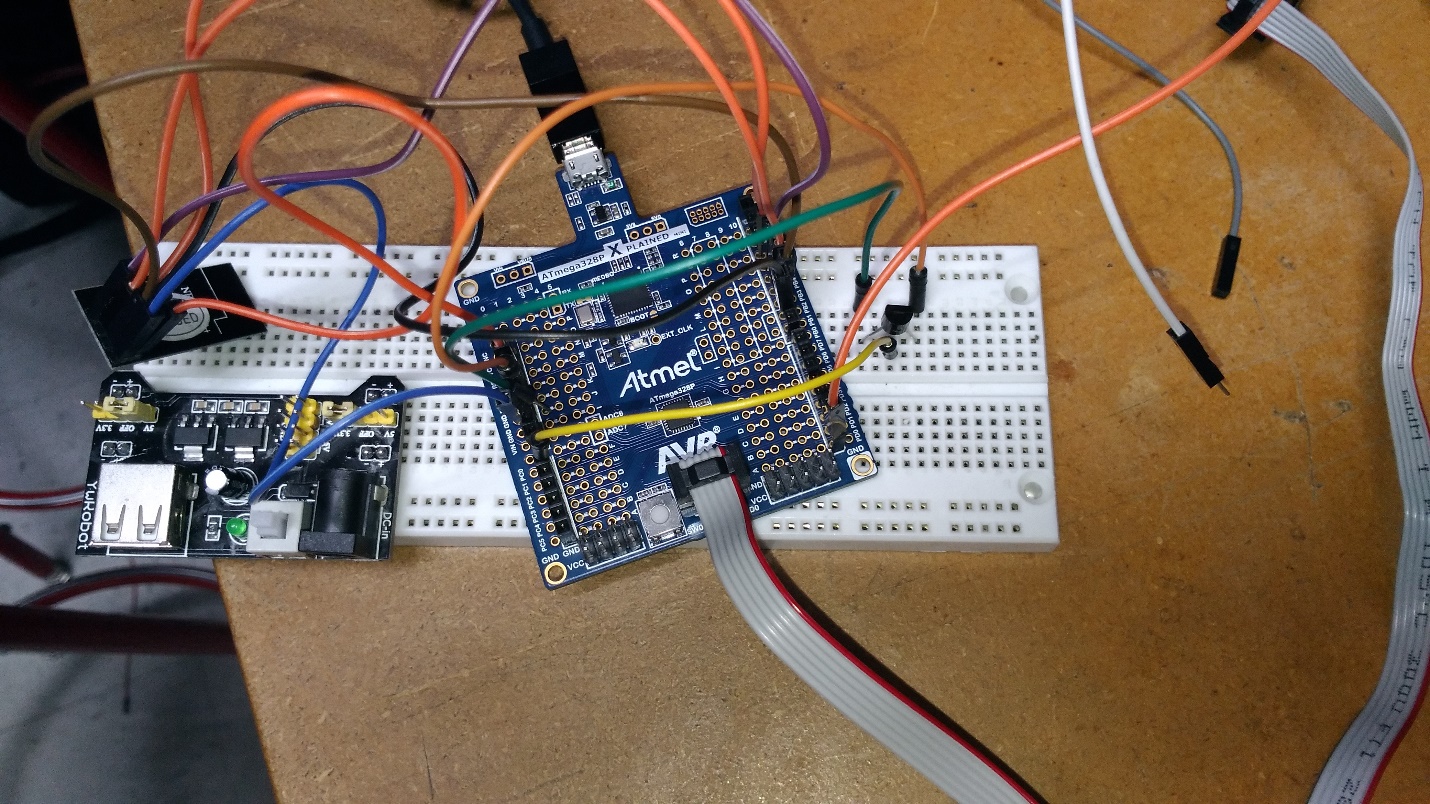
Adrian Ruiz

CPE301

Midterm 2

Partner: Elizabeth Heider

Transmitting Control



#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#include <string.h>

#include "nrf24l01.h"

#include "ioe.h"

void setup\_timer(void);

nRF24L01 \*setup\_rf(void);

void adc\_init(void);

volatile bool rf\_interrupt = false;

volatile bool send\_message = false;

volatile int temp;

char outs[3];

int main(void) {

usart0\_init\_();//init the uart

uint8\_t to\_address[5] = { 0x11, 0x11, 0x11, 0x11, 0x11 };

DDRC = 0x00; //set Port C read the temp

sei();

nRF24L01 \*rf = setup\_rf();

setup\_timer();

while (true) {

while(ADCSRA & (1<<ADSC)) {}

temp = ADC;

adc\_temp = (5\*adc\_temp\*100/1024); //calculate temperature from voltage in celuis

adc\_temp = adc\_temp \* (2);

adc\_temp = adc\_temp +32; //temp in far

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;

nRF24L01Message msg;

sprintf(outs, "%d" , temp);

memcpy(msg.data, outs, 3);

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);

}

// each one second interrupt

ISR(TIMER1\_COMPA\_vect) {

send\_message = true;

}

// nRF24L01 interrupt

ISR(INT0\_vect) {

rf\_interrupt = true;

}

//init the adc

void adc\_init()

{

ADMUX =0;

ADCSRA= (1<<ADPS0) | (1<<ADPS1) | (1<<ADPS2); //sample rate of 125kHz at 16MHz

ADMUX |= (1<<REFS0); //Set ADC reference to AVCC

ADMUX |= (1<<ADLAR); //Left adjust ADC Result to allow 8 bit reading

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

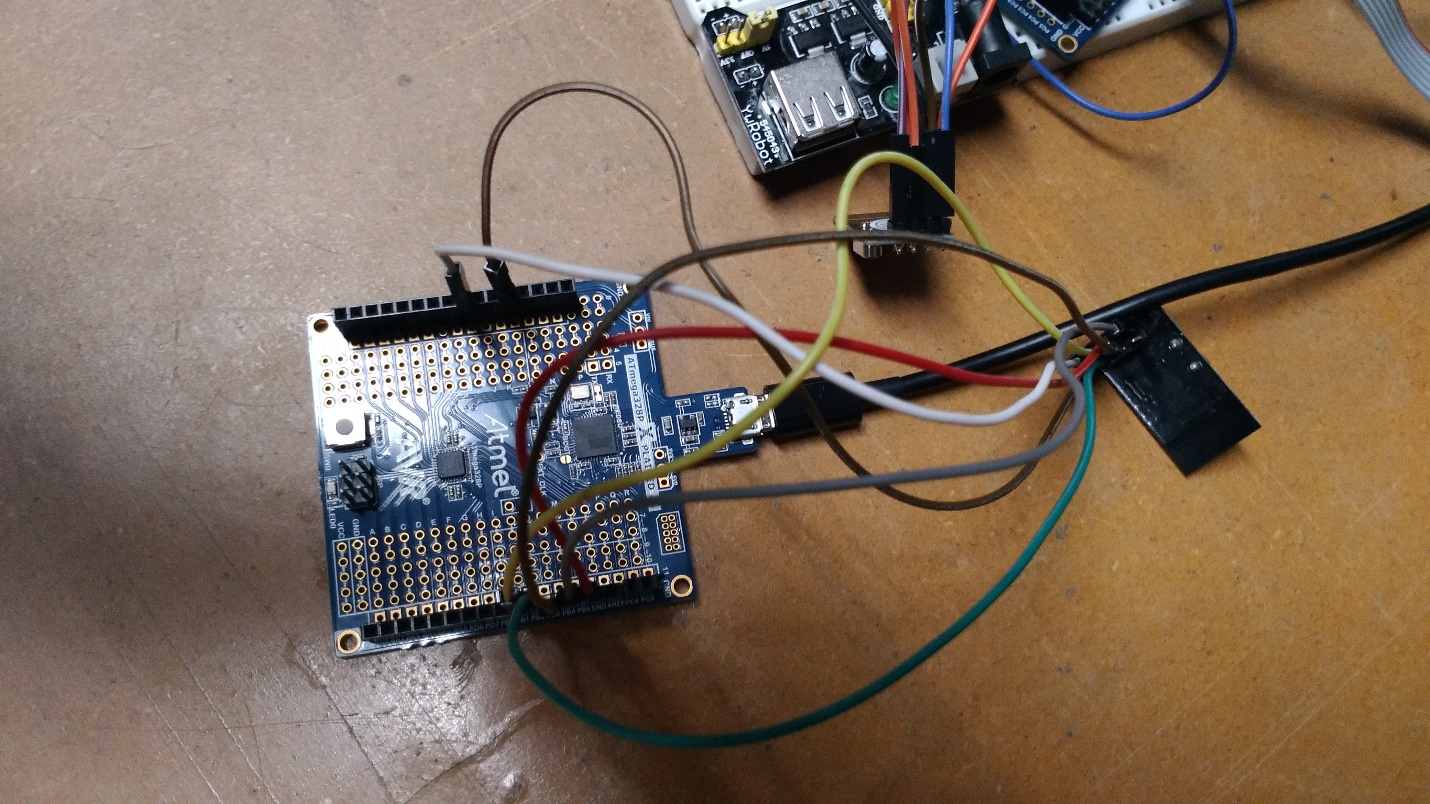
ADCSRA |= (1<<ADATE); //set ADC auto trigger enable

ADCSRB = 0; //0 for free running mode

ADCSRA |= (1<<ADSC) ; //start A2D Conversion

}

Receiving Control and Code



#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#include <string.h>

#ifndef F\_CPU

#define F\_CPU 16000000UL

#endif

#include <util/delay.h>

#include "nrf24l01.h"

#include "ioe.h"

#include "nrf24l01-mnemonics.h"

#define UBRR\_2400 207

//function protypes

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);

void USART\_INIT(unsigned int ubrr);

void USART\_tx\_string(char\* data);

volatile bool rf\_interrupt = false;

char\* outs[20]; //string buffer

int main(void) {

usart0\_init\_(); //init the uart

\_delay\_ms(2000);

uint8\_t address[5] = { 0x11, 0x11, 0x11, 0x11, 0x11 }; //the address that the modules try to match

printm("Address INIT\r\n");

prepare\_led\_pin(); //not used

printm("LED PREP\r\n");

sei();

nRF24L01 \*rf = setup\_rf();

nRF24L01\_listen(rf, 0, address); //listen for the address

uint8\_t addr[5];

nRF24L01\_read\_register(rf, CONFIG, addr, 1);

while (true) {

if (rf\_interrupt) { //when an intrupt happens

rf\_interrupt = false; //print the temp

while (nRF24L01\_data\_received(rf)) {

nRF24L01Message msg;

nRF24L01\_read\_received\_data(rf, &msg);

process\_message((char \*)msg.data);

}

nRF24L01\_listen(rf, 0, address);

}

}

return 0;

}

//setup the nrf module

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;

}

//function that prints the temp

void process\_message(char \*message) {

printm(message);

\_delay\_ms(1000);

}

inline void prepare\_led\_pin(void) {

DDRB |= \_BV(PB0);

PORTB &= ~\_BV(PB0);

}

//not used

inline void set\_led\_high(void) {

PORTB |= \_BV(PB0);

}

//not used

inline void set\_led\_low(void) {

PORTB &= ~\_BV(PB0);

}

// nRF24L01 interrupt

ISR(INT0\_vect) {

rf\_interrupt = true;

}

//not used

void USART\_INIT(unsigned int ubrr){

UBRR0H = (unsigned char)(ubrr>>8);

UBRR0L = (unsigned char)ubrr;

UCSR0B = (1<<TXEN0);

UCSR0C = (1<<UCSZ00);

}

//not used

void USART\_tx\_string(char\* data){

while((\*data != '\0')){

while(!(UCSR0A & (1<<UDRE0)));

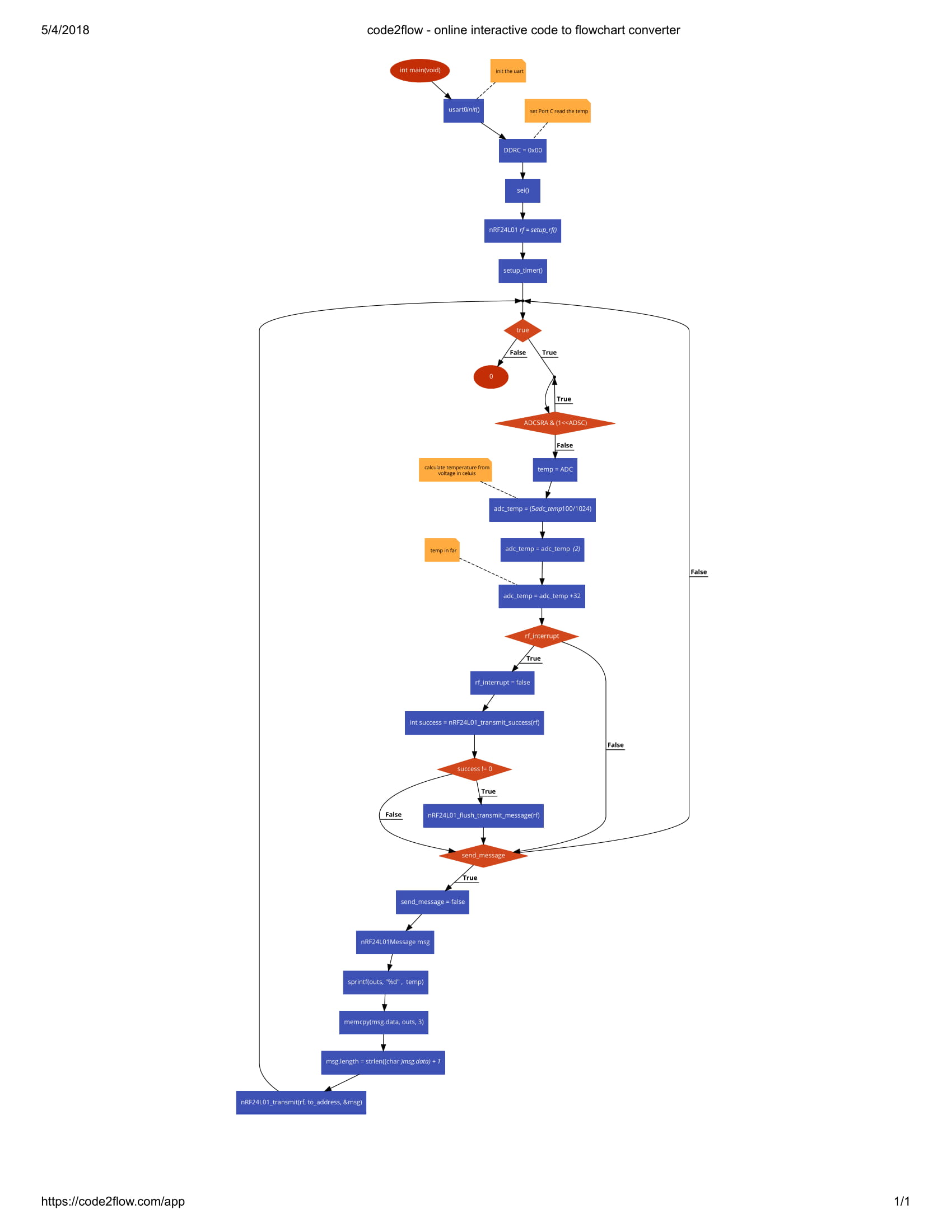
UDR0 = \*data;

data++;

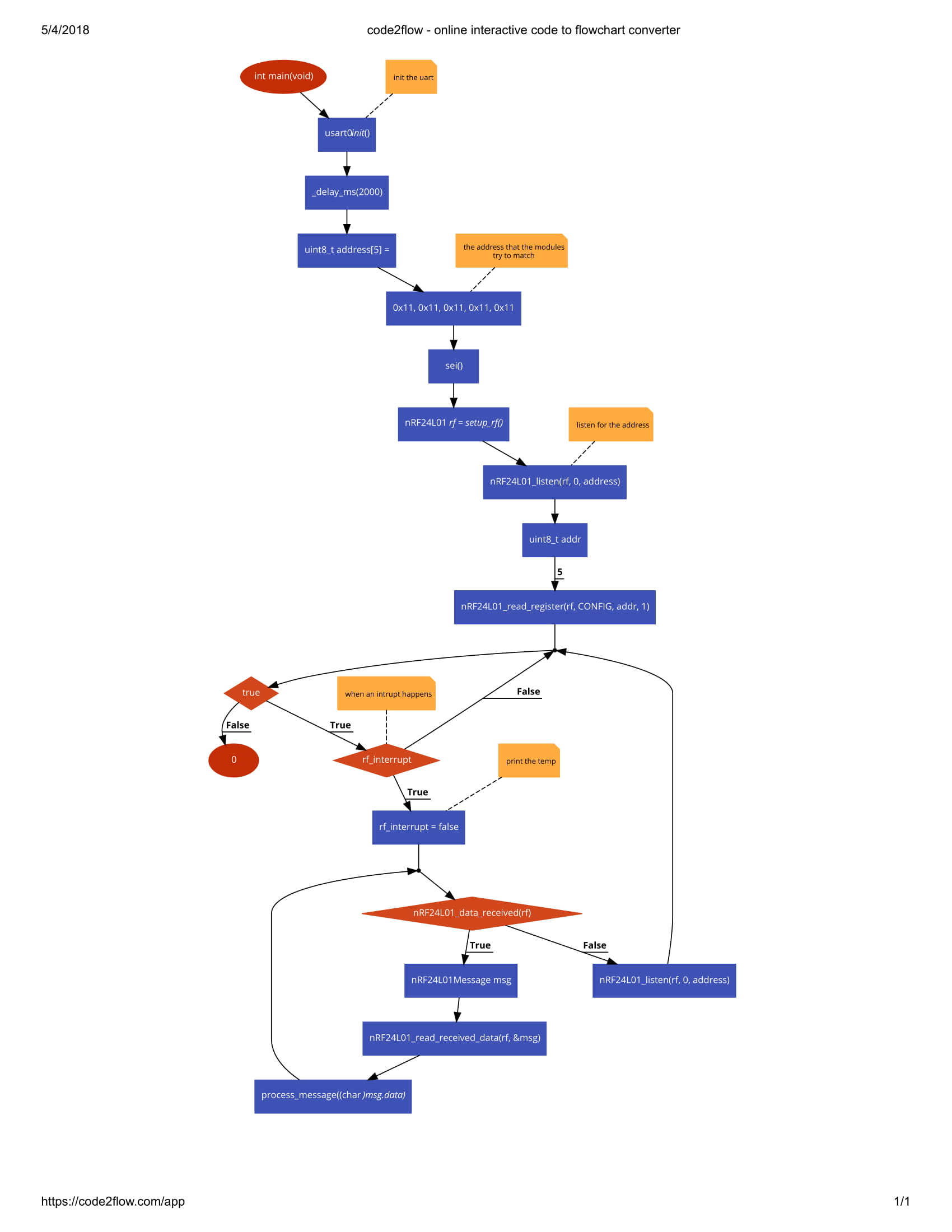
}

}

TX.C Flowchart



RX.c Flowchart



Video Link: