Brian Medrano Kiaer
Partner: Justine Macato

CPE301 – SPRING 2018

Midterm 2

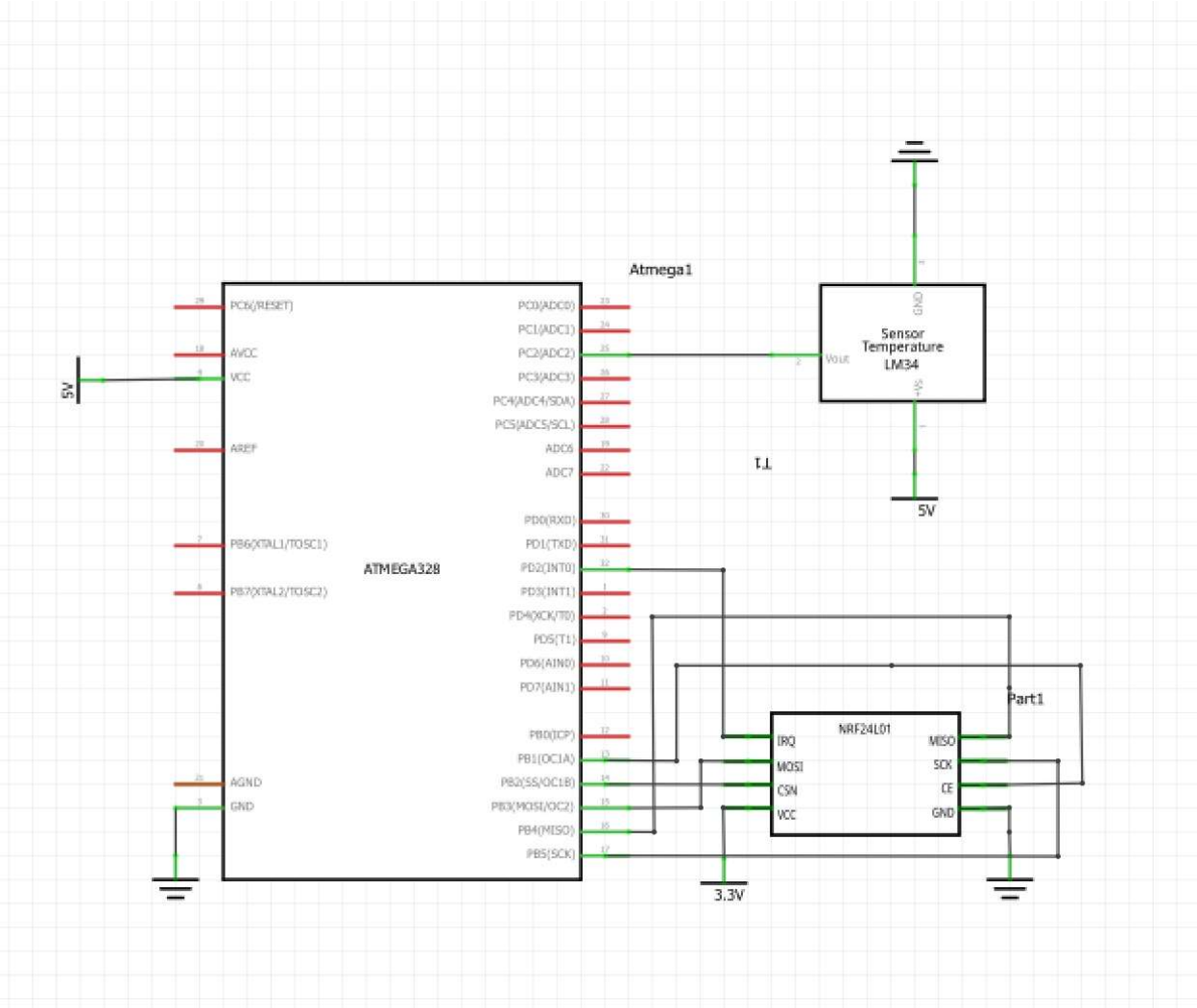
DO NOT REMOVE THIS PAGE DURING SUBMISSION:

The student understands that all required components should be submitted in complete for grading of this assignment.

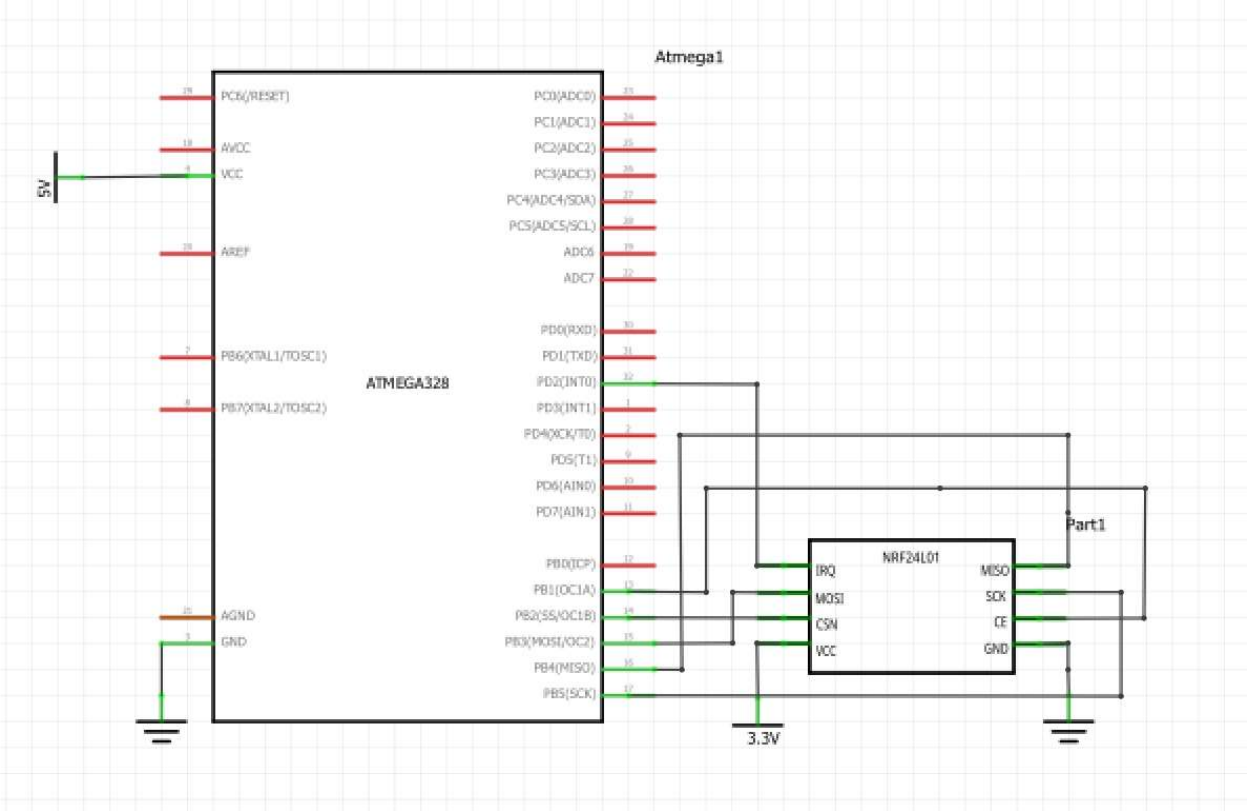
NO	SUBMISSION ITEM	COMPLETED (Y/N)	MARKS (/MAX)
1	COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS		
2.	INITIAL CODE OF TASK 1/A		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E		
4.	SCHEMATICS		
5.	SCREENSHOTS OF EACH TASK OUTPUT		
5.	SCREENSHOT OF EACH DEMO		
6.	VIDEO LINKS OF EACH DEMO		
7.	GOOGLECODE LINK OF THE DA		

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

LM34 Temperature Sensor
ATMega328P Xplained Mini x2
NRF24L01/+ x2



Block Diagram/Schematic for the Transmitter Circuit, the transmitter circuit includes one LM34 Temperature Sensor and one NRF24L01/+ module. We used the Xplained Mini's Micro-USB Serial Port for USART communication.



Above is the block diagram/schematic for the Receiver Circuit, the components used were one Xplained Mini Atmega328P board, and one NRF24L01/+ module.

2. TRANSMITTER CODE

```
/*
 * MT2_transmit.c
 *
 * Created: 4/24/2018 3:50:05 PM
 * Author : kiaer
 */
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include "nrf24l01.h"
#define BAUD 9600 //Baud Rate
#define MYUBRR F_CPU/16/BAUD-1 //calculate Baud

void setup_timer(void);
nRF24L01 *setup_rf(void);
volatile bool rf_interrupt = false;
volatile bool send_message = false;
```

```
void read_adc(void);  //Read ADC
void adc_init(void);  //initialize ADC
void USART_init( unsigned int ubrr ); //initialize USART
void USART_tx_string(char *data); //Print String USART
volatile unsigned int adc_temp;
char outs[256]; //array


volatile char received_data;


int main(void) {
	uint8_t to_address[5] = { 0x20, 0x30, 0x40, 0x51, 0x61 };
	bool on = false;
	adc_init(); // Initialize the ADC (Analog / Digital Converter)
	USART_init(MYUBRR); // Initialize the USART (RS232 interface)
	USART_tx_string("Connected!\r\n"); // shows theres a connection with USART
	_delay_ms(125); // wait a bit

	sei();
	nRF24L01 *rf = setup_rf();
	setup_timer();

	while (true) {
		if (rf_interrupt) {

			rf_interrupt = false;
			int success = nRF24L01_transmit_success(rf);
			if (success != 0)
			nRF24L01_flush_transmit_message(rf);

		}
		if (send_message) {
			send_message = false;
					on = !on;
			nRF24L01Message msg;
			read_adc();
			if (on){
				memcpy(msg.data, outs, 10); //sends ADC information through
RF.
				_delay_ms(100);
			}
			else memcpy(msg.data, "OFF", 4);

				snprintf(outs,sizeof(outs),"%3d F \r\n", adc_temp);// print it
				USART_tx_string(outs);

			msg.length = strlen((char *)msg.data) + 1;
			nRF24L01_transmit(rf, to_address, &msg);

		}
	}
	return 0;
}
nRF24L01 *setup_rf(void) {
	nRF24L01 *rf = nRF24L01_init();

rf->ss.port = &PORTB;
```

```
rf->ss.pin = PB2;
rf->ce.port = &PORTB;
rf->ce.pin = PB1;
rf->sck.port = &PORTB;
rf->sck.pin = PB5;
rf->mosi.port = &PORTB;
rf->mosi.pin = PB3;
rf->miso.port = &PORTB;
rf->miso.pin = PB4;
// interrupt on falling edge of INT0 (PD2)
EICRA |= _BV(ISC01);
EIMSK |= _BV(INT0);
nRF24L01_begin(rf);
return rf;
}
// setup timer to trigger interrupt every second when at 1MHz
void setup_timer(void) {
	TCCR1B |= _BV(WGM12);
	TIMSK1 |= _BV(OCIE1A);
	OCR1A = 15624;
	TCCR1B |= _BV(CS10) | _BV(CS11);
}
void adc_init(void) //initialize 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
	(1<<MUX1)| // ADC2 (PC2 PIN25)
	(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 Prescaler Select Bits
	(0<<ADPS1)|
	(1<<ADPS0);



}


void read_adc(void) {
	unsigned char i =4;
	adc_temp = 0; //initialize
	while (i--) {
		ADCSRA |= (1<<ADSC);
		while(ADCSRA & (1<<ADSC));
		adc_temp+= ADC;
		_delay_ms(50);
	}
	adc_temp = adc_temp / 4; // Average a few samples
```

```
}

/* INIT USART (RS-232) */
void USART_init( unsigned int ubrr ) {
	UBRR0H = (unsigned char)(ubrr>>8);
	UBRR0L = (unsigned char)ubrr;
	UCSR0B |= (1 << TXEN0) | (1 << RXEN0)| ( 1 << RXCIE0); // Enable receiver,
transmitter & RX interrupt
	UCSR0C |= (1<<UCSZ01) | (1 << UCSZ00);
}

void USART_tx_string( char *data ) {
	while ((*data != '\0')) {
		while (!(UCSR0A & (1 <<UDRE0)));
		UDR0 = *data;
		data++;
	}
}

// each one second interrupt
ISR(TIMER1_COMPA_vect) {
	send_message = true;
}
// nRF24L01 interrupt
ISR(INT0_vect) {
	rf_interrupt = true;
}
```

3. RECEIVER CODE

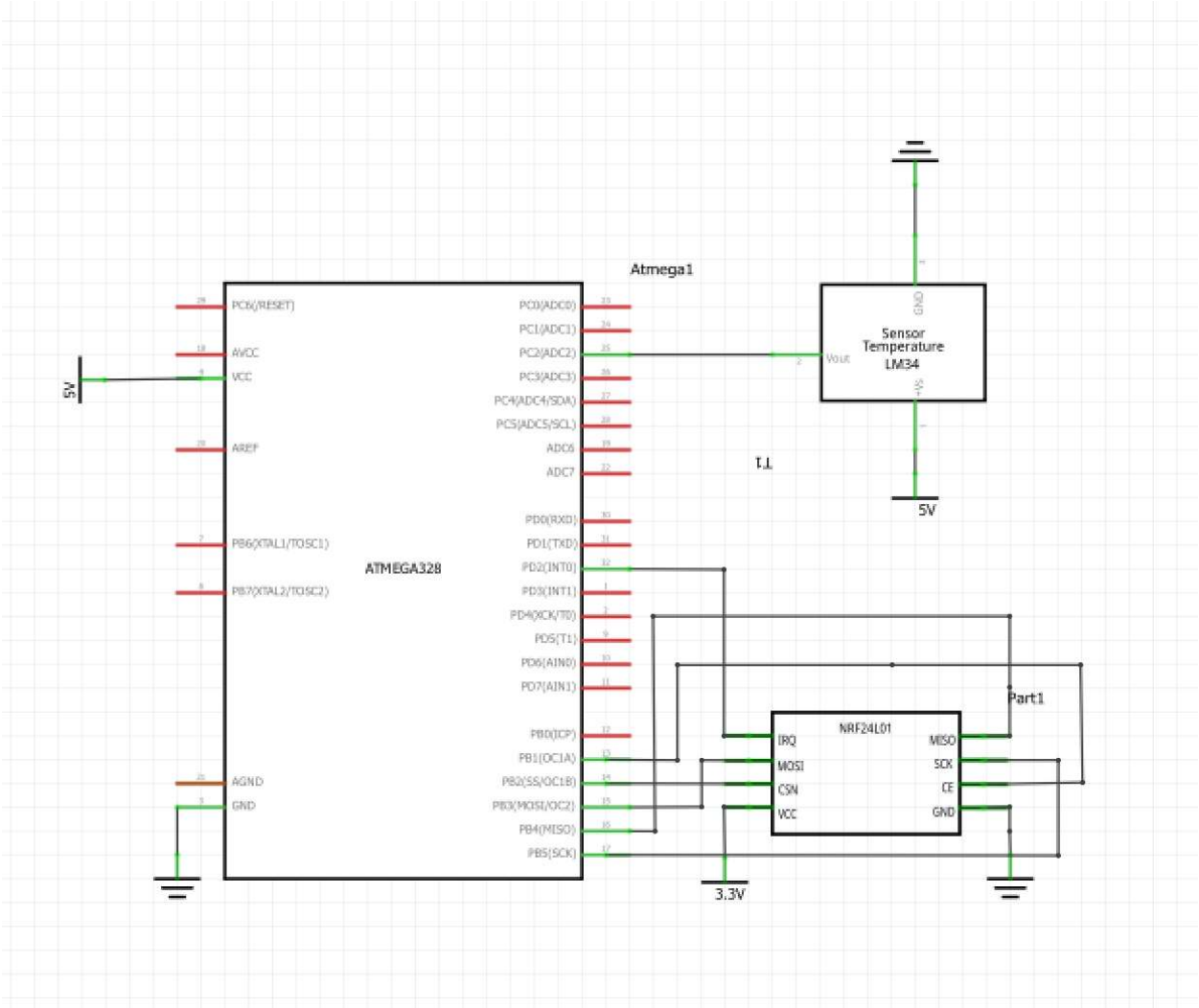
```
 * MT2_Receive.c
 *
 * Created: 4/24/2018 11:36:42 AM
 * Author : brian
 */

#define F_CPU 16000000UL //16MHz
#define BAUD 9600 //Baud Rate
#define MYUBRR F_CPU/16/BAUD-1 //calculate Baud

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <string.h>
#include <util/delay.h>
#include "nrf24l01.h"
#include "nrf24l01-mnemonics.h"

nRF24L01 *setup_rf(void);
void process_message(char *message);
inline void prepare_led_pin(void);
inline void set_led_high(void);
```

```
inline void set_led_low(void);
volatile bool rf_interrupt = false;


void read_adc(void);  //Read ADC
void adc_init(void);  //initialize ADC
void USART_init( unsigned int ubrr ); //initialize USART
void USART_tx_string(char *data); //Print String USART
volatile unsigned int adc_temp;
char outs[20]; //array


int main(void) {
	uint8_t address[5] = { 0x20, 0x30, 0x40, 0x51, 0x61 };
	prepare_led_pin();

		adc_init(); // Initialize the ADC (Analog / Digital Converter)
		USART_init(MYUBRR); // Initialize the USART (RS232 interface)
		USART_tx_string("Connected!\r\n"); // shows theres a connection with USART
		_delay_ms(125); // wait a bit


	sei();
	nRF24L01 *rf = setup_rf();
	nRF24L01_listen(rf, 0, address);
	uint8_t addr[5];
	nRF24L01_read_register(rf, CONFIG, addr, 1);
	while (true) {
		if (rf_interrupt) {
			rf_interrupt = false;
			while (nRF24L01_data_received(rf)) {
				nRF24L01Message msg;
				nRF24L01_read_received_data(rf, &msg);
				process_message((char *)msg.data);
				USART_tx_string(msg.data);
			}
			nRF24L01_listen(rf, 0, address);
		}
	}
	return 0;
}
nRF24L01 *setup_rf(void) {
	nRF24L01 *rf = nRF24L01_init();
	rf->ss.port = &PORTB;
	rf->ss.pin = PB2;
	rf->ce.port = &PORTB;
	rf->ce.pin = PB1;
	rf->sck.port = &PORTB;
	rf->sck.pin = PB5;
	rf->mosi.port = &PORTB;
	rf->mosi.pin = PB3;
	rf->miso.port = &PORTB;
	rf->miso.pin = PB4;
	// interrupt on falling edge of INT0 (PD2)
	EICRA |= _BV(ISC01);
	EIMSK |= _BV(INT0);
	nRF24L01_begin(rf);
	return rf;
```

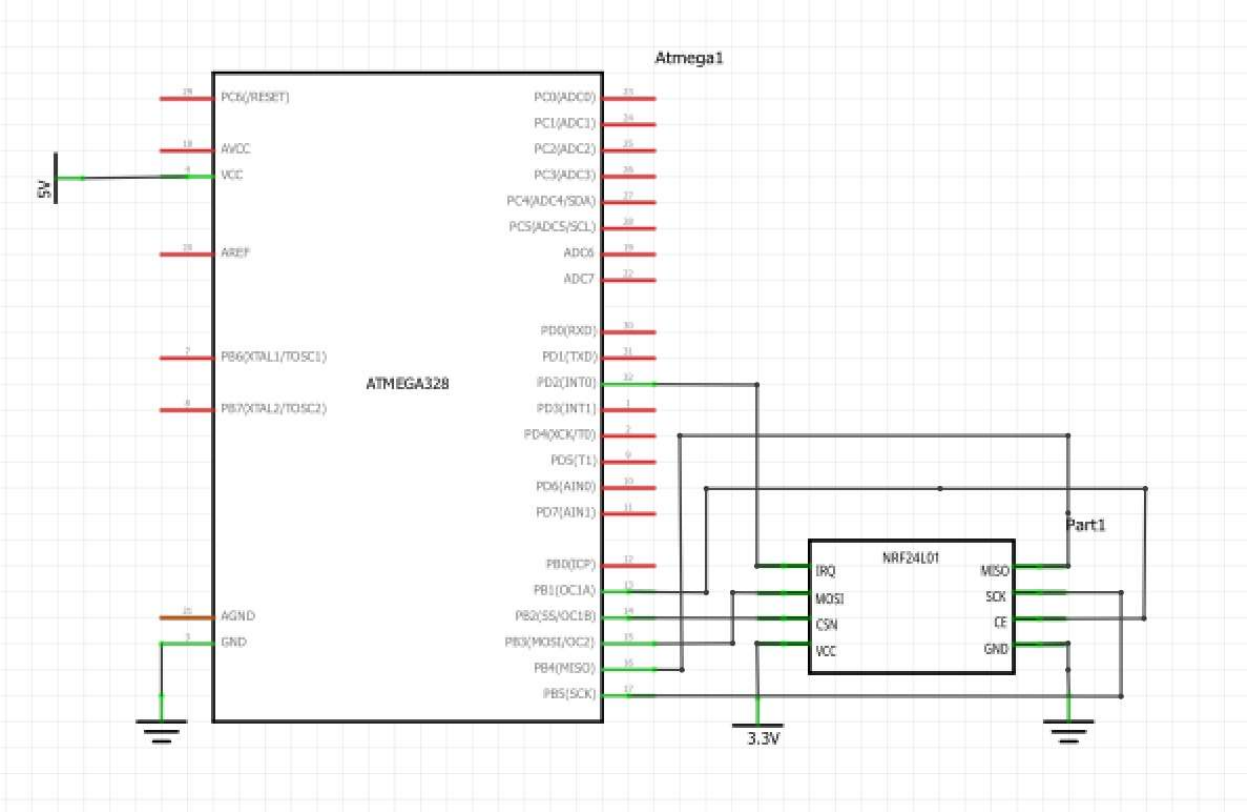
```
}

void process_message(char *message) {
	if (strcmp(message, "ON") == 0)
	set_led_high();
	else if (strcmp(message, "OFF") == 0)
	set_led_low();
}
inline void prepare_led_pin(void) {
	DDRB |= _BV(PB0);
	PORTB &= ~_BV(PB0);
}
inline void set_led_high(void) {
	PORTB |= _BV(PB0);
}
inline void set_led_low(void) {
	PORTB &= ~_BV(PB0);
}

void adc_init(void)
{

	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
	(1<<MUX1)| // ADC2 (PC2 PIN25)
	(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 Prescaler Select Bits
	(0<<ADPS1)|
	(1<<ADPS0);

}
void read_adc(void) {
	unsigned char i =4;
	adc_temp = 0; //initialize
	while (i--) {
		ADCSRA |= (1<<ADSC);
		while(ADCSRA & (1<<ADSC));
		adc_temp+= ADC;
		_delay_ms(50);
	}
	adc_temp = adc_temp / 4; // Average a few samples

}

/* INIT USART (RS-232) */
void USART_init( unsigned int ubrr ) {
	UBRR0H = (unsigned char)(ubrr>>8);
	UBRR0L = (unsigned char)ubrr;
```

```
        UCSR0B = (1 << TXEN0); // Enable receiver, transmitter & RX interrupt
        UCSR0C = (3 << UCSZ00); //asynchronous 8 N 1
}

void USART_tx_string( char *data ) {
        while ((*data != '\0')) {
                while (!(UCSR0A & (1 <<UDRE0)));
                UDR0 = *data;
                data++;
        }
}
// nRF24L01 interrupt
ISR(INT0_vect) {
        rf_interrupt = true;
}
```

4. SCHEMATICS

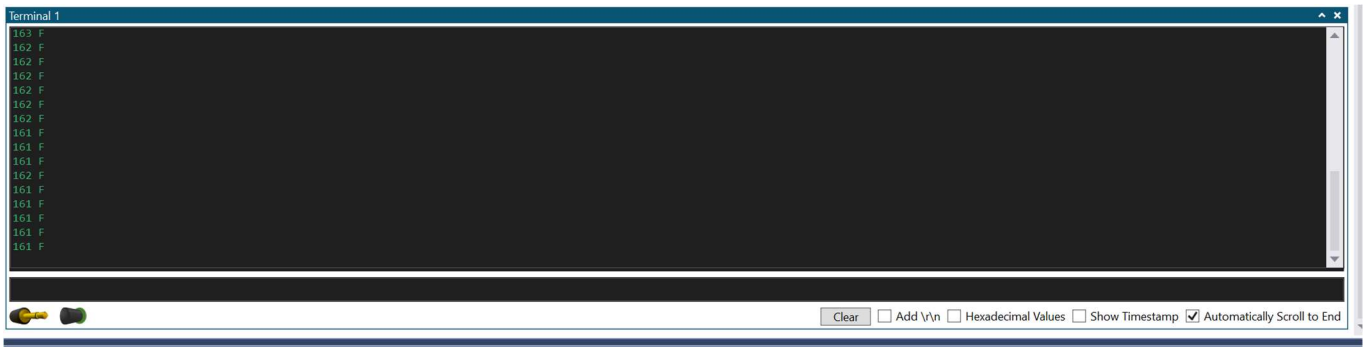


Transmitter Circuit that uses a LM34 Temperature Sensor and NRF240L module to transmit ADC values via radio frequency to a receiver.



Receiver circuit that is responsible for retrieving information that is being sent by the transmitter and displaying the output via USART.

5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)



Above is the output of the Receiver and the Transmitter they should be identical. A live video has been recorded below displaying identical values displayed via USART on two ATmega328P boards on the Data Visualizer through Atmel Studio.
https://youtu.be/ym67fLimz0E

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

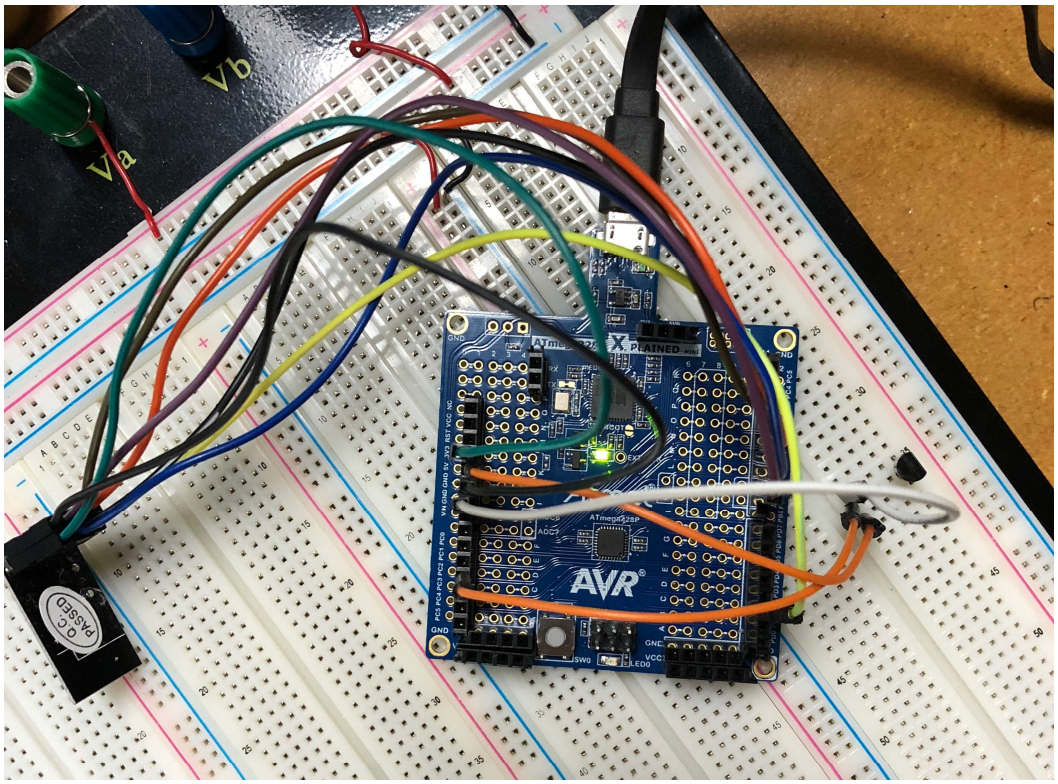


Figure 1. Transmitter Board Setup, using the Micro-USB mEDBG Virtual COM port to display USART values.

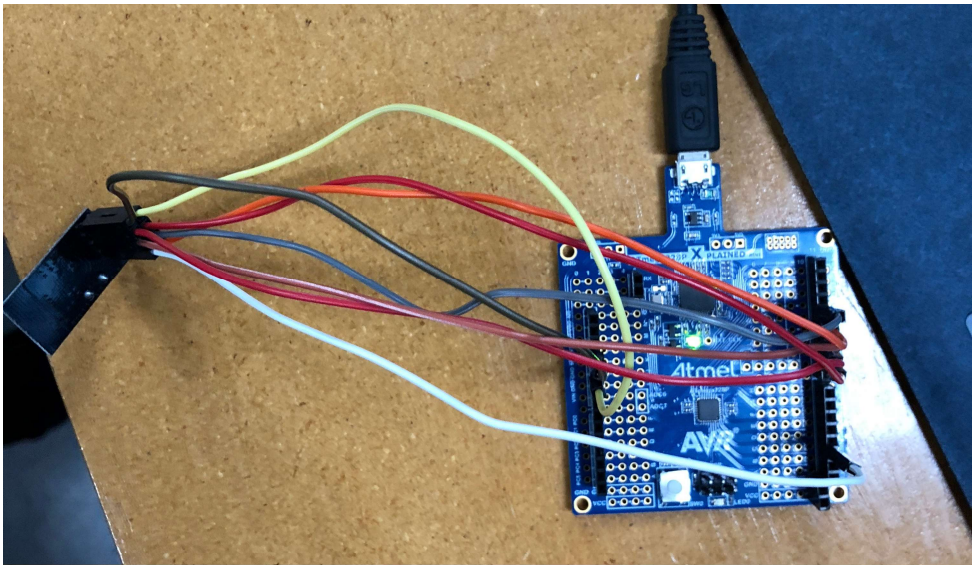


Figure 2. Receiver Board setup, also using the Micro-USB mEDBG Virtual COM port to display received values through USART on the data visualizer.

7. VIDEO LINKS OF EACH DEMO

https://youtu.be/ym67fLimz0E

8. GITHUB LINK OF THIS DA

https://github.com/bkiaer/MIDTERM2

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
Brian Medrano Kiaer