CPE301 – SPRING 2019

Design Assignment DA5

Student Name: Daniel Senda

Student #: 5002362633

Student Email: sendad1@unlv.nevada.edu

Primary Github address: https://github.com/dsenda/Smiles

Directory: DA5

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

*List of Components used:*

Atmel Studio 7

ATmega328P Xplained Mini

LM34 Temperature Sensor

FTDI Basic chip

NRF24L01+ RF Module

Breadboard

Jumper wires

Classmates set-up in same channel

*Block diagram about the transmission connections:*

LM34 Temperature

Sensor #1

ATmega328p

#1

RF Module

#1

RF Module

#2

ATmega328p

#2

LM34 Temperature

Sensor #2

1. **INITIAL/MODIFIED CODE FROM TASK DA3b**

// c\_code\_temperature\_on\_serial2usb.c

// Daniel Senda

// Monitors LM34 to display temperature on serial terminal every 1s.

// Uses a timer with interrupt for the 1s delay.

#define *F\_CPU* 16000000UL // Needed Definitions.

#define BAUD\_RATE 9600

#include <avr/io.h> // Needed Libraries.

#include <util/delay.h>

#include <avr/interrupt.h>

volatile int O\_F = 0; // Used for overflow count.

int num = 0; // Used for while loops.

void usart\_init ();

void usart\_send (unsigned char ch);

void initiate\_timer()

{

TCCR0A = 0b10; // Sets CTC mode.

TCCR0B = 0b00000101; // Sets 1024 pre-scaler.

OCR0A = 255; // Sets output compare register A to 255.

TIMSK0 |= (1 << OCIE0A); // Enables output compare match interrupt.

sei(); // Enables global interrupts.

TCNT0 = 0; // Resets timer.

}

ISR(TIMER0\_COMPA\_vect) // Keeps count of overflows.

{

O\_F++; // Increments overflow count.

}

int main (void)

{

usart\_init ();

// Setup and enable ADC

ADMUX = (0<<REFS1)| // Reference Selection Bits.

(1<<REFS0)| // AVcc - external cap at AREF.

(0<<ADLAR)| // ADC Left Adjust Result.

(1<<MUX2)| // Analog Channel Selection Bits.

(0<<MUX1)| // ADC4 (PC5).

(1<<MUX0);

ADCSRA = (1<<ADEN)| // ADC ENable.

(0<<ADSC)| // ADC Start Conversion.

(0<<ADATE)| // ADC Auto Trigger Enable.

(0<<ADIF)| // ADC Interrupt Flag.

(0<<ADIE)| // ADC Interrupt Enable.

(1<<ADPS2)| // ADC Pre-scaler Select Bits.

(0<<ADPS1)|

(1<<ADPS0);

initiate\_timer(); // Function that initiates timer.

while (1)

{

ADCSRA|=(1<<ADSC); // Starts conversion.

while((ADCSRA&(1<<ADIF))==0); // Waits for conversion to finish.

ADCSRA |= (1<<ADIF);

num = 0; // Resets num count.

TCNT0 = 0; // Resets timer.

while (num <= 0)

{

if (O\_F >= 61)

{

int a = ADCL;

a = a | (ADCH<<8);

a = (a/1024.0) \* 5000/10;

usart\_send((a/100)+'0');

a = a % 100;

usart\_send((a/10)+'0');

a = a % 10;

usart\_send((a)+'0');

usart\_send('\r');

num++; // Increments num to exit while loop.

O\_F = 0; // Resets overflow count.

}

}

}

return 0;

}

void usart\_init (void) // Initializes USART.

{

UCSR0B = (1<<TXEN0);

UCSR0C = (1<< UCSZ01)|(1<<UCSZ00);

UBRR0L = *F\_CPU*/16/BAUD\_RATE-1;

}

void usart\_send (unsigned char ch) // Transmits characters to computer.

{

while (! (UCSR0A & (1<<UDRE0))); // Waits until UDR0 is empty.

UDR0 = ch; // Transmit ch.

}

void usart\_print(char\* str) // Prints characters on computer.

{

int i = 0;

while(str[i] != 0)

{

usart\_send(str[i]);

}

}

1. **DEVELOPED MODIFIED CODE OF TASK DA5**

// c\_code\_RF\_temperature\_serial2usb.c

// Daniel Senda

// Interface the provided NRF24L01+ RF Module to the ATmega328p using the SPI interface.

// Using the earlier developed code for ADC, transmit the ADC value of the internal

// temp sensor, or LM35 sensor between two RF Modules. The ATmega328p interfacing the

// RF Modules should alternate between TX and RX modes every 0.5 secs (hopefully they are

// not both at TX and RX modes in the same interval). The temp of both ATmega328p’s

// should be displayed on both ATmega328p’s.

#ifndef *F\_CPU* // Sets clock frequency.

#define *F\_CPU* 16000000UL

#endif

#include <avr/io.h> // Includes needed libraries.

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#include <stdio.h>

#include <string.h>

#include "inc\nrf24l01.c" // Include nRF24L01+ library.

#include "inc\nrf24l01-mnemonics.h"

#include "inc\spi.c"

#ifndef BAUD // Sets up UART for printf();

#define BAUD 9600

#endif

#include "inc\STDIO\_UART.c"

void print\_config(void); // Functions

void adc\_init(void);

volatile bool message\_received = false; // Used in IRQ ISR.

volatile bool status = false; // Used in IRQ ISR.

int tempf = 0; // Used for temperature data.

int main(void)

{

adc\_init(); // Initializes the ADC.

char tx\_message[32]; // Defines string array.

*strcpy*(tx\_message,"Hello World!"); // Copies string into array.

uart\_init(); // Initializes UART.

nrf24\_init(); // Initializes nRF24L01+ and print configuration info.

print\_config(); // Configures prints.

nrf24\_start\_listening(); // Start listening to incoming messages.

nrf24\_send\_message(tx\_message); // Sends message.

while (1)

{

ADCSRA|=(1<<ADSC); // Starts conversion.

while((ADCSRA&(1<<ADIF))==0); // Waits for conversion to finish.

ADCSRA |= (1<<ADIF); // Resets flag for conversion.

tempf = ADCL; // Records temp sensor data.

tempf = tempf | (ADCH<<8);

tempf = (tempf/1024.0) \* 5000/10;

char temp[5]; // Variable used to store tempf string.

*itoa*(tempf, temp, 10); // Converts tempf integer to string.

if (message\_received)

{

message\_received = false; // Message received, print it.

*printf*("Received message: %s\n",nrf24\_read\_message());

*\_delay\_ms*(500);

status = nrf24\_send\_message(temp); // Send message as response.

if (status == true) *printf*("Message sent successfully\n");

}

}

}

ISR(INT0\_vect) // Interrupt on IRQ pin.

{

message\_received = true;

}

void print\_config(void)

{

*uint8\_t* data;

*printf*("Startup successful\n\n nRF24L01+ configured as:\n");

*printf*("-------------------------------------------\n");

nrf24\_read(CONFIG,&data,1);

*printf*("CONFIG 0x%x\n",data);

nrf24\_read(EN\_AA,&data,1);

*printf*("EN\_AA 0x%x\n",data);

nrf24\_read(EN\_RXADDR,&data,1);

*printf*("EN\_RXADDR 0x%x\n",data);

nrf24\_read(SETUP\_RETR,&data,1);

*printf*("SETUP\_RETR 0x%x\n",data);

nrf24\_read(RF\_CH,&data,1);

*printf*("RF\_CH 0x%x\n",data);

nrf24\_read(RF\_SETUP,&data,1);

*printf*("RF\_SETUP 0x%x\n",data);

nrf24\_read(STATUS,&data,1);

*printf*("STATUS 0x%x\n",data);

nrf24\_read(FEATURE,&data,1);

*printf*("FEATURE 0x%x\n",data);

*printf*("-------------------------------------------\n\n");

}

void adc\_init (void) // Sets up and enables ADC.

{

ADMUX = (0<<REFS1)| // Reference Selection Bits.

(1<<REFS0)| // AVcc - external cap at AREF.

(0<<ADLAR)| // ADC Left Adjust Result.

(0<<MUX2)| // Analog Channel Selection Bits.

(0<<MUX1)| // ADC0 (PC0).

(0<<MUX0);

ADCSRA = (1<<ADEN)| // ADC Enable.

(0<<ADSC)| // ADC Start Conversion.

(0<<ADATE)| // ADC Auto Trigger Enable.

(0<<ADIF)| // ADC Interrupt Flag.

(0<<ADIE)| // ADC Interrupt Enable.

(1<<ADPS2)| // ADC Pre-scaler Select Bits.

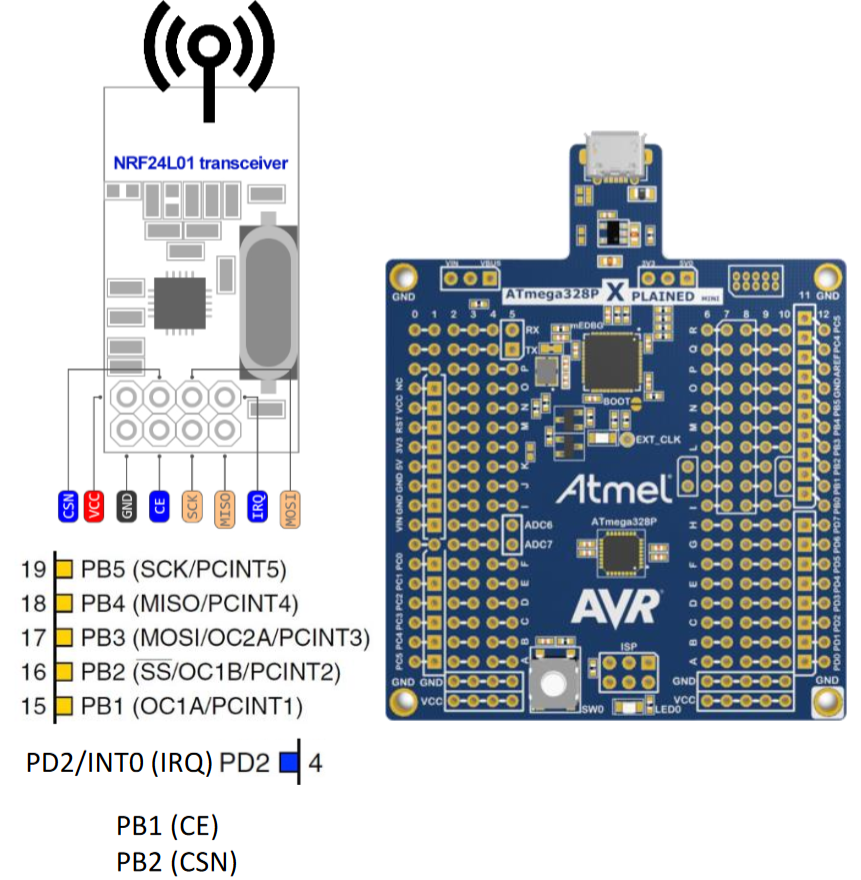
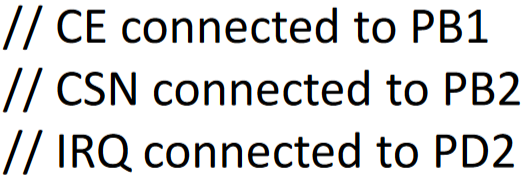
(0<<ADPS1)|

(1<<ADPS0);

}

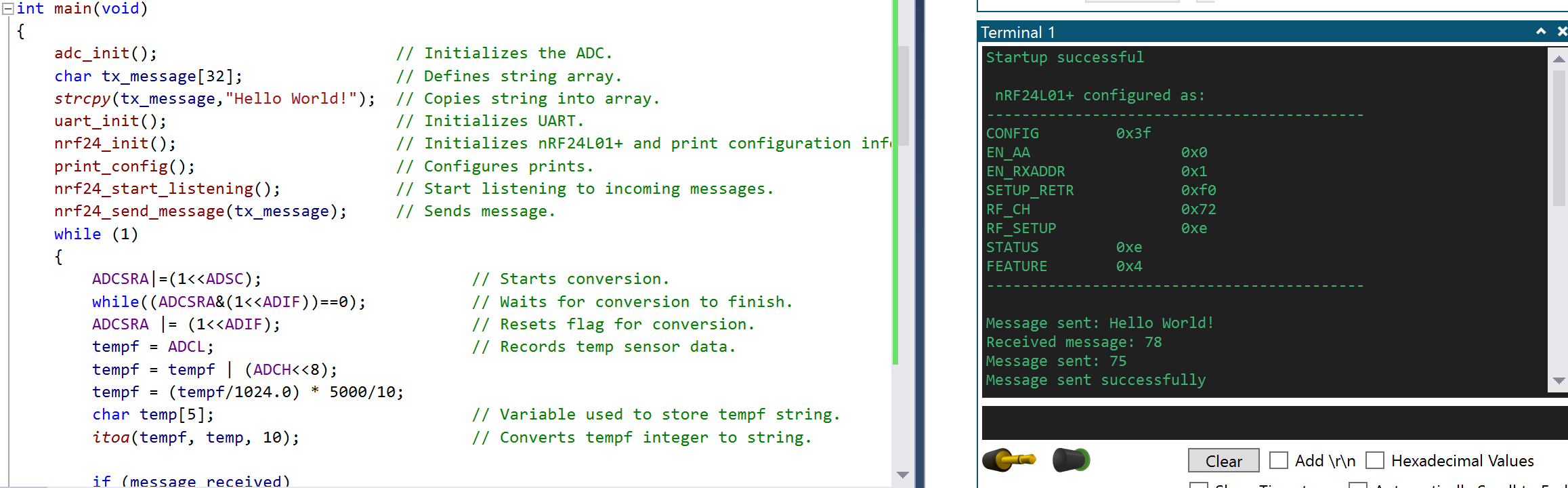
1. **SCHEMATICS**

The following schematic from the slides describe the connections that are made relatively well.



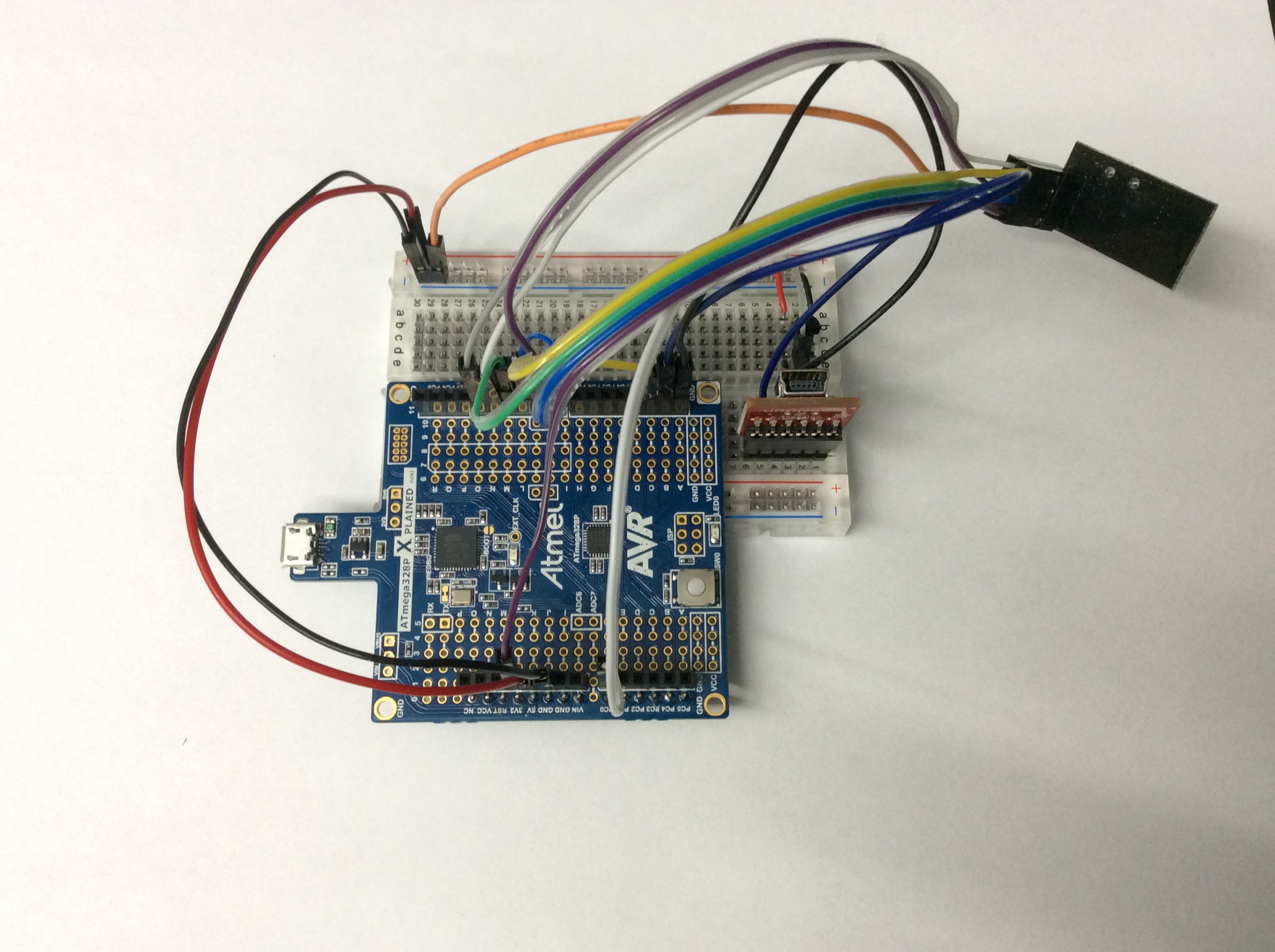
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

The following screen shot shows that the student successfully sent and received temperature information through the RF module.



1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

The following is a picture of the board setup:



1. **VIDEO LINKS OF EACH DEMO**

https://youtu.be/B2bz1ifVJcg

1. **GITHUB LINK OF THIS DA**

<https://github.com/dsenda/Smiles/tree/master/DA5>

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Daniel Senda