Dominique Anguiano

CPE301 – SPRING 2016

Design Assignment 6

**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)** |
| 0. | COMPONENTS LIST |  |  |
| 1. | INITIAL CODE OF TASK 1/A |  |  |
| 2. | LIBRARY USED IN TASK 1/A |  |  |
| 3. | SCHEMATIC |  |  |
| 4. | FLOW CHART |  |  |
| 5. | SCREENSHOT OF EACH DEMO |  |  |
| 6. | VIDEO LINKS OF EACH DEMO |  |  |
| 7. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |

* Atmega328p
* Nokia 5110 GLCD Module
* LM34

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | INITIAL CODE OF TASK 1/A |  |  |

/\*

\* Da 6.c

\*

\* Created: 4/21/2016 6:18:09 PM

\* Author : Dominique

\*/

#define *F\_CPU* 8000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <stdint.h> // needed for uint8\_t

#include <avr/interrupt.h>

#include "nokia5110.h"

volatile *uint32\_t* mathy;

volatile *uint16\_t* ADCvalue; // Global variable, set to volatile if used with ISR

volatile char ones, tens; // Global variable for the characters to be transmit.

void delay1s();

int main(void)

{

nokia\_lcd\_init();

nokia\_lcd\_clear();

nokia\_lcd\_write\_string("Temp in F", 1);

ADMUX = 0; // use ADC0

ADMUX |= (1 << REFS0); // use AVcc as the reference

ADCSRA |= (1 << ADPS2) | (1 << ADPS1); // 64 prescaler for 8Mhz

ADCSRA |= (1 << ADATE); // Set ADC Auto Trigger Enable

ADCSRB = 0; // 0 for free running mode

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

ADCSRA |= (1 << ADIE); // Enable Interrupts

ADCSRA |= (1 << ADSC); // Start the ADC conversion

sei();

while(1)

{

}

}

ISR(ADC\_vect)

{

ADCvalue = ADC; // only need to read the low value for 8 bit

mathy = ADCvalue; // Value to perform the conversion between ADC and temperature

mathy = mathy \* 5; // Math to convert from the given ADC value to a temperature value

mathy = mathy \* 100;

mathy = mathy / 1024;

ones = mathy % 10; // Obtaining the ones digit for the temperature value

tens = mathy / 10; // Obtaining the tens digit for the temperature value

nokia\_lcd\_set\_cursor(0, 10);

nokia\_lcd\_write\_char(tens+'0', 3); // Output the digits

nokia\_lcd\_write\_char(ones+'0', 3); // Output the digits

nokia\_lcd\_write\_string(" :)", 3);

nokia\_lcd\_render();

delay1s(); // Delay for 1 second before converting the next value

}

void delay1s(){

for(int i = 500; i > 0; i--)

*\_delay\_ms*(2);

}

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | LIBRARIES USED FOR TASK 1/A |  |  |

/\* Nokia 5110 LCD AVR Library

\*

\* Copyright (C) 2015 Sergey Denisov.

\* Written by Sergey Denisov aka LittleBuster (DenisovS21@gmail.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public Licence

\* as published by the Free Software Foundation; either version 3

\* of the Licence, or (at your option) any later version.

\*

\* Original library written by SkewPL, http://skew.tk

\*/

#include "nokia5110.h"

#include <avr/pgmspace.h>

#include <avr/io.h>

#include <util/delay.h>

#include "nokia5110\_chars.h"

static struct {

/\* screen byte massive \*/

*uint8\_t* screen[504];

/\* cursor position \*/

*uint8\_t* cursor\_x;

*uint8\_t* cursor\_y;

} nokia\_lcd = {

.cursor\_x = 0,

.cursor\_y = 0

};

/\*\*

\* Sending data to LCD

\* @bytes: data

\* @is\_data: transfer mode: 1 - data; 0 - command;

\*/

static void write(*uint8\_t* bytes, *uint8\_t* is\_data)

{

register *uint8\_t* i;

/\* Enable controller \*/

PORT\_LCD &= ~(1 << LCD\_SCE);

/\* We are sending data \*/

if (is\_data)

PORT\_LCD |= (1 << LCD\_DC);

/\* We are sending commands \*/

else

PORT\_LCD &= ~(1 << LCD\_DC);

/\* Send bytes \*/

for (i = 0; i < 8; i++) {

/\* Set data pin to byte state \*/

if ((bytes >> (7-i)) & 0x01)

PORT\_LCD |= (1 << LCD\_DIN);

else

PORT\_LCD &= ~(1 << LCD\_DIN);

/\* Blink clock \*/

PORT\_LCD |= (1 << LCD\_CLK);

PORT\_LCD &= ~(1 << LCD\_CLK);

}

/\* Disable controller \*/

PORT\_LCD |= (1 << LCD\_SCE);

}

static void write\_cmd(*uint8\_t* cmd)

{

write(cmd, 0);

}

static void write\_data(*uint8\_t* data)

{

write(data, 1);

}

/\*

\* Public functions

\*/

void nokia\_lcd\_init(void)

{

register unsigned i;

/\* Set pins as output \*/

DDR\_LCD |= (1 << LCD\_SCE);

DDR\_LCD |= (1 << LCD\_RST);

DDR\_LCD |= (1 << LCD\_DC);

DDR\_LCD |= (1 << LCD\_DIN);

DDR\_LCD |= (1 << LCD\_CLK);

/\* Reset display \*/

PORT\_LCD |= (1 << LCD\_RST);

PORT\_LCD |= (1 << LCD\_SCE);

*\_delay\_ms*(10);

PORT\_LCD &= ~(1 << LCD\_RST);

*\_delay\_ms*(70);

PORT\_LCD |= (1 << LCD\_RST);

/\*

\* Initialize display

\*/

/\* Enable controller \*/

PORT\_LCD &= ~(1 << LCD\_SCE);

/\* -LCD Extended Commands mode- \*/

write\_cmd(0x21);

/\* LCD bias mode 1:48 \*/

write\_cmd(0x13);

/\* Set temperature coefficient \*/

write\_cmd(0x06);

/\* Default VOP (3.06 + 66 \* 0.06 = 7V) \*/

write\_cmd(0xC2);

/\* Standard Commands mode, powered down \*/

write\_cmd(0x20);

/\* LCD in normal mode \*/

write\_cmd(0x09);

/\* Clear LCD RAM \*/

write\_cmd(0x80);

write\_cmd(LCD\_CONTRAST);

for (i = 0; i < 504; i++)

write\_data(0x00);

/\* Activate LCD \*/

write\_cmd(0x08);

write\_cmd(0x0C);

}

void nokia\_lcd\_clear(void)

{

register unsigned i;

/\* Set column and row to 0 \*/

write\_cmd(0x80);

write\_cmd(0x40);

/\*Cursor too \*/

nokia\_lcd.cursor\_x = 0;

nokia\_lcd.cursor\_y = 0;

/\* Clear everything (504 bytes = 84cols \* 48 rows / 8 bits) \*/

for(i = 0;i < 504; i++)

nokia\_lcd.screen[i] = 0x00;

}

void nokia\_lcd\_power(*uint8\_t* on)

{

write\_cmd(on ? 0x20 : 0x24);

}

void nokia\_lcd\_set\_pixel(*uint8\_t* x, *uint8\_t* y, *uint8\_t* value)

{

*uint8\_t* \*byte = &nokia\_lcd.screen[y/8\*84+x];

if (value)

\*byte |= (1 << (y % 8));

else

\*byte &= ~(1 << (y %8 ));

}

void nokia\_lcd\_write\_char(char code, *uint8\_t* scale)

{

register *uint8\_t* x, y;

for (x = 0; x < 5\*scale; x++)

for (y = 0; y < 7\*scale; y++)

if (pgm\_read\_byte(&CHARSET[code-32][x/scale]) & (1 << y/scale))

nokia\_lcd\_set\_pixel(nokia\_lcd.cursor\_x + x, nokia\_lcd.cursor\_y + y, 1);

else

nokia\_lcd\_set\_pixel(nokia\_lcd.cursor\_x + x, nokia\_lcd.cursor\_y + y, 0);

nokia\_lcd.cursor\_x += 5\*scale + 1;

if (nokia\_lcd.cursor\_x >= 84) {

nokia\_lcd.cursor\_x = 0;

nokia\_lcd.cursor\_y += 7\*scale + 1;

}

if (nokia\_lcd.cursor\_y >= 48) {

nokia\_lcd.cursor\_x = 0;

nokia\_lcd.cursor\_y = 0;

}

}

void nokia\_lcd\_write\_string(const char \*str, *uint8\_t* scale)

{

while(\*str)

nokia\_lcd\_write\_char(\*str++, scale);

}

void nokia\_lcd\_set\_cursor(*uint8\_t* x, *uint8\_t* y)

{

nokia\_lcd.cursor\_x = x;

nokia\_lcd.cursor\_y = y;

}

void nokia\_lcd\_render(void)

{

register unsigned i;

/\* Set column and row to 0 \*/

write\_cmd(0x80);

write\_cmd(0x40);

/\* Write screen to display \*/

for (i = 0; i < 504; i++)

write\_data(nokia\_lcd.screen[i]);

}

/\* Nokia 5110 LCD AVR Library

\*

\* Copyright (C) 2015 Sergey Denisov.

\* Written by Sergey Denisov aka LittleBuster (DenisovS21@gmail.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public Licence

\* as published by the Free Software Foundation; either version 3

\* of the Licence, or (at your option) any later version.

\*

\* Original library written by SkewPL, http://skew.tk

\*/

#ifndef \_\_NOKIA\_5110\_H\_\_

#define \_\_NOKIA\_5110\_H\_\_

#include <avr/pgmspace.h>

#include <stdint.h>

/\*

\* LCD's port

\*/

#define PORT\_LCD PORTB

#define DDR\_LCD DDRB

/\*

\* LCD's pins

\*/

#define LCD\_SCE 0

#define LCD\_RST 1

#define LCD\_DC 2

#define LCD\_DIN 3

#define LCD\_CLK 5

#define LCD\_CONTRAST 0x40

/\*

\* Must be called once before any other function, initializes display

\*/

void nokia\_lcd\_init(void);

/\*

\* Clear screen

\*/

void nokia\_lcd\_clear(void);

/\*\*

\* Power of display

\* @lcd: lcd nokia struct

\* @on: 1 - on; 0 - off;

\*/

void nokia\_lcd\_power(*uint8\_t* on);

/\*\*

\* Set single pixel

\* @x: horizontal pozition

\* @y: vertical position

\* @value: show/hide pixel

\*/

void nokia\_lcd\_set\_pixel(*uint8\_t* x, *uint8\_t* y, *uint8\_t* value);

/\*\*

\* Draw single char with 1-6 scale

\* @code: char code

\* @scale: size of char

\*/

void nokia\_lcd\_write\_char(char code, *uint8\_t* scale);

/\*\*

\* Draw string. Example: writeString("abc",3);

\* @str: sending string

\* @scale: size of text

\*/

void nokia\_lcd\_write\_string(const char \*str, *uint8\_t* scale);

/\*\*

\* Set cursor position

\* @x: horizontal position

\* @y: vertical position

\*/

void nokia\_lcd\_set\_cursor(*uint8\_t* x, *uint8\_t* y);

/\*

\* Render screen to display

\*/

void nokia\_lcd\_render(void);

#endif

#include <avr/pgmspace.h>

const *uint8\_t* CHARSET[][5] PROGMEM = {

{ 0x00, 0x00, 0x00, 0x00, 0x00 }, // 20 space

{ 0x00, 0x00, 0x5f, 0x00, 0x00 }, // 21 !

{ 0x00, 0x07, 0x00, 0x07, 0x00 }, // 22 "

{ 0x14, 0x7f, 0x14, 0x7f, 0x14 }, // 23 #

{ 0x24, 0x2a, 0x7f, 0x2a, 0x12 }, // 24 $

{ 0x23, 0x13, 0x08, 0x64, 0x62 }, // 25 %

{ 0x36, 0x49, 0x55, 0x22, 0x50 }, // 26 &

{ 0x00, 0x05, 0x03, 0x00, 0x00 }, // 27 '

{ 0x00, 0x1c, 0x22, 0x41, 0x00 }, // 28 (

{ 0x00, 0x41, 0x22, 0x1c, 0x00 }, // 29 )

{ 0x14, 0x08, 0x3e, 0x08, 0x14 }, // 2a \*

{ 0x08, 0x08, 0x3e, 0x08, 0x08 }, // 2b +

{ 0x00, 0x50, 0x30, 0x00, 0x00 }, // 2c ,

{ 0x08, 0x08, 0x08, 0x08, 0x08 }, // 2d -

{ 0x00, 0x60, 0x60, 0x00, 0x00 }, // 2e .

{ 0x20, 0x10, 0x08, 0x04, 0x02 }, // 2f /

{ 0x3e, 0x51, 0x49, 0x45, 0x3e }, // 30 0

{ 0x00, 0x42, 0x7f, 0x40, 0x00 }, // 31 1

{ 0x42, 0x61, 0x51, 0x49, 0x46 }, // 32 2

{ 0x21, 0x41, 0x45, 0x4b, 0x31 }, // 33 3

{ 0x18, 0x14, 0x12, 0x7f, 0x10 }, // 34 4

{ 0x27, 0x45, 0x45, 0x45, 0x39 }, // 35 5

{ 0x3c, 0x4a, 0x49, 0x49, 0x30 }, // 36 6

{ 0x01, 0x71, 0x09, 0x05, 0x03 }, // 37 7

{ 0x36, 0x49, 0x49, 0x49, 0x36 }, // 38 8

{ 0x06, 0x49, 0x49, 0x29, 0x1e }, // 39 9

{ 0x00, 0x36, 0x36, 0x00, 0x00 }, // 3a :

{ 0x00, 0x56, 0x36, 0x00, 0x00 }, // 3b ;

{ 0x08, 0x14, 0x22, 0x41, 0x00 }, // 3c <

{ 0x14, 0x14, 0x14, 0x14, 0x14 }, // 3d =

{ 0x00, 0x41, 0x22, 0x14, 0x08 }, // 3e >

{ 0x02, 0x01, 0x51, 0x09, 0x06 }, // 3f ?

{ 0x32, 0x49, 0x79, 0x41, 0x3e }, // 40 @

{ 0x7e, 0x11, 0x11, 0x11, 0x7e }, // 41 A

{ 0x7f, 0x49, 0x49, 0x49, 0x36 }, // 42 B

{ 0x3e, 0x41, 0x41, 0x41, 0x22 }, // 43 C

{ 0x7f, 0x41, 0x41, 0x22, 0x1c }, // 44 D

{ 0x7f, 0x49, 0x49, 0x49, 0x41 }, // 45 E

{ 0x7f, 0x09, 0x09, 0x09, 0x01 }, // 46 F

{ 0x3e, 0x41, 0x49, 0x49, 0x7a }, // 47 G

{ 0x7f, 0x08, 0x08, 0x08, 0x7f }, // 48 H

{ 0x00, 0x41, 0x7f, 0x41, 0x00 }, // 49 I

{ 0x20, 0x40, 0x41, 0x3f, 0x01 }, // 4a J

{ 0x7f, 0x08, 0x14, 0x22, 0x41 }, // 4b K

{ 0x7f, 0x40, 0x40, 0x40, 0x40 }, // 4c L

{ 0x7f, 0x02, 0x0c, 0x02, 0x7f }, // 4d M

{ 0x7f, 0x04, 0x08, 0x10, 0x7f }, // 4e N

{ 0x3e, 0x41, 0x41, 0x41, 0x3e }, // 4f O

{ 0x7f, 0x09, 0x09, 0x09, 0x06 }, // 50 P

{ 0x3e, 0x41, 0x51, 0x21, 0x5e }, // 51 Q

{ 0x7f, 0x09, 0x19, 0x29, 0x46 }, // 52 R

{ 0x46, 0x49, 0x49, 0x49, 0x31 }, // 53 S

{ 0x01, 0x01, 0x7f, 0x01, 0x01 }, // 54 T

{ 0x3f, 0x40, 0x40, 0x40, 0x3f }, // 55 U

{ 0x1f, 0x20, 0x40, 0x20, 0x1f }, // 56 V

{ 0x3f, 0x40, 0x38, 0x40, 0x3f }, // 57 W

{ 0x63, 0x14, 0x08, 0x14, 0x63 }, // 58 X

{ 0x07, 0x08, 0x70, 0x08, 0x07 }, // 59 Y

{ 0x61, 0x51, 0x49, 0x45, 0x43 }, // 5a Z

{ 0x00, 0x7f, 0x41, 0x41, 0x00 }, // 5b [

{ 0x02, 0x04, 0x08, 0x10, 0x20 }, // 5c backslash

{ 0x00, 0x41, 0x41, 0x7f, 0x00 }, // 5d ]

{ 0x04, 0x02, 0x01, 0x02, 0x04 }, // 5e ^

{ 0x40, 0x40, 0x40, 0x40, 0x40 }, // 5f \_

{ 0x00, 0x01, 0x02, 0x04, 0x00 }, // 60 `

{ 0x20, 0x54, 0x54, 0x54, 0x78 }, // 61 a

{ 0x7f, 0x48, 0x44, 0x44, 0x38 }, // 62 b

{ 0x38, 0x44, 0x44, 0x44, 0x20 }, // 63 c

{ 0x38, 0x44, 0x44, 0x48, 0x7f }, // 64 d

{ 0x38, 0x54, 0x54, 0x54, 0x18 }, // 65 e

{ 0x08, 0x7e, 0x09, 0x01, 0x02 }, // 66 f

{ 0x0c, 0x52, 0x52, 0x52, 0x3e }, // 67 g

{ 0x7f, 0x08, 0x04, 0x04, 0x78 }, // 68 h

{ 0x00, 0x44, 0x7d, 0x40, 0x00 }, // 69 i

{ 0x20, 0x40, 0x44, 0x3d, 0x00 }, // 6a j

{ 0x7f, 0x10, 0x28, 0x44, 0x00 }, // 6b k

{ 0x00, 0x41, 0x7f, 0x40, 0x00 }, // 6c l

{ 0x7c, 0x04, 0x18, 0x04, 0x78 }, // 6d m

{ 0x7c, 0x08, 0x04, 0x04, 0x78 }, // 6e n

{ 0x38, 0x44, 0x44, 0x44, 0x38 }, // 6f o

{ 0x7c, 0x14, 0x14, 0x14, 0x08 }, // 70 p

{ 0x08, 0x14, 0x14, 0x18, 0x7c }, // 71 q

{ 0x7c, 0x08, 0x04, 0x04, 0x08 }, // 72 r

{ 0x48, 0x54, 0x54, 0x54, 0x20 }, // 73 s

{ 0x04, 0x3f, 0x44, 0x40, 0x20 }, // 74 t

{ 0x3c, 0x40, 0x40, 0x20, 0x7c }, // 75 u

{ 0x1c, 0x20, 0x40, 0x20, 0x1c }, // 76 v

{ 0x3c, 0x40, 0x30, 0x40, 0x3c }, // 77 w

{ 0x44, 0x28, 0x10, 0x28, 0x44 }, // 78 x

{ 0x0c, 0x50, 0x50, 0x50, 0x3c }, // 79 y

{ 0x44, 0x64, 0x54, 0x4c, 0x44 }, // 7a z

{ 0x00, 0x08, 0x36, 0x41, 0x00 }, // 7b {

{ 0x00, 0x00, 0x7f, 0x00, 0x00 }, // 7c |

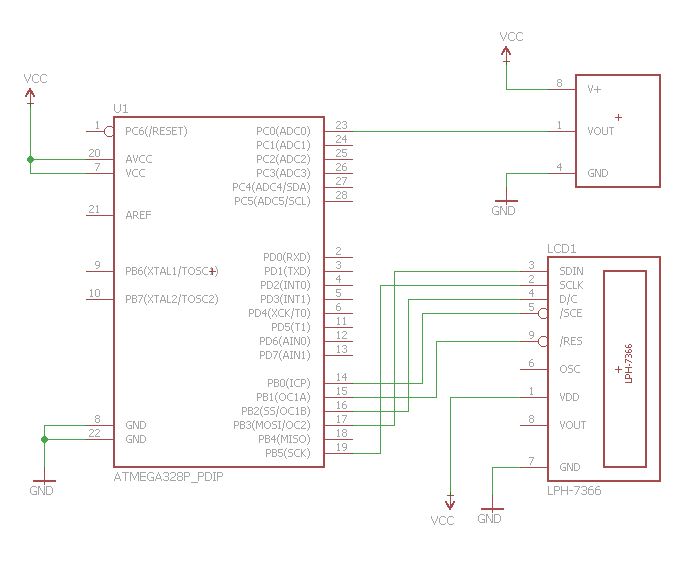
{ 0x00, 0x41, 0x36, 0x08, 0x00 }, // 7d }

{ 0x10, 0x08, 0x08, 0x10, 0x08 }, // 7e ~

{ 0x00, 0x00, 0x00, 0x00, 0x00 } // 7f

};

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3. | SCHEMATICS |  |  |  |



|  |  |  |  |
| --- | --- | --- | --- |
| 4. | FLOW CHART |  |  |

Initialize the ADC and Nokia LCD

Check if ADC Interrupt Flag is set

Wait

Not Set

Yes, it is set

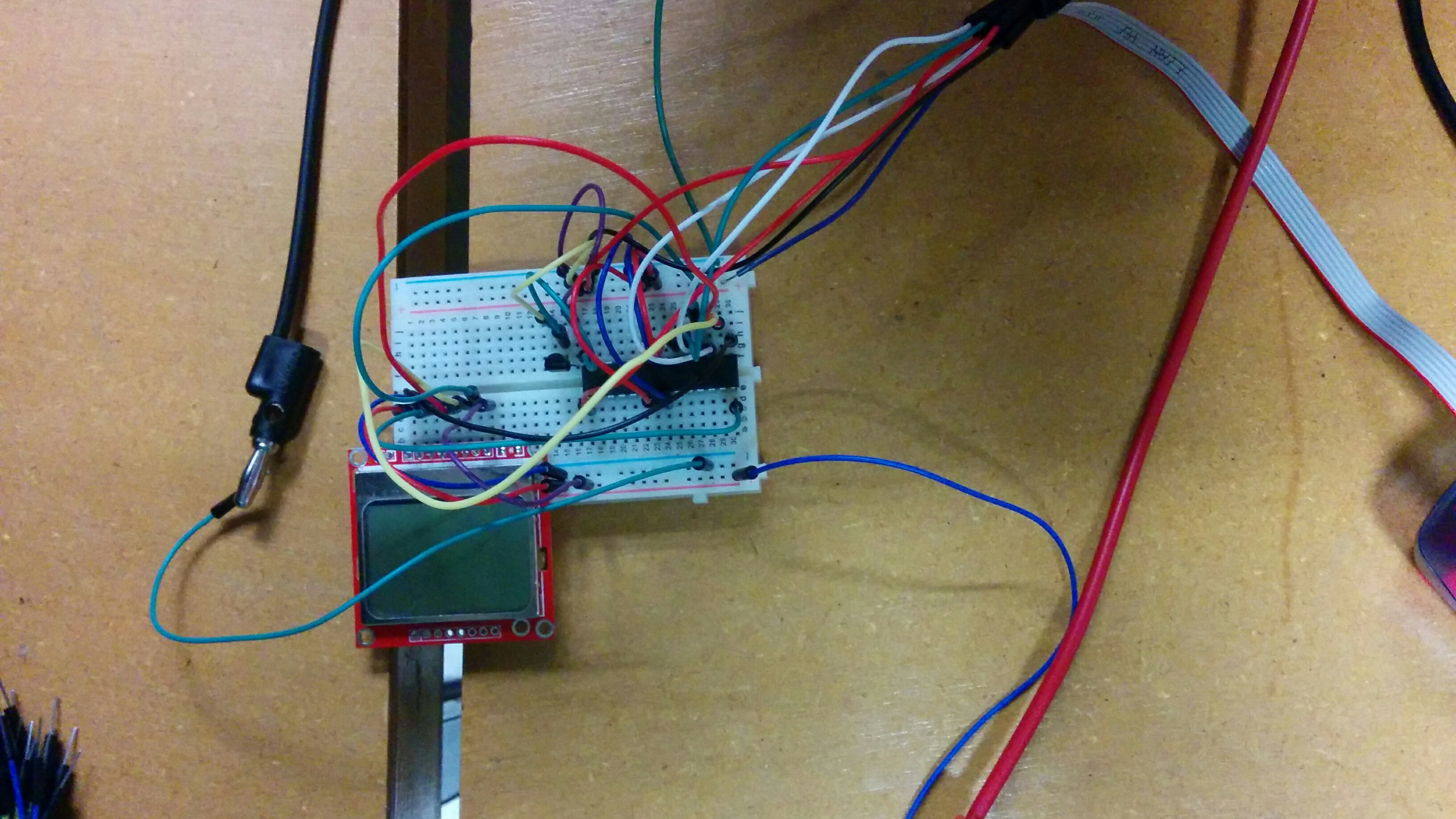
Head to Interrupt and convert ADC value to temperature.

Output the temperature to the Nokia LCD.

Wait 1 second and exit interrupt

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | SCREENSHOT OF EACH DEMO |  |  |

TASK 1/A: Nokia LCD connected to Atmega328p



|  |  |  |  |
| --- | --- | --- | --- |
| 6. | VIDEO LINKS OF EACH DEMO |  |  |
| <https://www.youtube.com/watch?v=bT6w2a4MR8s> | | | |
| 7. | GOOGLECODE LINK OF THE DA |  |  |
| <https://github.com/Anguian3/anguian3-submissions> | | | |

**Student Academic Misconduct Policy**

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

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

Dominique Anguiano