buff;
uint8_t txU1_index = 0;
uint8_t txU1_buff_dav = 0;             // Transmission ready flag

// === Parsed Message Fields ===
char RCV_preamble[10];
volatile uint16_t txmtr_address;
volatile uint16_t rcv_data_len;
char payload[40];                      // Payload from message
volatile int16_t RSSI;
volatile int16_t SNR;
uint8_t payload_index = 0;

void USART1_init(void);
void parse_rxU1_buff(void);

int main(void) {
    USART1_init();
    sei();  // Enable global interrupts

    while (1) {
        if (rxU1_buff_dav) {
            parse_rxU1_buff();               // Parse and prep reply
            USART1.CTRLA |= USART_DREIE_bm;  // Enable TX ISR
        }
    }
}

// === USART1 Initialization ===
void USART1_init(void) {
    PORTC.DIRCLR = PIN0_bm;  // PC0 = RX (input)
    PORTC.DIRSET = PIN1_bm;  // PC1 = TX (output)

    USART1.CTRLB = USART_RXEN_bm | USART_TXEN_bm;
    USART1.CTRLC = USART_CHSIZE_8BIT_gc;
    USART1.BAUD = USART1_BAUD;
    USART1.CTRLA = USART_RXCIE_bm;  // Enable RX interrupt
}

// === USART1 RX ISR: Receives characters into rxU1_buff ===
ISR(USART1_RXC_vect) {
    char c = USART1.RXDATAL;

    if (rxU1_index < RX_BUFFER_SIZE - 1) {
        rxU1_buff[rxU1_index++] = c;

        if (c == '\n') {
            rxU1_buff[rxU1_index] = '\0';  // Null terminate
            rxU1_buff_dav = 1;             // Message complete
            rxU1_index = 0;
            USART1.CTRLA &= ~USART_RXCIE_bm;  // Disable RX interrupt until 
              processed
        }
    } else {
        rxU1_index = 0;  // Prevent overflow
    }
}

// === USART1 TX ISR: Transmit txU1_buff one byte at a time ===
ISR(USART1_DRE_vect) {
    if (txU1_buff_dav && txU1_buff[txU1_index] != '\0') {
        USART1.TXDATAL = txU1_buff[txU1_index++];
    } else {
        USART1.CTRLA &= ~USART_DREIE_bm;  // Disable TX interrupt
        txU1_buff_dav = 0;
    }
}

```

```c
// === Parse incoming +RCV=... and construct AT+SEND=... ===
void parse_rxU1_buff(void) {
    sscanf(rxU1_buff, "%[^=]=%u,%u,%[^,],%d,%d", RCV_preamble, &txmtr_address, &rcv_data_len, payload, &RSSI, &SNR);

    // Construct reply: AT+SEND=<dest_addr>,<length>,<payload>\r\n
    snprintf(txU1_buff, sizeof(txU1_buff), "AT+SEND=%d,%d,%s\r\n", PC_ADDRESS, rcv_data_len, payload);

    // Clear flags and prep TX
    rxU1_buff_dav = 0;
    txU1_buff_dav = 1;
    txU1_index = 0;

    memset(rxU1_buff, 0, sizeof(rxU1_buff));
    USART1.CTRLA |= USART_RXCIE_bm;  // Re-enable RX
}

```