CPE301 – SPRING 2019

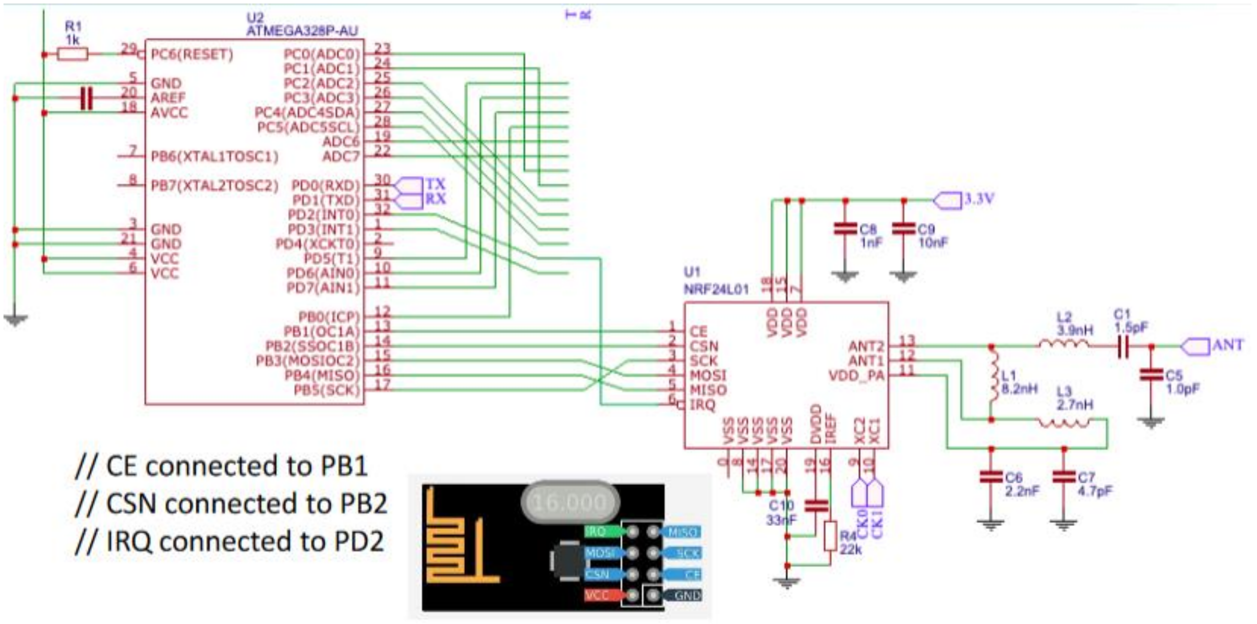
Design Assignment 5

Student Name: John Duriman
Student #: 5002373995
Student Email: duriman@unlv.nevada.edu
Primary Github address: https://github.com/johnduriman/pirahnaplant.git
Directory:

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

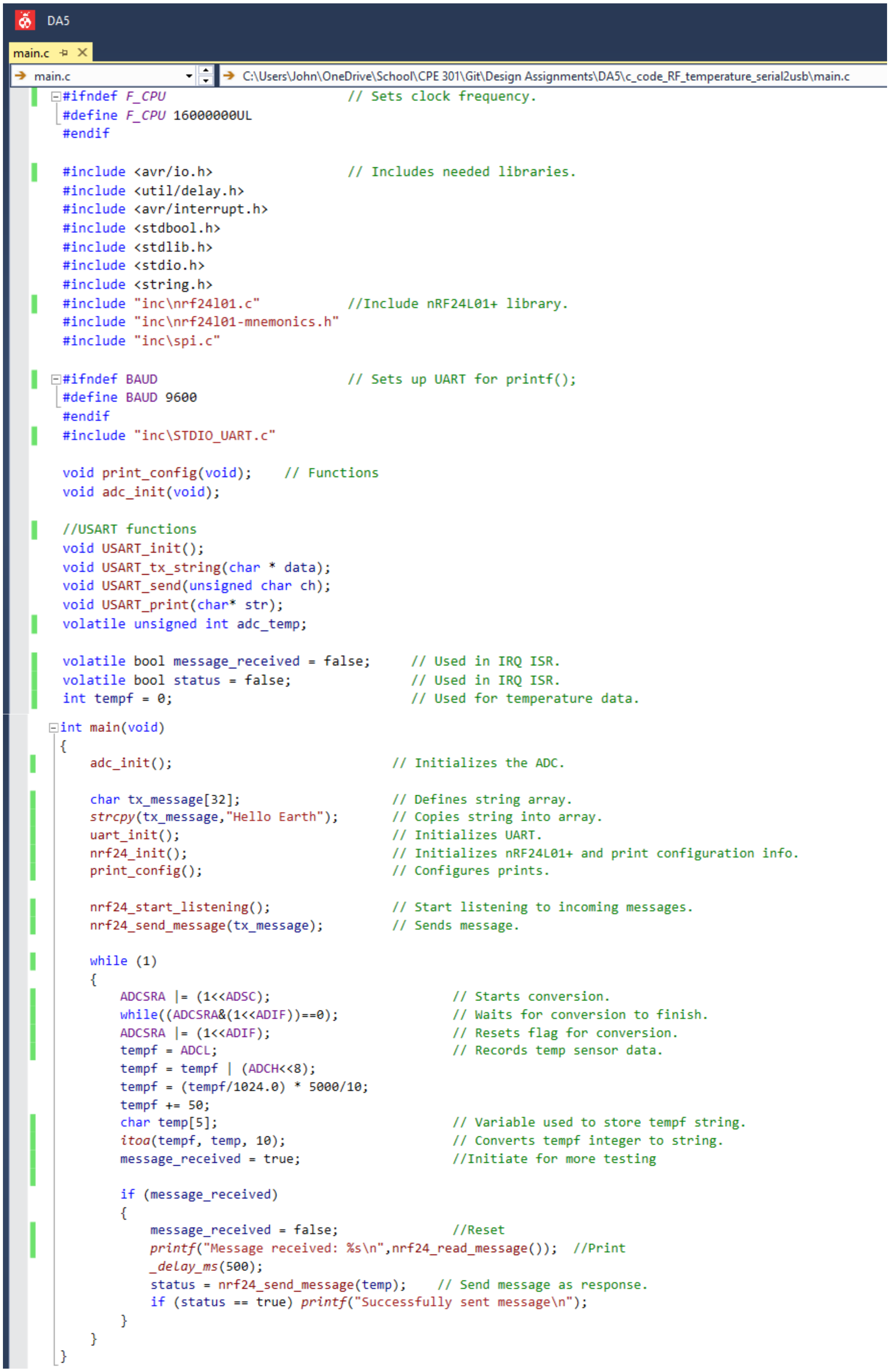
List of Components used:
Atmega328P
Temperature Sensor LM35
Jumper Wires
Bread board
NRF24L01 + RF Module

Block diagram with pins used in the Atmega328P

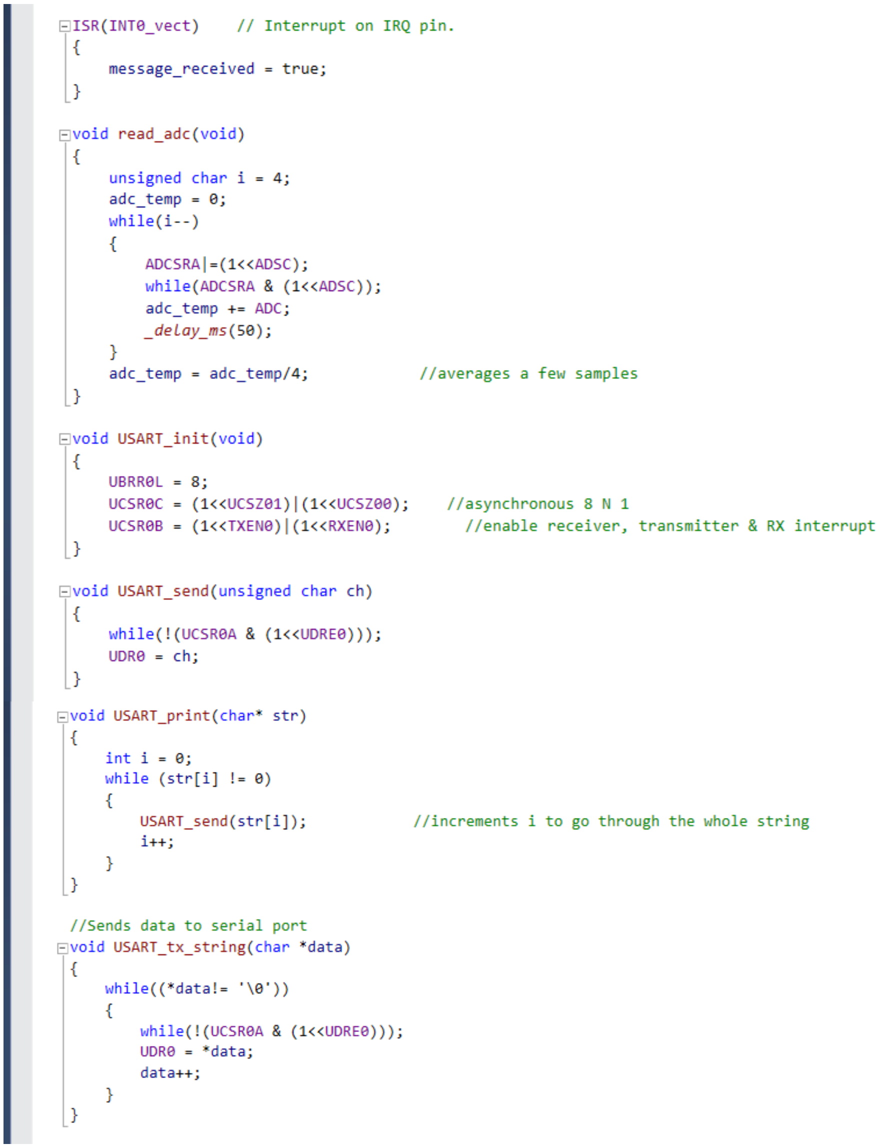


```
// CE connected to PB1
// CSN connected to PB2
// IRQ connected to PD2
```

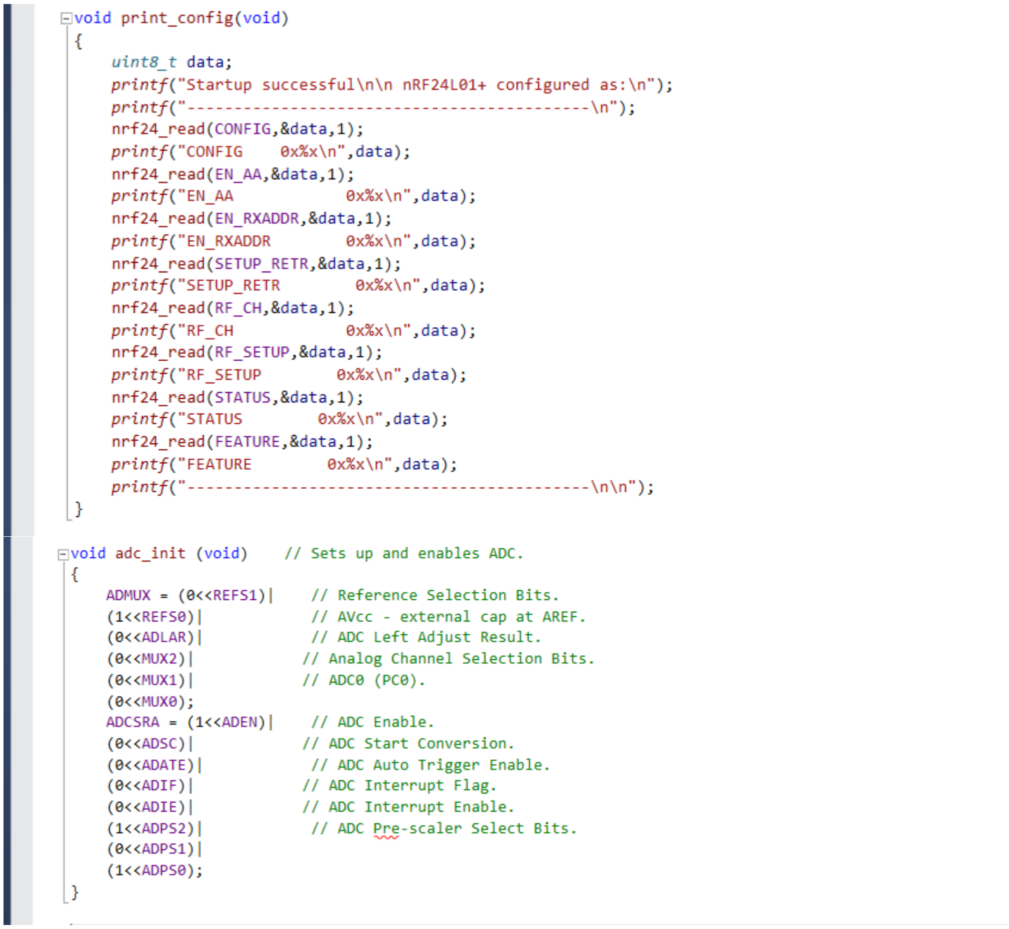
2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A


DA5
main.c
main.c
C:\Users\John\OneDrive\School\CPE 301\Git\Design Assignments\DA5\c_code_RF_temperature_serial2usb\main.c

```c
#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 <stdlib.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);

//USART functions
void USART_init();
void USART_tx_string(char * data);
void USART_send(unsigned char ch);
void USART_print(char* str);
volatile unsigned int adc_temp;

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 Earth");       // 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;
        tempf += 50;
        char temp[5];                               // Variable used to store tempf string.
        itoa(tempf, temp, 10);                      // Converts tempf integer to string.
        message_received = true;                    //Initiate for more testing

        if (message_received)
        {
            message_received = false;               //Reset
            printf("Message received: %s\n",nrf24_read_message());  //Print
            _delay_ms(500);
            status = nrf24_send_message(temp);    // Send message as response.
            if (status == true) printf("Successfully sent message\n");
        }
    }
}
```



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

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

void USART_init(void)
{
    UBRR0L = 8;
    UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);    //asynchronous 8 N 1
    UCSR0B = (1<<TXEN0)|(1<<RXEN0);        //enable receiver, transmitter & RX interrupt
}

void USART_send(unsigned char ch)
{
    while(!(UCSR0A & (1<<UDRE0)));
    UDR0 = ch;
}

void USART_print(char* str)
{
    int i = 0;
    while (str[i] != 0)
    {
        USART_send(str[i]);           //increments i to go through the whole string
        i++;
    }
}

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

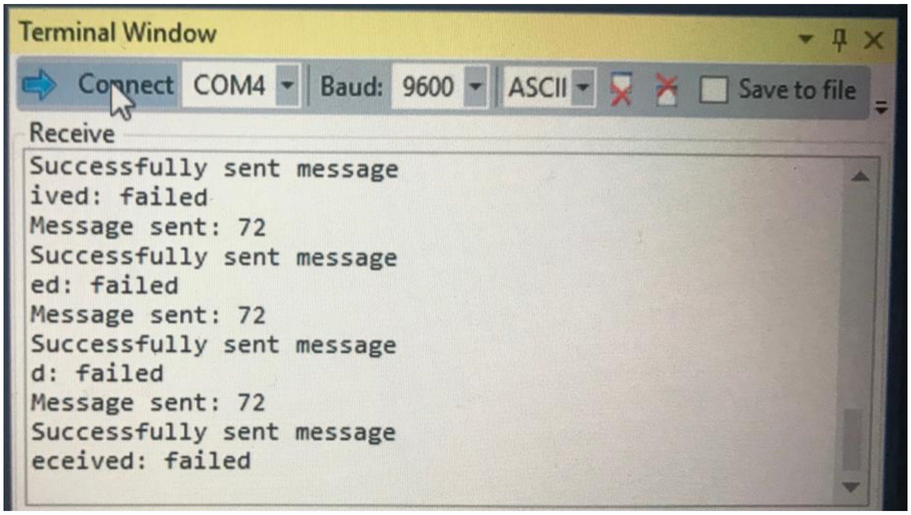


```
void print_config(void)
{
    uint8_t data;
    printf("Startup successful\n\n nRF24L01+ configured as:\n");
    printf("-------------------------------------------\n");
    nrf24_read(CONFIG,&data,1);
    printf("CONFIG    0x%x\n",data);
    nrf24_read(EN_AA,&data,1);
    printf("EN_AA            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);
}
```

3. SCHEMATICS

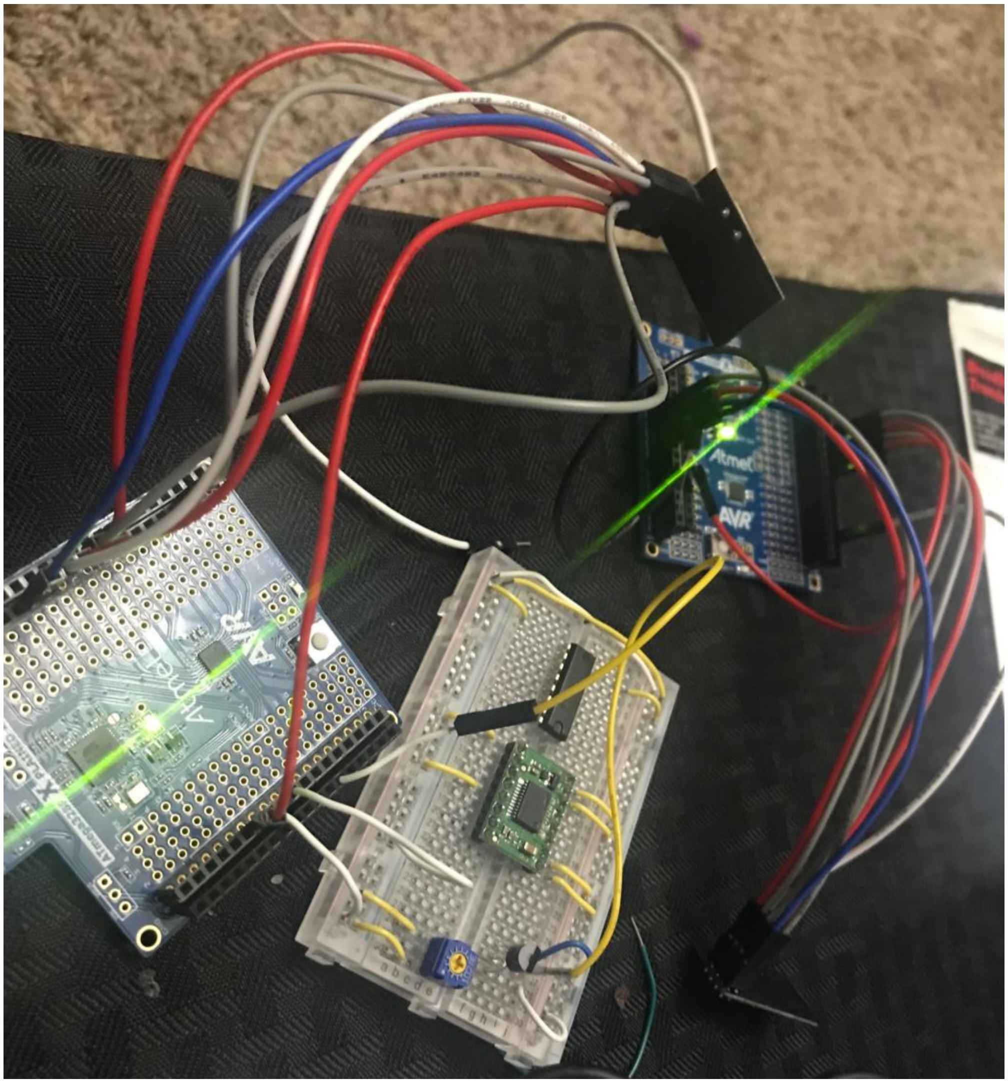
Use fritzing.org

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



```
Successfully sent message
ived: failed
Message sent: 72
Successfully sent message
ed: failed
Message sent: 72
Successfully sent message
d: failed
Message sent: 72
Successfully sent message
eceived: failed
```

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)



6. VIDEO LINKS OF EACH DEMO

N/A

7. GITHUB LINK OF THIS DA

https://github.com/johnduriman/pirahnaplant.git

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

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