CPE301 – SPRING 2019

Design Assignment 5A

Student Name: Saul Alejandro Mendoza Guzman
Student #: 2000540481
Student Email: mendos1@unlv.nevada.edu
Primary Github address: https://github.com/mendos1/subnission_da
Directory: DA5A

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

- Atmega328p x2
- Nrf24l01 + x2
- Wires
- Lm3x x2
- Atmel Studio 7

2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A

~~~~~~ MAIN.C ~~~~~~

```
//     Set clock frequency
#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
unsigned int ADC_TEMP;
//     Set up UART for printf();
#ifndef BAUD
#define BAUD 9600
#endif
#include "STDIO_UART.h"

//     Include nRF24L01+ library
#include "nrf24l01.h"
#include "nrf24l01-mnemonics.h"
#include "spi.h"
void print_config(void);
void ADC_INIT(void);
void READ_ADC(void);

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

int main(void){
       //     Set cliche message to send (message cannot exceed 32 characters)
       char tx_message[32];                    // Define string array
       char *tx_ptr = tx_message;
       strcpy(tx_message,"Hi :) !");      // Copy string into array

       //     Initialize UART
       uart_init();

       //     Initialize nRF24L01+ and print configuration info
    nrf24_init();
       print_config();

       ADC_INIT();
       //     Start listening to incoming messages
       printf("start listening\n");
       nrf24_start_listening();
       printf("Done listening\n");

       ADC_TEMP = 0;
```
~~~~~~

```
    while (1){
                    //printf("outside if\n");
                    READ_ADC();
                    tx_ptr = tx_message;
                    //sprintf(tx_ptr, "%d", ADC_TEMP);
                    nrf24_send_message(tx_ptr);

                    delay_ms(100);
                    //continue;
                    //;
            if (message_received){
                    printf("inside if condition\n");
                    //     Message received, print it
                    message_received = false;
                    printf("Received message: %s\n",nrf24_read_message());
                    //     Send message as response
                    _delay_ms(500);
                    status = nrf24_send_message(tx_message);
                    if (status == true) printf("Message sent successfully\n");
            }
    }
}

//     Interrupt on IRQ pin
ISR(INT0_vect) {
       message_received = true;
}

void ADC_INIT(void){

       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)| //
       (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);

       // Timer/Counter1 Interrupt Mask Register
       TIMSK1 |= (1<<TOIE1); // enable overflow interrupt
       TCCR1B |= (1<<CS12)|(1<<CS10); // clock
       TCNT1 = 49911; //((16MHz/1024)*1)-1 = 15624

}
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/8 ; // Average
```

```
}

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%02X\n",data);
	nrf24_read(EN_AA,&data,1);
	printf("EN_AA			0x%02X\n",data);
	nrf24_read(EN_RXADDR,&data,1);
	printf("EN_RXADDR		0x%02X\n",data);
	nrf24_read(SETUP_RETR,&data,1);
	printf("SETUP_RETR		0x%02X\n",data);
	nrf24_read(RF_CH,&data,1);
	printf("RF_CH			0x%02X\n",data);
	nrf24_read(RF_SETUP,&data,1);
	printf("RF_SETUP		0x%02X\n",data);
	nrf24_read(STATUS,&data,1);
	printf("STATUS					0x%02X\n",data);
	nrf24_read(FEATURE,&data,1);
	printf("FEATURE					0x%02X\n",data);
	printf("-------------------------------------------\n\n");
}
```

~~~~~~

| THESE ARE THE LIBRARY FUNCTIONS |

~~~~~~ NRF24L01.C ~~~~~~

```
// MIT License
//
// Copyright (c) 2018 Helvijs Adams
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in all
// copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
// SOFTWARE.

// Set clock frequency
#ifndef F_CPU
#define F_CPU 16000000UL
#endif

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

```
// nRF24L01+ include files
#include "nrf24l01.h"
#include "nrf24l01-mnemonics.h"
#include "spi.h"

// Settings
uint8_t rx_address[5] = { 0x73, 0x73, 0x73, 0x73, 0x73 };     // Read pipe address
uint8_t tx_address[5] = { 0x42, 0x42, 0x42, 0x42, 0x42 };     // Write pipe address
#define READ_PIPE           0                                                   // Number of
read pipe
//
// -AUTO_ACK can be disabled when running on 2MBPS @ <= 32 byte messages.
// -250KBPS and 1MBPS with AUTO_ACK disabled lost many packets
// if the packet size was bigger than 4 bytes.
// -If AUTO_ACK is enabled, tx_address = rx_address.
//
#define AUTO_ACK     false         // Auto acknowledgment
#define DATARATE     RF_DR_2MBPS   // 250kbps, 1mbps, 2mbps
#define POWER        POWER_MAX     // Set power (MAX 0dBm..HIGH -6dBm..LOW -12dBm.. MIN -18dBm)
#define CHANNEL                    0x74                                        // 2.4GHz-
2.5GHz channel selection (0x01 - 0x7C)
#define DYN_PAYLOAD         true   // Dynamic payload enabled
#define CONTINUOUS          false  // Continuous carrier transmit mode (not tested)
//
// ISR(INT0_vect) is triggered depending on config (only one can be true)
//
#define RX_INTERRUPT true   // Interrupt when message is received (RX)
#define TX_INTERRUPT false  // Interrupt when message is sent (TX)
#define RT_INTERRUPT false  // Interrupt when maximum re-transmits are reached (MAX_RT)
//
// -PIN map.
// -If CE or CSN is changed to different PIN e.g. PC0
// then change DDRB -> DDRC, PORTB -> PORTC and so on
//
// CE
#define CE_DDR              DDRB
#define CE_PORT             PORTB
#define CE_PIN              DDB1                                                // CE
connected to PB1
// CSN
#define CSN_DDR             DDRB
#define CSN_PORT     PORTB
#define CSN_PIN             DDB2                                                // CSN
connected to PB2
// IRQ
#define IRQ_DDR             DDRD
#define IRQ_PORT     PORTD
#define IRQ_PIN             DDD2                                                // IRQ
connected to PD2
// MOSI
#define MOSI_DDR     DDRB
#define MOSI_PORT    PORTB
#define MOSI_PIN     DDB3
// MISO
#define MISO_DDR     DDRB
#define MISO_PORT    PORTB
#define MISO_PIN     DDB4
// SCK
#define SCK_DDR             DDRB
#define SCK_PORT     PORTB
#define SCK_PIN             DDB5

// PIN toggling
#define setbit(port, bit) (port) |= (1 << (bit))
```

```
#define clearbit(port, bit) (port) &= ~(1 << (bit))
#define ce_low clearbit(CE_PORT,CE_PIN)
#define ce_high setbit(CE_PORT,CE_PIN)
#define csn_low clearbit(CSN_PORT,CSN_PIN)
#define csn_high setbit(CSN_PORT,CSN_PIN)

// Used to store SPI commands
uint8_t data;

uint8_t nrf24_send_spi(uint8_t register_address, void *data, unsigned int bytes)
{
	uint8_t status;
	csn_low;
	status = spi_exchange(register_address);
	for (unsigned int i = 0; i < bytes; i++)
	((uint8_t*)data)[i] = spi_exchange(((uint8_t*)data)[i]);
	csn_high;
	return status;
}

uint8_t nrf24_write(uint8_t register_address, uint8_t *data, unsigned int bytes)
{
	return nrf24_send_spi(W_REGISTER | register_address, data, bytes);
}

uint8_t nrf24_read(uint8_t register_address, uint8_t *data, unsigned int bytes)
{
	return nrf24_send_spi(R_REGISTER | register_address, data, bytes);
}

void nrf24_init(void)
{
	// Interrupt on falling edge of INT0 (PD2) from IRQ pin
	cli();                              // Disable interrupts
	EICRA |= (1 << ISC01);
	EIMSK |= (1 << INT0);
	sei();                              // Enable interrupts

	// CSN and CE as outputs and initial states
	setbit(CE_DDR,CE_PIN);
	setbit(CSN_DDR,CSN_PIN);
	csn_high;
	ce_low;

	// Initialize SPI
	spi_master_init();
	_delay_ms(100);                         // Power on reset 100ms

	// Start nRF24L01+ config
	data =
	(!(RX_INTERRUPT) << MASK_RX_DR) |  // IRQ interrupt on RX (0 = enabled)
	(!(TX_INTERRUPT) << MASK_TX_DS) |  // IRQ interrupt on TX (0 = enabled)
	(!(RT_INTERRUPT) << MASK_MAX_RT) |// IRQ interrupt on auto retransmit counter overflow (0 =
enabled)
	(1 << EN_CRC) |                                 // CRC enable
	(1 << CRC0) |                                   // CRC scheme
	(1 << PWR_UP) |                                 // Power up
	(1 << PRIM_RX);                                 // TX/RX select
	nrf24_write(CONFIG,&data,1);

	// Auto-acknowledge on all pipes
	data =
	(AUTO_ACK << ENAA_P5) |
	(AUTO_ACK << ENAA_P4) |
```

```
        (AUTO_ACK << ENAA_P3) |
        (AUTO_ACK << ENAA_P2) |
        (AUTO_ACK << ENAA_P1) |
        (AUTO_ACK << ENAA_P0);
        nrf24_write(EN_AA,&data,1);

        // Set retries
        data = 0xF0;         // Delay 4000us with 1 re-try (will be added in settings)
        nrf24_write(SETUP_RETR,&data,1);

        // Disable RX addresses
        data = 0;
        nrf24_write(EN_RXADDR, &data, 1);

        // Set channel
        data = CHANNEL;
        nrf24_write(RF_CH,&data,1);

        // Setup
        data =
        (CONTINUOUS << CONT_WAVE) |                     // Continuous carrier transmit
        ((DATARATE >> RF_DR_HIGH) << RF_DR_HIGH) |      // Data rate
        ((POWER >> RF_PWR) << RF_PWR);                  // PA level
        nrf24_write(RF_SETUP,&data,1);

        // Status - clear TX/RX FIFO's and MAX_RT by writing 1 into them
        data =
        (1 << RX_DR) |              // RX FIFO
        (1 << TX_DS) |              // TX FIFO
        (1 << MAX_RT);              // MAX RT
        nrf24_write(STATUS,&data,1);

        // Dynamic payload on all pipes
        data =
        (DYN_PAYLOAD << DPL_P0) |
        (DYN_PAYLOAD << DPL_P1) |
        (DYN_PAYLOAD << DPL_P2) |
        (DYN_PAYLOAD << DPL_P3) |
        (DYN_PAYLOAD << DPL_P4) |
        (DYN_PAYLOAD << DPL_P5);
        nrf24_write(DYNPD, &data,1);

        // Enable dynamic payload
        data =
        (DYN_PAYLOAD << EN_DPL) |
        (AUTO_ACK << EN_ACK_PAY) |
        (AUTO_ACK << EN_DYN_ACK);
        nrf24_write(FEATURE,&data,1);

        // Flush TX/RX
        // Clear RX FIFO which will reset interrupt
        uint8_t data = (1 << RX_DR) | (1 << TX_DS) | (1 << MAX_RT);
        nrf24_write(FLUSH_RX,0,0);
        nrf24_write(FLUSH_TX,0,0);

        // Open pipes
        nrf24_write(RX_ADDR_P0 + READ_PIPE,rx_address,5);
        nrf24_write(TX_ADDR,tx_address,5);
        nrf24_write(EN_RXADDR,&data,1);
        data |= (1 << READ_PIPE);
        nrf24_write(EN_RXADDR,&data,1);
}

void nrf24_write_ack(void)
```

```
{
	const void *ack = "A";
	unsigned int length = 1;
	csn_low;
	spi_send(W_ACK_PAYLOAD);
	while (length--) spi_send(*(uint8_t *)ack++);
	csn_high;
}

void nrf24_state(uint8_t state)
{
	uint8_t config_register;
	nrf24_read(CONFIG,&config_register,1);

	switch (state)
	{
		case POWERUP:
		// Check if already powered up
		if (!(config_register & (1 << PWR_UP)))
		{
			data = config_register | (1 << PWR_UP);
			nrf24_write(CONFIG,&data,1);
			// 1.5ms from POWERDOWN to start up
			_delay_ms(2);
		}
		break;
		case POWERDOWN:
		data = config_register & ~(1 << PWR_UP);
		nrf24_write(CONFIG,&data,1);
		break;
		case RECEIVE:
		data = config_register | (1 << PRIM_RX);
		nrf24_write(CONFIG,&data,1);
		// Clear STATUS register
		data = (1 << RX_DR) | (1 << TX_DS) | (1 << MAX_RT);
		nrf24_write(STATUS,&data,1);
		break;
		case TRANSMIT:
		data = config_register & ~(1 << PRIM_RX);
		nrf24_write(CONFIG,&data,1);
		break;
		case STANDBY1:
		ce_low;
		break;
		case STANDBY2:
		data = config_register & ~(1 << PRIM_RX);
		nrf24_write(CONFIG,&data,1);
		ce_high;
		_delay_us(150);
		break;
	}
}

void nrf24_start_listening(void)
{
	nrf24_state(RECEIVE);                    // Receive mode
	//if (AUTO_ACK) nrf24_write_ack(); // Write acknowledgment
	ce_high;
	_delay_us(150);                          // Settling time
}

uint8_t nrf24_send_message(const void *tx_message)
{
	// For printf();
```

```
	char temp[32];
	memset(temp,0,32);
	strcpy(temp,tx_message);

	// Message length
	uint8_t length = strlen(tx_message);

	// Transmit mode
	nrf24_state(TRANSMIT);

	// Flush TX/RX and clear TX interrupt
	nrf24_write(FLUSH_RX,0,0);
	nrf24_write(FLUSH_TX,0,0);
	data = (1 << TX_DS);
	nrf24_write(STATUS,&data,1);

	// Disable interrupt on RX
	nrf24_read(CONFIG,&data,1);
	data |= (1 << MASK_RX_DR);
	nrf24_write(CONFIG,&data,1);

	// Start SPI, load message into TX_PAYLOAD
	csn_low;
	if (AUTO_ACK) spi_send(W_TX_PAYLOAD);
	else spi_send(W_TX_PAYLOAD_NOACK);
	while (length--) spi_send(*(uint8_t *)tx_message++);
	spi_send(0);
	csn_high;

	// Send message by pulling CE high for more than 10us
	ce_high;
	_delay_us(15);
	ce_low;

	// Wait for message to be sent (TX_DS flag raised)
	nrf24_read(STATUS,&data,1);
	while(!(data & (1 << TX_DS))) nrf24_read(STATUS,&data,1);
	printf("Message sent: %s\n",temp);

	// Enable interrupt on RX
	nrf24_read(CONFIG,&data,1);
	data &= ~(1 << MASK_RX_DR);
	nrf24_write(CONFIG,&data,1);

	// Continue listening
	nrf24_start_listening();

	return 1;
}

unsigned int nrf24_available(void)
{
	uint8_t config_register;
	nrf24_read(FIFO_STATUS,&config_register,1);
	if (!(config_register & (1 << RX_EMPTY))) return 1;
	return 0;
}

const char * nrf24_read_message(void)
{
	// Message placeholder
	static char rx_message[32];
	memset(rx_message,0,32);
```

```
        // Write ACK message
        if (AUTO_ACK) nrf24_write_ack();

        // Get length of incoming message
        nrf24_read(R_RX_PL_WID,&data,1);

        // Read message
        if (data > 0) nrf24_send_spi(R_RX_PAYLOAD,&rx_message,data+1);

        // Check if there is message in array
        if (strlen(rx_message) > 0)
        {
                // Clear RX interrupt
                data = (1 << RX_DR);
                nrf24_write(STATUS,&data,1);

                return rx_message;
        }

        // Clear RX interrupt
        data = (1 << RX_DR);
        nrf24_write(STATUS,&data,1);

        return "failed";
}
```

~~~~~~ NRF24L01.H ~~~~~~

```
// MIT License
//
// Copyright (c) 2018 Helvijs Adams
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in all
// copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
// SOFTWARE.

#ifndef _NRF24L01_H
#define _NRF24L01_H

//      States
#define POWERUP      1
#define POWERDOWN    2
#define RECEIVE      3
#define TRANSMIT     4
#define STANDBY1     5
#define STANDBY2     6

//      Forward declarations
uint8_t nrf24_send_spi(uint8_t register_address, void *data, unsigned int bytes);
uint8_t nrf24_write(uint8_t register_address, uint8_t *data, unsigned int bytes);
uint8_t nrf24_read(uint8_t register_address, uint8_t *data, unsigned int bytes);
```
~~~~~~

```
void nrf24_init(void);
void nrf24_state(uint8_t state);
void nrf24_start_listening(void);
unsigned int nrf24_available(void);
const char * nrf24_read_message(void);
uint8_t nrf24_send_message(const void *tx_message);

#endif /*_NRF24L01_H*/
```

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ NRF24101-MNEMONICS.H ~~~~~~

```
/*  Copyright (c) 2015 B. Sidhipong <bsidhipong@gmail.com>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>. */

#ifndef __NORDIC_NRF24L01_RADIO_H__
#define __NORDIC_NRF24L01_RADIO_H__
#include <avr/common.h>

/* SPI commands */
#define REGISTER_MASK       0b00011111
#define R_REGISTER          0b00000000 /* 000A AAAA | AAAAA = 5-bit register map address */
#define W_REGISTER          0b00100000 /* 001A AAAA */
#define R_RX_PAYLOAD        0b01100001
#define W_TX_PAYLOAD        0b10100000
#define FLUSH_TX            0b11100001
#define FLUSH_RX            0b11100010
#define REUSE_TX_PL         0b11100011
#define ACTIVATE            0b01010000
#define R_RX_PL_WID         0b01100000
#define ACK_PAYLOAD_MASK    0b00000111
#define W_ACK_PAYLOAD       0b10101000 /* 1010 1PPP | PPP = pipe number */
#define W_TX_PAYLOAD_NOACK  0b10110000
#define NOP                 0b11111111

/* CONFIG: configuration register */
#define CONFIG              0x00
#define MASK_RX_DR          6
#define MASK_TX_DS          5
#define MASK_MAX_RT         4
#define EN_CRC              3
#define CRC0                2
#define PWR_UP              1
#define PRIM_RX             0

/* EN_AA: Enhanced ShockBurst™
 * Enable 'Auto Acknowledgment' function.  Disable this functionality
 * to be compatible with nRF2401. */
#define EN_AA               0x01
#define ENAA_P5             5
#define ENAA_P4             4
#define ENAA_P3             3
#define ENAA_P2             2
#define ENAA_P1             1
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
#define ENAA_P0             0
/* EN_RXADDR: enable RX addresses */
#define EN_RXADDR           0x02
#define ERX_P5              5
#define ERX_P4              4
#define ERX_P3              3
#define ERX_P2              2
#define ERX_P1              1
#define ERX_P0              0

/* SETUP_AW: seup of address widths */
#define SETUP_AW            0x03
#define AW                  0 /* 1:0 */

/* SETUP_RETR: setup of automatic retransmission */
#define SETUP_RETR          0x04
#define ARD                 4 /* 7:4 */
#define ARC                 0 /* 3:0 */

#define RF_CH               0x05

/* RF_SETUP: RF setup register */
#define RF_SETUP            0x06
#define CONT_WAVE           7
#define RF_DR_LOW           5
#define PLL_LOCK            4
#define RF_DR_HIGH          3       /* RF_DR:RF_DR_LOW = RF data rate */
#define RF_DR_250KBPS       0b00100000    /* available on nRF24L01+ */
#define RF_DR_1MBPS         0b00000000
#define RF_DR_2MBPS         0b00001000
#define RF_PWR              1             /* 2:1 */
#define POWER_MIN           0b00000000
#define POWER_LOW           0b00000010
#define POWER_HIGH          0b00000100
#define POWER_MAX           0b00000110

/* STATUS: status register */
#define STATUS              0x07
#define RX_DR               6
#define TX_DS               5
#define MAX_RT              4
#define RX_P_NO             1       /* 3:1 */
#define TX_FULL             0

/* OBSERVE_TX: transmit observe register */
#define OBSERVE_TX          0x08
#define PLOS_CNT            4       /* 7:4 */
#define ARC_CNT             0       /* 3:0 */

#define RPD                 0x09
#define RX_ADDR_P0          0x0A
#define RX_ADDR_P1          0x0B
#define RX_ADDR_P2          0x0C
#define RX_ADDR_P3          0x0D
#define RX_ADDR_P4          0x0E
#define RX_ADDR_P5          0x0F
#define TX_ADDR             0x10
#define RX_PW_P0            0x11
#define RX_PW_P1            0x12
#define RX_PW_P2            0x13
#define RX_PW_P3            0x14
#define RX_PW_P4            0x15
#define RX_PW_P5            0x16
```

```
/* FIFO_STATUS: FIFO status register */
#define FIFO_STATUS         0x17
#define TX_REUSE            6
#define FIFO_FULL           5
#define TX_EMPTY            4
#define RX_FULL             1
#define RX_EMPTY            0

/* DYNPD: enable dynamic payload length */
#define DYNPD               0x1C
#define DPL_P5              5
#define DPL_P4              4
#define DPL_P3              3
#define DPL_P2              2
#define DPL_P1              1
#define DPL_P0              0

/* FEATURE: */
#define FEATURE             0x1D
#define EN_DPL              2
#define EN_ACK_PAY          1
#define EN_DYN_ACK          0

#endif /* __NORDIC_NRF24L01_RADIO_H__ */
```

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ SPI.C ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
/*  Copyright (c) 2015 B. Sidhipong <bsidhipong@gmail.com>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>. */

#include <avr/common.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include "spi.h"

#define DDR_SPI             DDRB
#define DD_MOSI             DDB3
#define DD_MISO             DDB4
#define DD_SCK              DDB5

void spi_master_init( void )
{
       DDR_SPI &= ~_BV(DD_MISO);
       DDR_SPI |= (_BV(DD_MOSI) | _BV(DD_SCK));
       /* 19.5.1 SPCR – SPI Control Register
        *
        * • Bit 7 – SPIE: SPI Interrupt Enable
        * This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and
the if the Global
        * Interrupt Enable bit in SREG is set.
        *
        * • Bit 6 – SPE: SPI Enable
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
        * When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any
SPI operations.
        *
        * • Bit 5 – DORD: Data Order
        * When the DORD bit is written to one, the LSB of the data word is transmitted first.
        * When the DORD bit is written to zero, the MSB of the data word is transmitted first.
        *
        * • Bit 4 – MSTR: Master/Slave Select
        * This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic
zero. If SS is
        * configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in
SPSR will become
        * set. The user will then have to set MSTR to re-enable SPI Master mode.
        *
        * • Bit 3 – CPOL: Clock Polarity
        * When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is
low when idle.
        *
        * • Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0
        * These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have
no effect on
        * the Slave. The relationship between SCK and the Oscillator Clock frequency fosc is shown in
the following table:
        *
        * Table 19-5 Relationship Between SCK and Oscillator Frequency
        * SPI2X | SPR1 | SPR0 | SCK Frequency
        * ===================================
        *   0   |   0  |   0  | fosc/4
        *   0   |   0  |   1  | fosc/16
        *   0   |   1  |   0  | fosc/64
        *   0   |   1  |   1  | fosc/128
        *   1   |   0  |   0  | fosc/2
        *   1   |   0  |   1  | fosc/8
        *   1   |   1  |   0  | fosc/32
        *   1   |   1  |   1  | fosc/64
        */
       SPCR = (0 << SPIE) |
              (1 << SPE)  |
              (0 << DORD) |
              (1 << MSTR) |
              (0 << CPOL) |
              (0 << SPR1) | (0 << SPR0);
       /* 19.5.2   SPSR – SPI Status Register
        *
        * • Bit 7 – SPIF: SPI Interrupt Flag
        * When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE
in SPCR is set and
        * global interrupts are enabled. If SS is an input and is driven low when the SPI is in Master
mode, this will also
        * set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt
handling vector.
        * Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF
set, then accessing the
        * SPI Data Register (SPDR).
        *
        * • Bit 0 – SPI2X: Double SPI Speed Bit
        * When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI
is in Master
        * mode (see Table 19-5). This means that the minimum SCK period will be two CPU clock periods.
When the SPI
        * is configured as Slave, the SPI is only guaranteed to work at fosc/4 or lower. */
       SPSR |= _BV(SPI2X);
}
```

```
void spi_bulk_send( uint8_t *send_buffer, uint8_t count )
{
	while ( count-- ) {
		SPDR = *send_buffer++;
		loop_until_bit_is_set(SPSR, SPIF);
	}
}

void spi_send( uint8_t send_data )
{
	SPDR = send_data;
	loop_until_bit_is_set(SPSR, SPIF);
}

void spi_bulk_exchange( uint8_t *send_buffer, uint8_t *receive_buffer, uint8_t count )
{
	while ( count-- ) {
		SPDR = *send_buffer++;
		loop_until_bit_is_set(SPSR, SPIF);
		*receive_buffer++ = SPDR;
	}
}

uint8_t spi_exchange( uint8_t send_data )
{
	SPDR = send_data;
	loop_until_bit_is_set(SPSR, SPIF);
	return SPDR;
}
```

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ SPI.H ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
/*  Copyright (c) 2015 B. Sidhipong <bsidhipong@gmail.com>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>. */

#ifndef __SMALL_SPI_H__
#define __SMALL_SPI_H__
#include <avr/common.h>

void spi_master_init( void );
void spi_bulk_send( uint8_t *send_buffer, uint8_t count );
void spi_send( uint8_t send_data );
void spi_bulk_exchange( uint8_t *send_buffer, uint8_t *receive_buffer, uint8_t count );
uint8_t spi_exchange( uint8_t send_data );

#endif /* __SPI_H__ */
```

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ STDIO_UART.C ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
//     https://www.gnu.org/savannah-checkouts/non-gnu/avr-libc/user-manual/group__avr__stdio.html

#ifndef F_CPU
```

```c
#define F_CPU 16000000UL
#endif

#include <stdio.h>
#include <avr/io.h>
#include "STDIO_UART.h"

#ifndef BAUD
#define BAUD 9600
#endif

#define MYUBRR (((F_CPU / (BAUD * 16UL))) - 1)

static FILE mystdout = FDEV_SETUP_STREAM(uart_putchar, NULL, _FDEV_SETUP_WRITE);
static FILE mystdin = FDEV_SETUP_STREAM(NULL, uart_getchar, _FDEV_SETUP_READ);

void uart_init(void)
{
    UBRR0H = MYUBRR >> 8;
    UBRR0L = MYUBRR;
    UCSR0B = (1<<RXEN0)|(1<<TXEN0);

    stdout = &mystdout;
    stdin  = &mystdin;
}

// Redirect stdout to UART
int uart_putchar(char c, FILE *stream) {
       if (c == '\n') {
              uart_putchar('\r', stream);
       }
       loop_until_bit_is_set(UCSR0A, UDRE0);
       UDR0 = c;
       return 0;
}

// Redirect stdin to UART
int uart_getchar(FILE *stream) {
       loop_until_bit_is_set(UCSR0A, RXC0);
       return UDR0;
}
```

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ STDIO_UART.C ~~~~~~

```c
//     https://www.gnu.org/savannah-checkouts/non-gnu/avr-libc/user-manual/group__avr__stdio.html

#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#include <stdio.h>
#include <avr/io.h>
#include "STDIO_UART.h"

#ifndef BAUD
#define BAUD 9600
#endif

#define MYUBRR (((F_CPU / (BAUD * 16UL))) - 1)

static FILE mystdout = FDEV_SETUP_STREAM(uart_putchar, NULL, _FDEV_SETUP_WRITE);
static FILE mystdin = FDEV_SETUP_STREAM(NULL, uart_getchar, _FDEV_SETUP_READ);

void uart_init(void)
{
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```c
    UBRR0H = MYUBRR >> 8;
    UBRR0L = MYUBRR;
    UCSR0B = (1<<RXEN0)|(1<<TXEN0);

    stdout = &mystdout;
    stdin  = &mystdin;
}

// Redirect stdout to UART
int uart_putchar(char c, FILE *stream) {
       if (c == '\n') {
              uart_putchar('\r', stream);
       }
       loop_until_bit_is_set(UCSR0A, UDRE0);
       UDR0 = c;
       return 0;
}

// Redirect stdin to UART
int uart_getchar(FILE *stream) {
       loop_until_bit_is_set(UCSR0A, RXC0);
       return UDR0;
}
```

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ STDIO_UART.H ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```c
//     https://www.gnu.org/savannah-checkouts/non-gnu/avr-libc/user-manual/group__avr__stdio.html

#ifndef STDIO_UART_H_
#define STDIO_UART_H_

void uart_init(void);
int uart_putchar(char c, FILE *stream);
int uart_getchar(FILE *stream);

#endif /* STDIO_UART_H_ */
```

~~~~~~

| THIS IS MY PARTNERS CODE |

~~~~~~

~~~~~~ MAIN.C ~~~~~~

```c
//     Set clock frequency
#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
unsigned int ADC_TEMP;
//     Set up UART for printf();
#ifndef BAUD
#define BAUD 9600
#endif
#include "STDIO_UART.h"

//     Include nRF24L01+ library
#include "nrf24l01.h"
#include "nrf24l01-mnemonics.h"
```

```c
#include "spi.h"
void print_config(void);
void ADC_INIT(void);
void READ_ADC(void);

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

int main(void){
       //     Set cliche message to send (message cannot exceed 32 characters)
       char tx_message[32];                      // Define string array
       char *tx_ptr = tx_message;
       strcpy(tx_message,"Hi :) !");      // Copy string into array

       //     Initialize UART
       uart_init();

       //     Initialize nRF24L01+ and print configuration info
    nrf24_init();
       print_config();

       ADC_INIT();
       //     Start listening to incoming messages
       printf("start listening\n");
       nrf24_start_listening();
       printf("Done listening\n");

       ADC_TEMP = 0;
    while (1){
                    //printf("outside if\n");
                    READ_ADC();
                    tx_ptr = tx_message;
                    //sprintf(tx_ptr, "%d", ADC_TEMP);
                    nrf24_send_message(tx_ptr);

                    delay_ms(100);
                    //continue;
                    //;
             if (message_received){
                    printf("inside if condition\n");
                    //     Message received, print it
                    message_received = false;
                    printf("Received message: %s\n",nrf24_read_message());
                    //     Send message as response
                    _delay_ms(500);
                    status = nrf24_send_message(tx_message);
                    if (status == true) printf("Message sent successfully\n");
             }
    }
}

//     Interrupt on IRQ pin
ISR(INT0_vect) {
       message_received = true;
}

void ADC_INIT(void){

       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)| //
        (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);

        // Timer/Counter1 Interrupt Mask Register
        TIMSK1 |= (1<<TOIE1); // enable overflow interrupt
        TCCR1B |= (1<<CS12)|(1<<CS10); // clock
        TCNT1 = 49911; //((16MHz/1024)*1)-1 = 15624

}
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/8 ; // Average

}

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%02X\n",data);
        nrf24_read(EN_AA,&data,1);
        printf("EN_AA           0x%02X\n",data);
        nrf24_read(EN_RXADDR,&data,1);
        printf("EN_RXADDR       0x%02X\n",data);
        nrf24_read(SETUP_RETR,&data,1);
        printf("SETUP_RETR      0x%02X\n",data);
        nrf24_read(RF_CH,&data,1);
        printf("RF_CH           0x%02X\n",data);
        nrf24_read(RF_SETUP,&data,1);
        printf("RF_SETUP        0x%02X\n",data);
        nrf24_read(STATUS,&data,1);
        printf("STATUS                  0x%02X\n",data);
        nrf24_read(FEATURE,&data,1);
        printf("FEATURE                 0x%02X\n",data);
        printf("-------------------------------------------\n\n");
}
```

~~~~~~

| THESE ARE THE LIBRARY FUNCTIONS |

~~~~~~ NRF24L01.C ~~~~~~

```
// MIT License
//
// Copyright (c) 2018 Helvijs Adams
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
```

```
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in all
// copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
// SOFTWARE.

// Set clock frequency
#ifndef F_CPU
#define F_CPU 16000000UL
#endif

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

// nRF24L01+ include files
#include "nrf24l01.h"
#include "nrf24l01-mnemonics.h"
#include "spi.h"
 0x73, 0x73, 0x73, 0x73,
// Settings
uint8_t rx_address[5] = { 0x42, 0x42, 0x42, 0x42, 0x42 };     // Read pipe address
uint8_t tx_address[5] = { 0x73, 0x73, 0x73, 0x73, 0x73 };     // Write pipe address
#define READ_PIPE    0                                        // Number of read pipe
//
// -AUTO_ACK can be disabled when running on 2MBPS @ <= 32 byte messages.
// -250KBPS and 1MBPS with AUTO_ACK disabled lost many packets
// if the packet size was bigger than 4 bytes.
// -If AUTO_ACK is enabled, tx_address = rx_address.
//
#define AUTO_ACK     false         // Auto acknowledgment
#define DATARATE     RF_DR_2MBPS   // 250kbps, 1mbps, 2mbps
#define POWER        POWER_MAX     // Set power (MAX 0dBm..HIGH -6dBm..LOW -12dBm.. MIN -18dBm)
#define CHANNEL      0x74          // 2.4GHz-2.5GHz channel selection (0x01 - 0x7C)
#define DYN_PAYLOAD        true    // Dynamic payload enabled
#define CONTINUOUS         false   // Continuous carrier transmit mode (not tested)
//
// ISR(INT0_vect) is triggered depending on config (only one can be true)
//
#define RX_INTERRUPT true   // Interrupt when message is received (RX)
#define TX_INTERRUPT false  // Interrupt when message is sent (TX)
#define RT_INTERRUPT false  // Interrupt when maximum re-transmits are reached (MAX_RT)
//
// -PIN map.
// -If CE or CSN is changed to different PIN e.g. PC0
// then change DDRB -> DDRC, PORTB -> PORTC and so on
//
// CE
#define CE_DDR             DDRB
#define CE_PORT            PORTB
```

```
#define CE_PIN              DDB1                                                    // CE
connected to PB1
// CSN
#define CSN_DDR             DDRB
#define CSN_PORT     PORTB
#define CSN_PIN             DDB2                                                    // CSN
connected to PB2
// IRQ
#define IRQ_DDR             DDRD
#define IRQ_PORT     PORTD
#define IRQ_PIN             DDD2                                                    // IRQ
connected to PD2
// MOSI
#define MOSI_DDR     DDRB
#define MOSI_PORT    PORTB
#define MOSI_PIN     DDB3
// MISO
#define MISO_DDR     DDRB
#define MISO_PORT    PORTB
#define MISO_PIN     DDB4
// SCK
#define SCK_DDR             DDRB
#define SCK_PORT     PORTB
#define SCK_PIN             DDB5

// PIN toggling
#define setbit(port, bit) (port) |= (1 << (bit))
#define clearbit(port, bit) (port) &= ~(1 << (bit))
#define ce_low clearbit(CE_PORT,CE_PIN)
#define ce_high setbit(CE_PORT,CE_PIN)
#define csn_low clearbit(CSN_PORT,CSN_PIN)
#define csn_high setbit(CSN_PORT,CSN_PIN)

// Used to store SPI commands
uint8_t data;

uint8_t nrf24_send_spi(uint8_t register_address, void *data, unsigned int bytes)
{
       uint8_t status;
       csn_low;
       status = spi_exchange(register_address);
       for (unsigned int i = 0; i < bytes; i++)
       ((uint8_t*)data)[i] = spi_exchange(((uint8_t*)data)[i]);
       csn_high;
       return status;
}

uint8_t nrf24_write(uint8_t register_address, uint8_t *data, unsigned int bytes)
{
       return nrf24_send_spi(W_REGISTER | register_address, data, bytes);
}

uint8_t nrf24_read(uint8_t register_address, uint8_t *data, unsigned int bytes)
{
       return nrf24_send_spi(R_REGISTER | register_address, data, bytes);
}

void nrf24_init(void)
{
       // Interrupt on falling edge of INT0 (PD2) from IRQ pin
       cli();                              // Disable interrupts
       EICRA |= (1 << ISC01);
       EIMSK |= (1 << INT0);
       sei();                              // Enable interrupts
```

```
	// CSN and CE as outputs and initial states
	setbit(CE_DDR,CE_PIN);
	setbit(CSN_DDR,CSN_PIN);
	csn_high;
	ce_low;

	// Initialize SPI
	spi_master_init();
	_delay_ms(100);                          // Power on reset 100ms

	// Start nRF24L01+ config
	data =
	(!(RX_INTERRUPT) << MASK_RX_DR) |  // IRQ interrupt on RX (0 = enabled)
	(!(TX_INTERRUPT) << MASK_TX_DS) |  // IRQ interrupt on TX (0 = enabled)
	(!(RT_INTERRUPT) << MASK_MAX_RT) |// IRQ interrupt on auto retransmit counter overflow (0 =
enabled)
	(1 << EN_CRC) |                                  // CRC enable
	(1 << CRC0) |                                    // CRC scheme
	(1 << PWR_UP) |                                  // Power up
	(1 << PRIM_RX);                                  // TX/RX select
	nrf24_write(CONFIG,&data,1);

	// Auto-acknowledge on all pipes
	data =
	(AUTO_ACK << ENAA_P5) |
	(AUTO_ACK << ENAA_P4) |
	(AUTO_ACK << ENAA_P3) |
	(AUTO_ACK << ENAA_P2) |
	(AUTO_ACK << ENAA_P1) |
	(AUTO_ACK << ENAA_P0);
	nrf24_write(EN_AA,&data,1);

	// Set retries
	data = 0xF0;         // Delay 4000us with 1 re-try (will be added in settings)
	nrf24_write(SETUP_RETR,&data,1);

	// Disable RX addresses
	data = 0;
	nrf24_write(EN_RXADDR, &data, 1);

	// Set channel
	data = CHANNEL;
	nrf24_write(RF_CH,&data,1);

	// Setup
	data =
	(CONTINUOUS << CONT_WAVE) |                      // Continuous carrier transmit
	((DATARATE >> RF_DR_HIGH) << RF_DR_HIGH) |       // Data rate
	((POWER >> RF_PWR) << RF_PWR);                   // PA level
	nrf24_write(RF_SETUP,&data,1);

	// Status - clear TX/RX FIFO's and MAX_RT by writing 1 into them
	data =
	(1 << RX_DR) |              // RX FIFO
	(1 << TX_DS) |              // TX FIFO
	(1 << MAX_RT);              // MAX RT
	nrf24_write(STATUS,&data,1);

	// Dynamic payload on all pipes
	data =
	(DYN_PAYLOAD << DPL_P0) |
	(DYN_PAYLOAD << DPL_P1) |
	(DYN_PAYLOAD << DPL_P2) |
```

```
	(DYN_PAYLOAD << DPL_P3) |
	(DYN_PAYLOAD << DPL_P4) |
	(DYN_PAYLOAD << DPL_P5);
	nrf24_write(DYNPD, &data,1);

	// Enable dynamic payload
	data =
	(DYN_PAYLOAD << EN_DPL) |
	(AUTO_ACK << EN_ACK_PAY) |
	(AUTO_ACK << EN_DYN_ACK);
	nrf24_write(FEATURE,&data,1);

	// Flush TX/RX
	// Clear RX FIFO which will reset interrupt
	uint8_t data = (1 << RX_DR) | (1 << TX_DS) | (1 << MAX_RT);
	nrf24_write(FLUSH_RX,0,0);
	nrf24_write(FLUSH_TX,0,0);

	// Open pipes
	nrf24_write(RX_ADDR_P0 + READ_PIPE,rx_address,5);
	nrf24_write(TX_ADDR,tx_address,5);
	nrf24_write(EN_RXADDR,&data,1);
	data |= (1 << READ_PIPE);
	nrf24_write(EN_RXADDR,&data,1);
}

void nrf24_write_ack(void)
{
	const void *ack = "A";
	unsigned int length = 1;
	csn_low;
	spi_send(W_ACK_PAYLOAD);
	while (length--) spi_send(*(uint8_t *)ack++);
	csn_high;
}

void nrf24_state(uint8_t state)
{
	uint8_t config_register;
	nrf24_read(CONFIG,&config_register,1);

	switch (state)
	{
		case POWERUP:
		// Check if already powered up
		if (!(config_register & (1 << PWR_UP)))
		{
			data = config_register | (1 << PWR_UP);
			nrf24_write(CONFIG,&data,1);
			// 1.5ms from POWERDOWN to start up
			_delay_ms(2);
		}
		break;
		case POWERDOWN:
		data = config_register & ~(1 << PWR_UP);
		nrf24_write(CONFIG,&data,1);
		break;
		case RECEIVE:
		data = config_register | (1 << PRIM_RX);
		nrf24_write(CONFIG,&data,1);
		// Clear STATUS register
		data = (1 << RX_DR) | (1 << TX_DS) | (1 << MAX_RT);
		nrf24_write(STATUS,&data,1);
		break;
```

```
                case TRANSMIT:
                data = config_register & ~(1 << PRIM_RX);
                nrf24_write(CONFIG,&data,1);
                break;
                case STANDBY1:
                ce_low;
                break;
                case STANDBY2:
                data = config_register & ~(1 << PRIM_RX);
                nrf24_write(CONFIG,&data,1);
                ce_high;
                _delay_us(150);
                break;
        }
}

void nrf24_start_listening(void)
{
        nrf24_state(RECEIVE);                       // Receive mode
        //if (AUTO_ACK) nrf24_write_ack(); // Write acknowledgment
        ce_high;
        _delay_us(150);                             // Settling time
}

uint8_t nrf24_send_message(const void *tx_message)
{
        // For printf();
        char temp[32];
        memset(temp,0,32);
        strcpy(temp,tx_message);

        // Message length
        uint8_t length = strlen(tx_message);

        // Transmit mode
        nrf24_state(TRANSMIT);

        // Flush TX/RX and clear TX interrupt
        nrf24_write(FLUSH_RX,0,0);
        nrf24_write(FLUSH_TX,0,0);
        data = (1 << TX_DS);
        nrf24_write(STATUS,&data,1);

        // Disable interrupt on RX
        nrf24_read(CONFIG,&data,1);
        data |= (1 << MASK_RX_DR);
        nrf24_write(CONFIG,&data,1);

        // Start SPI, load message into TX_PAYLOAD
        csn_low;
        if (AUTO_ACK) spi_send(W_TX_PAYLOAD);
        else spi_send(W_TX_PAYLOAD_NOACK);
        while (length--) spi_send(*(uint8_t *)tx_message++);
        spi_send(0);
        csn_high;

        // Send message by pulling CE high for more than 10us
        ce_high;
        _delay_us(15);
        ce_low;

        // Wait for message to be sent (TX_DS flag raised)
        nrf24_read(STATUS,&data,1);
        while(!(data & (1 << TX_DS))) nrf24_read(STATUS,&data,1);
```

```
        printf("Message sent: %s\n",temp);

        // Enable interrupt on RX
        nrf24_read(CONFIG,&data,1);
        data &= ~(1 << MASK_RX_DR);
        nrf24_write(CONFIG,&data,1);

        // Continue listening
        nrf24_start_listening();

        return 1;
}

unsigned int nrf24_available(void)
{
        uint8_t config_register;
        nrf24_read(FIFO_STATUS,&config_register,1);
        if (!(config_register & (1 << RX_EMPTY))) return 1;
        return 0;
}

const char * nrf24_read_message(void)
{
        // Message placeholder
        static char rx_message[32];
        memset(rx_message,0,32);

        // Write ACK message
        if (AUTO_ACK) nrf24_write_ack();

        // Get length of incoming message
        nrf24_read(R_RX_PL_WID,&data,1);

        // Read message
        if (data > 0) nrf24_send_spi(R_RX_PAYLOAD,&rx_message,data+1);

        // Check if there is message in array
        if (strlen(rx_message) > 0)
        {
                // Clear RX interrupt
                data = (1 << RX_DR);
                nrf24_write(STATUS,&data,1);

                return rx_message;
        }

        // Clear RX interrupt
        data = (1 << RX_DR);
        nrf24_write(STATUS,&data,1);

        return "failed";
}
```

~~~~~~ NRF24L01.H ~~~~~~

```
// MIT License
//
// Copyright (c) 2018 Helvijs Adams
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
```
~~~~~~

```
//
// The above copyright notice and this permission notice shall be included in all
// copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
// SOFTWARE.

#ifndef _NRF24L01_H
#define _NRF24L01_H

//     States
#define POWERUP      1
#define POWERDOWN    2
#define RECEIVE      3
#define TRANSMIT     4
#define STANDBY1     5
#define STANDBY2     6

//     Forward declarations
uint8_t nrf24_send_spi(uint8_t register_address, void *data, unsigned int bytes);
uint8_t nrf24_write(uint8_t register_address, uint8_t *data, unsigned int bytes);
uint8_t nrf24_read(uint8_t register_address, uint8_t *data, unsigned int bytes);
void nrf24_init(void);
void nrf24_state(uint8_t state);
void nrf24_start_listening(void);
unsigned int nrf24_available(void);
const char * nrf24_read_message(void);
uint8_t nrf24_send_message(const void *tx_message);

#endif /*_NRF24L01_H*/

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ NRF24101-MNEMONICS.H ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*  Copyright (c) 2015 B. Sidhipong <bsidhipong@gmail.com>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>. */

#ifndef __NORDIC_NRF24L01_RADIO_H__
#define __NORDIC_NRF24L01_RADIO_H__
#include <avr/common.h>

/* SPI commands */
#define REGISTER_MASK       0b00011111
#define R_REGISTER          0b00000000 /* 000A AAAA | AAAAA = 5-bit register map address */
#define W_REGISTER          0b00100000 /* 001A AAAA */
#define R_RX_PAYLOAD        0b01100001
#define W_TX_PAYLOAD        0b10100000
#define FLUSH_TX            0b11100001
#define FLUSH_RX            0b11100010
```

```
#define REUSE_TX_PL         0b11100011
#define ACTIVATE            0b01010000
#define R_RX_PL_WID         0b01100000
#define ACK_PAYLOAD_MASK    0b00000111
#define W_ACK_PAYLOAD       0b10101000 /* 1010 1PPP | PPP = pipe number */
#define W_TX_PAYLOAD_NOACK  0b10110000
#define NOP                 0b11111111

/* CONFIG: configuration register */
#define CONFIG              0x00
#define MASK_RX_DR          6
#define MASK_TX_DS          5
#define MASK_MAX_RT         4
#define EN_CRC              3
#define CRC0                2
#define PWR_UP              1
#define PRIM_RX             0

/* EN_AA: Enhanced ShockBurst™
 * Enable 'Auto Acknowledgment' function.  Disable this functionality
 * to be compatible with nRF2401. */
#define EN_AA               0x01
#define ENAA_P5             5
#define ENAA_P4             4
#define ENAA_P3             3
#define ENAA_P2             2
#define ENAA_P1             1
#define ENAA_P0             0
/* EN_RXADDR: enable RX addresses */
#define EN_RXADDR           0x02
#define ERX_P5              5
#define ERX_P4              4
#define ERX_P3              3
#define ERX_P2              2
#define ERX_P1              1
#define ERX_P0              0

/* SETUP_AW: seup of address widths */
#define SETUP_AW            0x03
#define AW                  0 /* 1:0 */

/* SETUP_RETR: setup of automatic retransmission */
#define SETUP_RETR          0x04
#define ARD                 4 /* 7:4 */
#define ARC                 0 /* 3:0 */

#define RF_CH               0x05

/* RF_SETUP: RF setup register */
#define RF_SETUP            0x06
#define CONT_WAVE           7
#define RF_DR_LOW           5
#define PLL_LOCK            4
#define RF_DR_HIGH          3      /* RF_DR:RF_DR_LOW = RF data rate */
#define RF_DR_250KBPS       0b00100000    /* available on nRF24L01+ */
#define RF_DR_1MBPS         0b00000000
#define RF_DR_2MBPS         0b00001000
#define RF_PWR              1             /* 2:1 */
#define POWER_MIN           0b00000000
#define POWER_LOW           0b00000010
#define POWER_HIGH          0b00000100
#define POWER_MAX           0b00000110

/* STATUS: status register */
```

```
#define STATUS              0x07
#define RX_DR               6
#define TX_DS               5
#define MAX_RT              4
#define RX_P_NO             1       /* 3:1 */
#define TX_FULL             0

/* OBSERVE_TX: transmit observe register */
#define OBSERVE_TX          0x08
#define PLOS_CNT            4       /* 7:4 */
#define ARC_CNT             0       /* 3:0 */

#define RPD                 0x09
#define RX_ADDR_P0          0x0A
#define RX_ADDR_P1          0x0B
#define RX_ADDR_P2          0x0C
#define RX_ADDR_P3          0x0D
#define RX_ADDR_P4          0x0E
#define RX_ADDR_P5          0x0F
#define TX_ADDR             0x10
#define RX_PW_P0            0x11
#define RX_PW_P1            0x12
#define RX_PW_P2            0x13
#define RX_PW_P3            0x14
#define RX_PW_P4            0x15
#define RX_PW_P5            0x16

/* FIFO_STATUS: FIFO status register */
#define FIFO_STATUS         0x17
#define TX_REUSE            6
#define FIFO_FULL           5
#define TX_EMPTY            4
#define RX_FULL             1
#define RX_EMPTY            0

/* DYNPD: enable dynamic payload length */
#define DYNPD               0x1C
#define DPL_P5              5
#define DPL_P4              4
#define DPL_P3              3
#define DPL_P2              2
#define DPL_P1              1
#define DPL_P0              0

/* FEATURE: */
#define FEATURE             0x1D
#define EN_DPL              2
#define EN_ACK_PAY          1
#define EN_DYN_ACK          0

#endif /* __NORDIC_NRF24L01_RADIO_H__ */
```

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ SPI.C ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
/*  Copyright (c) 2015 B. Sidhipong <bsidhipong@gmail.com>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>. */

#include <avr/common.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include "spi.h"

#define DDR_SPI             DDRB
#define DD_MOSI             DDB3
#define DD_MISO             DDB4
#define DD_SCK              DDB5

void spi_master_init( void )
{
       DDR_SPI &= ~_BV(DD_MISO);
       DDR_SPI |= (_BV(DD_MOSI) | _BV(DD_SCK));
       /* 19.5.1 SPCR – SPI Control Register
        *
        * • Bit 7 – SPIE: SPI Interrupt Enable
        * This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and
the if the Global
        * Interrupt Enable bit in SREG is set.
        *
        * • Bit 6 – SPE: SPI Enable
        * When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any
SPI operations.
        *
        * • Bit 5 – DORD: Data Order
        * When the DORD bit is written to one, the LSB of the data word is transmitted first.
        * When the DORD bit is written to zero, the MSB of the data word is transmitted first.
        *
        * • Bit 4 – MSTR: Master/Slave Select
        * This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic
zero. If SS is
        * configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in
SPSR will become
        * set. The user will then have to set MSTR to re-enable SPI Master mode.
        *
        * • Bit 3 – CPOL: Clock Polarity
        * When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is
low when idle.
        *
        * • Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0
        * These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have
no effect on
        * the Slave. The relationship between SCK and the Oscillator Clock frequency fosc is shown in
the following table:
        *
        * Table 19-5 Relationship Between SCK and Oscillator Frequency
        * SPI2X | SPR1 | SPR0 | SCK Frequency
        * ===================================
        *   0   |   0  |   0  | fosc/4
        *   0   |   0  |   1  | fosc/16
        *   0   |   1  |   0  | fosc/64
        *   0   |   1  |   1  | fosc/128
        *   1   |   0  |   0  | fosc/2
        *   1   |   0  |   1  | fosc/8
        *   1   |   1  |   0  | fosc/32
        *   1   |   1  |   1  | fosc/64
        */
       SPCR = (0 << SPIE) |
              (1 << SPE)  |
```

```
		(0 << DORD) |
		(1 << MSTR) |
		(0 << CPOL) |
		(0 << SPR1) | (0 << SPR0);
	/* 19.5.2   SPSR – SPI Status Register
	 *
	 * • Bit 7 – SPIF: SPI Interrupt Flag
	 * When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE
in SPCR is set and
	 * global interrupts are enabled. If SS is an input and is driven low when the SPI is in Master
mode, this will also
	 * set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt
handling vector.
	 * Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF
set, then accessing the
	 * SPI Data Register (SPDR).
	 *
	 * • Bit 0 – SPI2X: Double SPI Speed Bit
	 * When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI
is in Master
	 * mode (see Table 19-5). This means that the minimum SCK period will be two CPU clock periods.
When the SPI
	 * is configured as Slave, the SPI is only guaranteed to work at fosc/4 or lower. */
	SPSR |= _BV(SPI2X);
}

void spi_bulk_send( uint8_t *send_buffer, uint8_t count )
{
	while ( count-- ) {
		SPDR = *send_buffer++;
		loop_until_bit_is_set(SPSR, SPIF);
	}
}

void spi_send( uint8_t send_data )
{
	SPDR = send_data;
	loop_until_bit_is_set(SPSR, SPIF);
}

void spi_bulk_exchange( uint8_t *send_buffer, uint8_t *receive_buffer, uint8_t count )
{
	while ( count-- ) {
		SPDR = *send_buffer++;
		loop_until_bit_is_set(SPSR, SPIF);
		*receive_buffer++ = SPDR;
	}
}

uint8_t spi_exchange( uint8_t send_data )
{
	SPDR = send_data;
	loop_until_bit_is_set(SPSR, SPIF);
	return SPDR;
}
```

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ SPI.H ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
/*  Copyright (c) 2015 B. Sidhipong <bsidhipong@gmail.com>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>. */

#ifndef __SMALL_SPI_H__
#define __SMALL_SPI_H__
#include <avr/common.h>

void spi_master_init( void );
void spi_bulk_send( uint8_t *send_buffer, uint8_t count );
void spi_send( uint8_t send_data );
void spi_bulk_exchange( uint8_t *send_buffer, uint8_t *receive_buffer, uint8_t count );
uint8_t spi_exchange( uint8_t send_data );

#endif /* __SPI_H__ */

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ STDIO_UART.C ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//     https://www.gnu.org/savannah-checkouts/non-gnu/avr-libc/user-manual/group__avr__stdio.html

#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#include <stdio.h>
#include <avr/io.h>
#include "STDIO_UART.h"

#ifndef BAUD
#define BAUD 9600
#endif

#define MYUBRR (((F_CPU / (BAUD * 16UL))) - 1)

static FILE mystdout = FDEV_SETUP_STREAM(uart_putchar, NULL, _FDEV_SETUP_WRITE);
static FILE mystdin = FDEV_SETUP_STREAM(NULL, uart_getchar, _FDEV_SETUP_READ);

void uart_init(void)
{
    UBRR0H = MYUBRR >> 8;
    UBRR0L = MYUBRR;
    UCSR0B = (1<<RXEN0)|(1<<TXEN0);

    stdout = &mystdout;
    stdin  = &mystdin;
}

// Redirect stdout to UART
int uart_putchar(char c, FILE *stream) {
       if (c == '\n') {
              uart_putchar('\r', stream);
       }
       loop_until_bit_is_set(UCSR0A, UDRE0);
       UDR0 = c;
       return 0;
}

// Redirect stdin to UART
int uart_getchar(FILE *stream) {
```

```
	loop_until_bit_is_set(UCSR0A, RXC0);
	return UDR0;
}
```

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ STDIO_UART.C ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
//	https://www.gnu.org/savannah-checkouts/non-gnu/avr-libc/user-manual/group__avr__stdio.html

#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#include <stdio.h>
#include <avr/io.h>
#include "STDIO_UART.h"

#ifndef BAUD
#define BAUD 9600
#endif

#define MYUBRR (((F_CPU / (BAUD * 16UL))) - 1)

static FILE mystdout = FDEV_SETUP_STREAM(uart_putchar, NULL, _FDEV_SETUP_WRITE);
static FILE mystdin = FDEV_SETUP_STREAM(NULL, uart_getchar, _FDEV_SETUP_READ);

void uart_init(void)
{
    UBRR0H = MYUBRR >> 8;
    UBRR0L = MYUBRR;
    UCSR0B = (1<<RXEN0)|(1<<TXEN0);

    stdout = &mystdout;
    stdin  = &mystdin;
}

// Redirect stdout to UART
int uart_putchar(char c, FILE *stream) {
	if (c == '\n') {
		uart_putchar('\r', stream);
	}
	loop_until_bit_is_set(UCSR0A, UDRE0);
	UDR0 = c;
	return 0;
}

// Redirect stdin to UART
int uart_getchar(FILE *stream) {
	loop_until_bit_is_set(UCSR0A, RXC0);
	return UDR0;
}
```

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ STDIO_UART.H ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

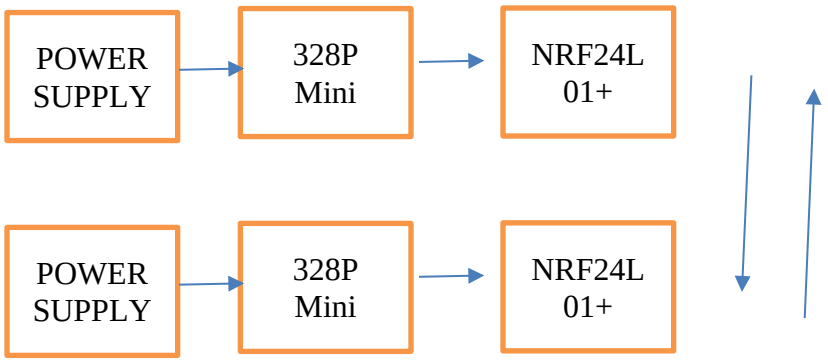
```
//	https://www.gnu.org/savannah-checkouts/non-gnu/avr-libc/user-manual/group__avr__stdio.html

#ifndef STDIO_UART_H_
#define STDIO_UART_H_

void uart_init(void);
int uart_putchar(char c, FILE *stream);
int uart_getchar(FILE *stream);

#endif /* STDIO_UART_H_ */
```

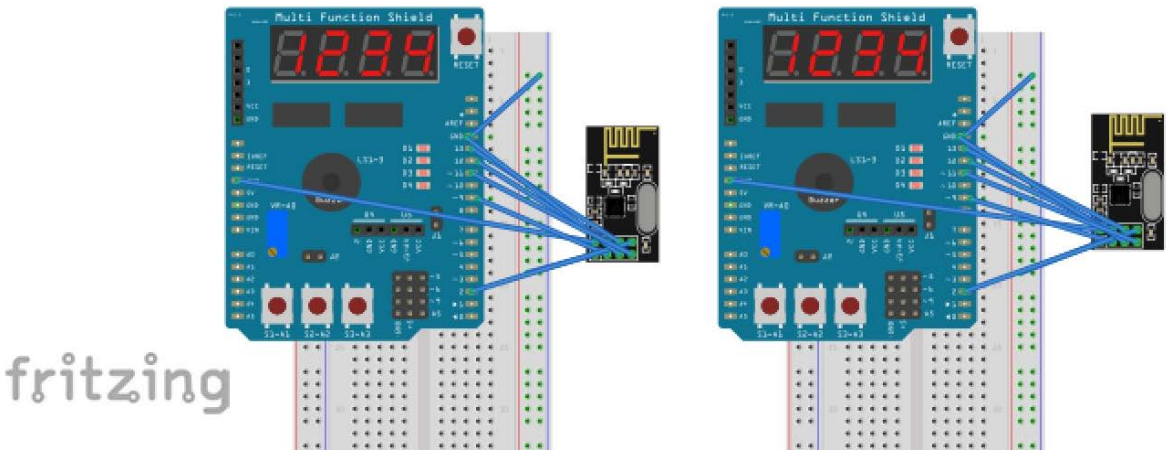
3. SCHEMATICS



4. PARTNERS NAME

My partner for this assignment was Ricky Perez.

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)



6. VIDEO LINKS OF EACH DEMO

https://www.youtube.com/watch?v=BPUSEU625Hc

7. GITHUB LINK OF THIS DA

https://github.com/mendos1/subnission_da

Student Academic Misconduct Policy

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

"This assignment submission is my own, original work".

NAME OF THE STUDENT