Phillip Sortomme

CPE301 – SPRING 2018

Midterm 2

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

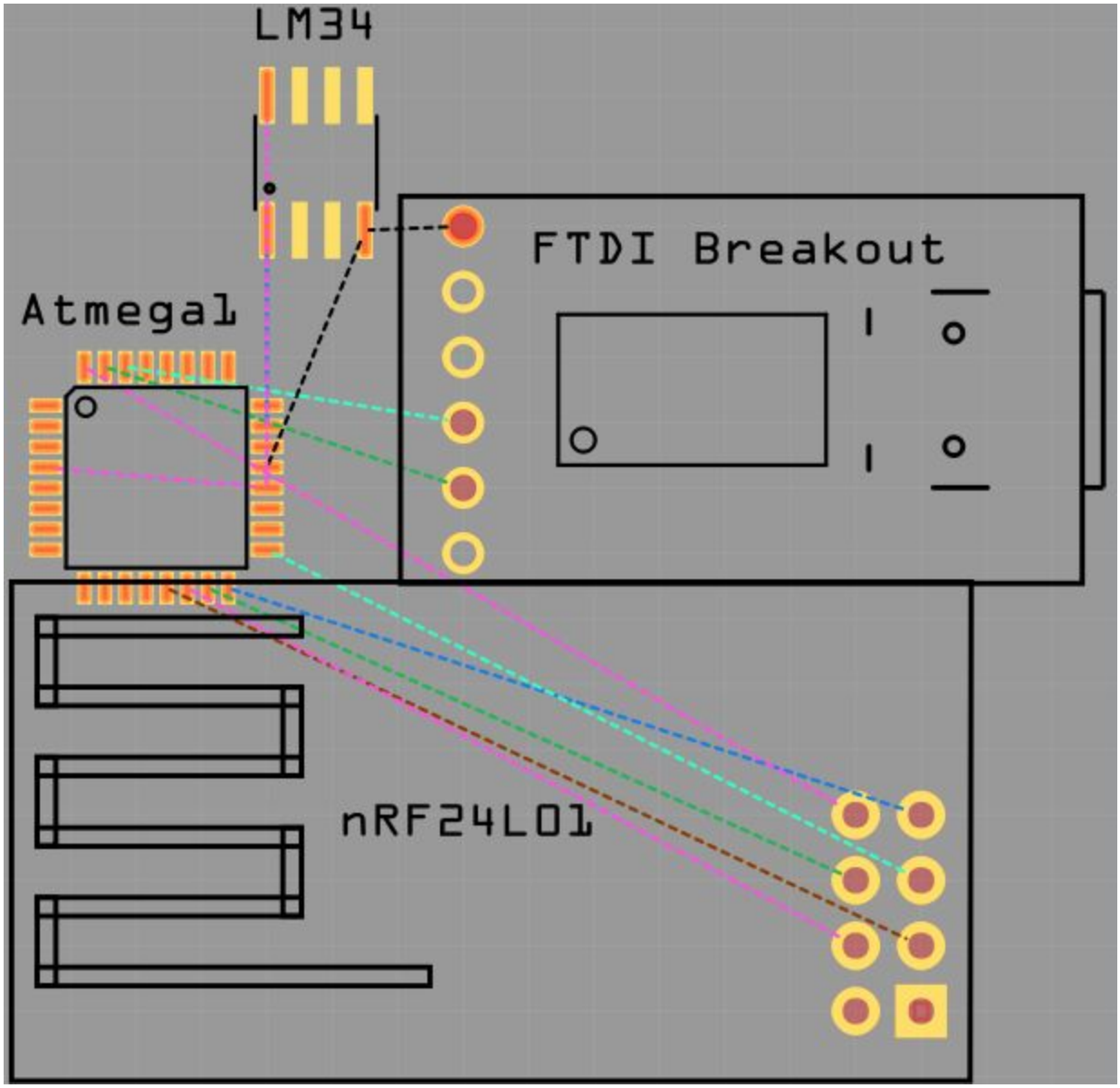
The student understands that all required components should be submitted in complete for grading of this assignment.

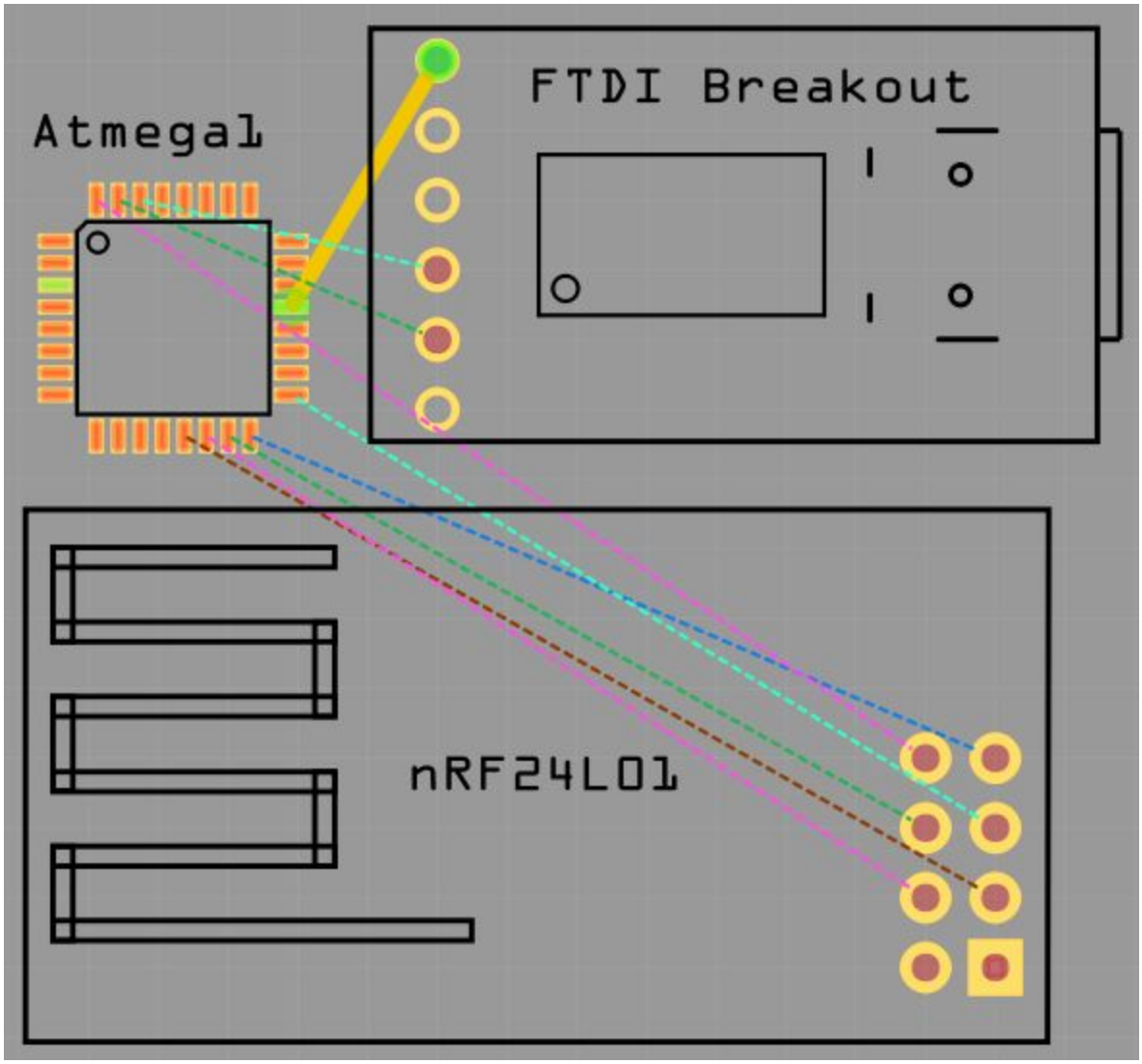
NO	SUBMISSION ITEM	COMPLETED (Y/N)	MARKS (/MAX)
1	COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS		
2.	INITIAL CODE OF TASK 1/A		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D		
3.	INCREMENTAL / DIFFERENTIAL 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		

COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

List of Components used
atmega328P
nRF24L01
FTDI Breakout
LM34 temperature sensor

Block diagram with pins used in the Atmega328P


LM34
FTDI Breakout
Atmega1
nRF24L01



INITIAL/DEVELOPED CODE OF TASK 1 (transmitter)

```
#define F_CPU 8000000              // Clock Speed
#define BAUD 9600                  // define BAUD
#define MYUBRR F_CPU/16/BAUD-1     // used to set UBRR0 to correct value

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

void setup_timer(void);
nRF24L01 *setup_rf(void);
void adc_init(void);
void USART_Init( unsigned int ubrr);
void USART_tx_string( char *data );

volatile bool rf_interrupt = false;
volatile bool send_message = false;
volatile unsigned int adc_temp;
```

```
char outs[20];

int main(void) {

   uint8_t to_address[5] = {0x08, 0xC1, 0x12, 0x33, 0xFC};
   bool on = false;
   USART_Init(MYUBRR);
   adc_init();
   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;
         if (on) {
         snprintf(outs, sizeof(outs),"%3d\r\n", adc_temp);// convert adc_temp from int to ascii
         memcpy(msg.data, outs, sizeof(outs));
         }
         else memcpy(msg.data, "OFF", 4);
         msg.length = strlen((char *)msg.data) + 1;
         nRF24L01_transmit(rf, to_address, &msg);
    USART_tx_string(outs);          // send the temperature data to serial output
      }
   }
   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);
}
// each one second interrupt
ISR(TIMER1_COMPA_vect) {
   send_message = true;
}
// nRF24L01 interrupt
ISR(INT0_vect) {
   rf_interrupt = true;
```

```
}

ISR(ADC_vect){
   unsigned  char i=4;
   adc_temp= 0;
   while (i--){
      adc_temp += (ADC+62);
      _delay_ms(50);
   ADCSRA |= (1<<ADSC);
   }
   adc_temp = adc_temp / 4;//    Average a   few sample
}

void adc_init(void){
   /** Setup   and enable  ADC **/
   ADMUX   =   (0<<REFS1)|//   Reference   Selection   Bits
   (1<<REFS0)|//   AVcc-external   cap at  AREF
   (0<<ADLAR)|//   ADC Left    Adjust  Result
   (0<<MUX3)|
   (0<<MUX2)|//    ANalogChannel   Selection   Bits
   (0<<MUX1)|//    ADC0    (PC0)
   (0<<MUX0);

   ADCSRA = (1<<ADEN)|//   ADC ENable
   (1<<ADSC)|//    ADC Start   Conversion
   (1<<ADATE)|//   ADC Auto    Trigger Enable
   (0<<ADIF)|//    ADC Interrupt   Flag
   (1<<ADIE)|//    ADC Interrupt   Enable
   (1<<ADPS2)|//   ADC PrescalerSelect Bits
   (1<<ADPS1)|
   (1<<ADPS0);
}

void USART_Init( unsigned int ubrr)
{
   /*Set baud rate */
   UBRR0H = (unsigned char)(ubrr>>8);
   UBRR0L = (unsigned char)ubrr;
   /*Enable receiver and transmitter */ // change to transmit only
   UCSR0B = (1<<RXEN0)|(1<<TXEN0);
   /* Set frame format: 8data, 1stop bit */
   UCSR0C = (1<<UCSZ00) | (1<<UCSZ01);
}

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

MODIFIED CODE OF TASK 1 (receiver)

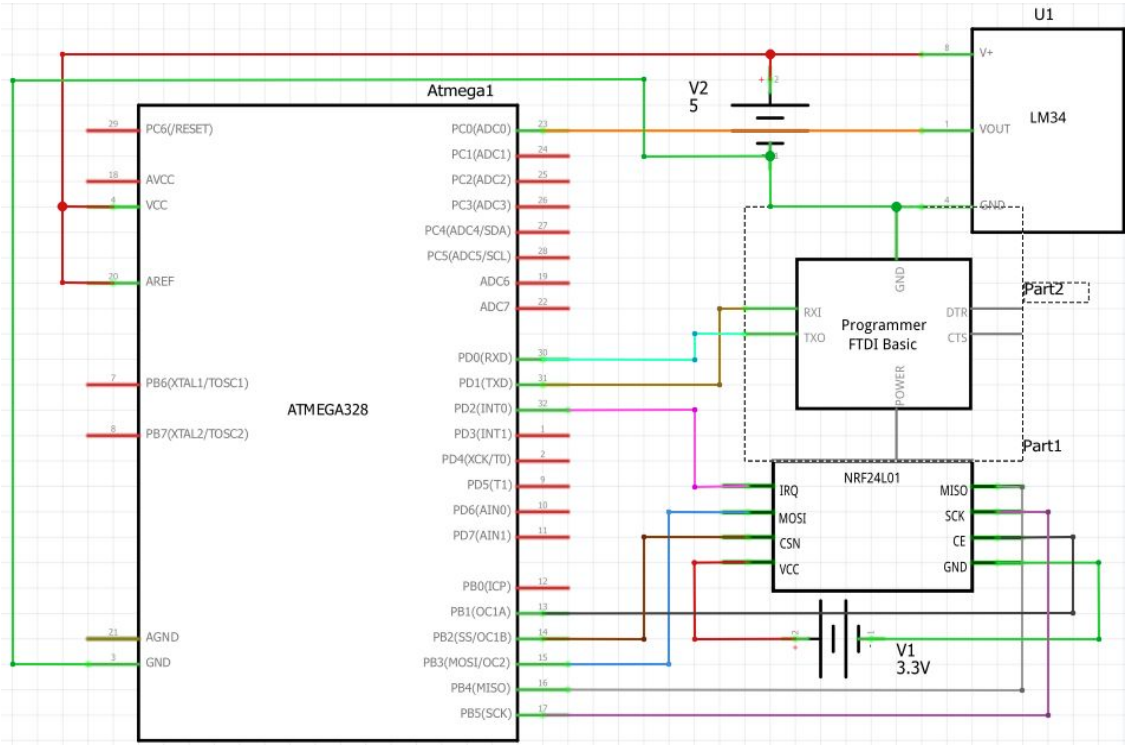
```
#define F_CPU 8000000UL            // Clock Speed
#define BAUD 9600                  // define BAUD
#define MYUBRR F_CPU/16/BAUD-1     // used to set UBRR0 to correct value
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <string.h>
#include <util/delay.h>
#include "nrf24l01.h"
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 USART_Init( unsigned int ubrr);
void USART_Transmit(unsigned char data);
void USART_tx_string(char * data);

int main(void) {
   uint8_t address[5] = {0x08, 0xC1, 0x12, 0x33, 0xFC};
   prepare_led_pin();
   USART_Init(MYUBRR);
   sei();
   nRF24L01 *rf = setup_rf();
   nRF24L01_listen(rf, 0, address);
   uint8_t addr[5];
   nRF24L01_read_register(rf, 0x00, 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((char *)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) {
```

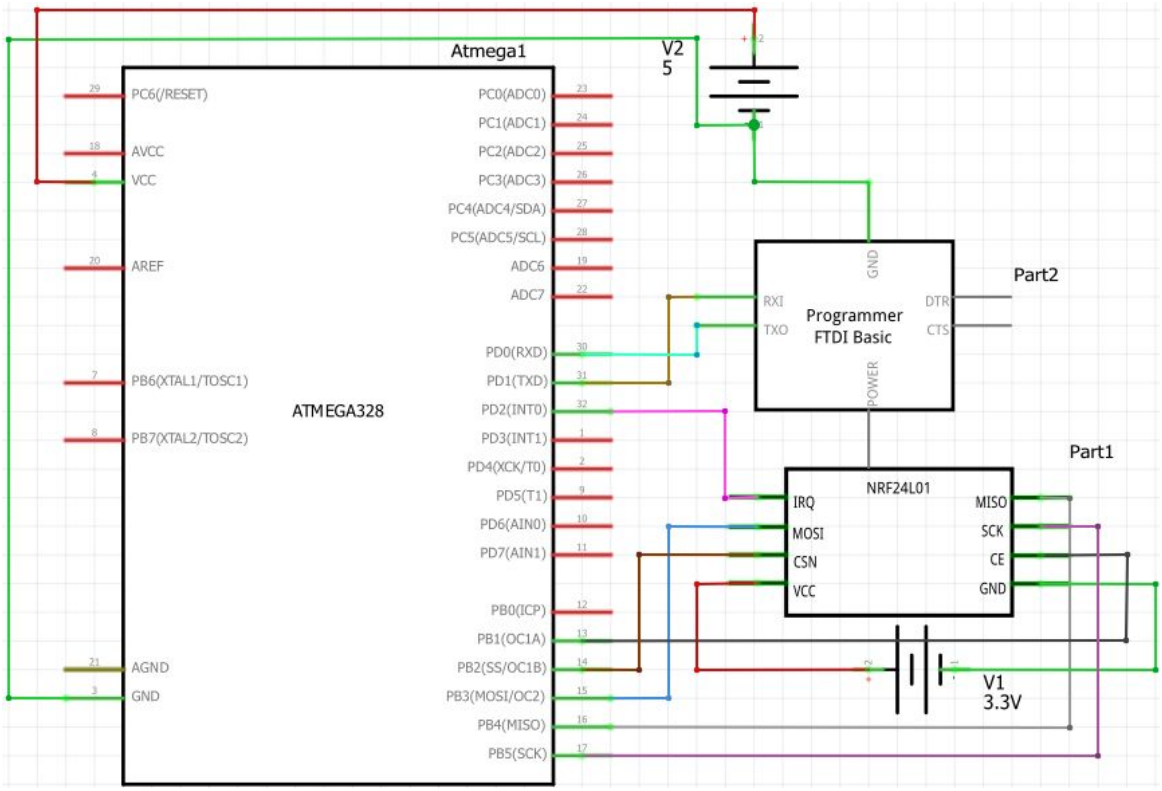
```
   PORTB &= ~_BV(PB0);
}
// nRF24L01 interrupt
ISR(INT0_vect) {
   USART_tx_string("test");
   rf_interrupt = true;
}

void USART_Init( unsigned int ubrr)
{
   /*Set baud rate */
   UBRR0H = (unsigned char)(ubrr>>8);
   UBRR0L = (unsigned char)ubrr;
   /*Enable receiver and transmitter */
   UCSR0B = (1<<RXEN0)|(1<<TXEN0);
   /* Set frame format: 8data, 1stop bit */
   UCSR0C = (1<<UCSZ00) | (1<<UCSZ01);
   }


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

SCHEMATICS

transmitter circuit



Receiver circuit



SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

SCREENSHOT OF EACH DEMO (BOARD SETUP)

VIDEO LINKS OF EACH DEMO

GITHUB LINK OF THIS DA

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

"*This assignment submission is my own, original work*".
Phillip Sortomme