CPE301 – SPRING 2020

Design Assignment 5

Student Name: Xianjie Cao

Student #: 5004222179

Student Email: [caox2@unlv.nevada.edu](mailto:caox2@unlv.nevada.edu)

Primary Github address: <https://github.com/c1029324620/Mocha.git>

Directory: Mocha/DesignAssignments/Lab5

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

Atmel Studio 7: Debugger, simulator, and assembler.

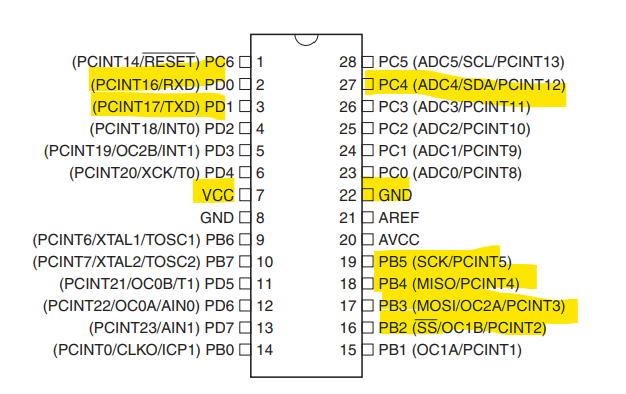
Atmega328pb X-mini

Jumper wires

Multifunctional Shield

LM35

DS18b20



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

**Task 1:**

#include <avr/io.h>

#include <stdio.h>

#ifndef *F\_CPU*

#define *F\_CPU* 16000000UL

#endif

#ifndef BAUD

#define BAUD 9600

#endif

#include <stdlib.h>

#include <util/delay.h>

#include <util/setbaud.h>

#define LATCH 4 /\* PD4 RCK \*/

#define CLOCK 7 /\* PD7 SRCK \*/

#define DATA 0 /\* PB0 SER IN \*/

#define LSBFIRST 0

#define MSBFIRST 1

unsigned int adc\_value;

//initialized adc module

void adc\_int(void)

{

//setup and enable ADC

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) | // ADC4(PC4)

(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);

}

void adc\_read(void)

{

ADCSRA |= (1 << ADSC); //enable start conversion

while(ADCSRA & (1<< ADSC)); //wait

adc\_value = ADC; // LM35 value stored in temp.

}

/\* Segment byte maps for numbers 0 to 9 \*/

const *uint8\_t* SEGMENT\_MAP[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99,

0x92, 0x82, 0xF8, 0X80, 0X90};

/\* Byte maps to select digit 1 to 4 \*/

const *uint8\_t* SEGMENT\_SELECT[] = {0xF1, 0xF2, 0xF4, 0xF8};

void shift\_out\_init(void) {

DDRB |= (1 << DATA);

DDRD |= (1 << CLOCK) | (1 << LATCH);

}

void uart\_init(void) {

UBRR0H = *UBRRH\_VALUE*;

UBRR0L = *UBRRL\_VALUE*;

#if *USE\_2X*

UCSR0A |= \_BV(U2X0);

#else

UCSR0A &= ~(\_BV(U2X0));

#endif

UCSR0C = \_BV(UCSZ01) | \_BV(UCSZ00); /\* 8-bit data \*/

UCSR0B = \_BV(RXEN0) | \_BV(TXEN0); /\* Enable RX and TX \*/

}

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;

}

int uart\_getchar(*FILE* \*stream) {

loop\_until\_bit\_is\_set(UCSR0A, RXC0);

return UDR0;

}

*FILE* uart\_output = *FDEV\_SETUP\_STREAM*(uart\_putchar, *NULL*, *\_FDEV\_SETUP\_WRITE*);

*FILE* uart\_input = *FDEV\_SETUP\_STREAM*(*NULL*, uart\_getchar, *\_FDEV\_SETUP\_READ*);

void shift\_out(*uint8\_t* indata) {

for (*uint8\_t* i = 0; i < 8; i++) {

/\* Write bit to data port. \*/

if (0 == (indata & \_BV(7 - i))) {

// digital\_write(SHIFT\_OUT\_DATA, LOW);

PORTB &= (0 << DATA);

} else {

// digital\_write(SHIFT\_OUT\_DATA, HIGH);

PORTB |= (1 << DATA);

}

/\* Pulse clock to write next bit. \*/

PORTD |= (1 << CLOCK);

PORTD &= (0 << CLOCK);

}

}

void shiftOut(*uint8\_t* dataPin, *uint8\_t* clockPin, *uint8\_t* bitOrder,

*uint8\_t* val) {

*uint8\_t* i;

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

if (bitOrder == LSBFIRST)

dataPin |= !!(val & (1 << i));

else

dataPin |= !!(val & (1 << (7 - i)));

PORTD |= (1 << CLOCK);

PORTD &= (0 << CLOCK);

}

}

void shift\_out\_latch(void) {

PORTD &= (0 << LATCH);

PORTD |= (1 << LATCH);

}

int main(void) {

shift\_out\_init();

uart\_init();

*stdout* = &uart\_output;

*stdin* = &uart\_input;

char binary[17];

adc\_int();

while (1) {

adc\_read();

int first\_digi = adc\_value % 10; //get the first digit

int second\_digi = adc\_value / 10; //get the ten digit

int third\_digi = adc\_value / 100; //get the hundreds digit

PORTD &= (0<< LATCH);

shift\_out(SEGMENT\_MAP[third\_digi]);

shift\_out(0xF2);

PORTD |= (1<< LATCH);

PORTD &= (0<< LATCH);

shift\_out(SEGMENT\_MAP[second\_digi]);

shift\_out(0xF4);

PORTD |= (1<< LATCH);

PORTD &= (0<< LATCH);

shift\_out(SEGMENT\_MAP[first\_digi]);

shift\_out(0xF8);

PORTD |= (1<< LATCH);

}

return 0;

}

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**

**Task 2:**

/\*

\* SPI\_ShiftRegister.c

\*

\* Created: 4/2/2020 12:55:58 AM

\* Author : VenkatesanMuthukumar

\*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#define SHIFT\_REGISTER DDRB

#define SHIFT\_PORT PORTB

#define DATA (1<<PB3) //MOSI (SI)

#define LATCH (1<<PB2) //SS (RCK)

#define CLOCK (1<<PB5) //SCK (SCK)

/\* Note the mapping for Multifunctional shield

Name - Board - Shield

DATA/MOSI PB3 (11) PB0 (8)

LATCH/SS PB2 (10) PD4 (4)

CLOCK/SCK PB5 (13) PD7 (7)

-/MISO PB4 (12) PB1 (9)

\*/

*uint8\_t* adc\_value;

/\* Segment byte maps for numbers 0 to 9 \*/

const *uint8\_t* SEGMENT\_MAP[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99,

0x92, 0x82, 0xF8, 0X80, 0X90};

/\* Byte maps to select digit 1 to 4 \*/

const *uint8\_t* SEGMENT\_SELECT[] = {0xF1, 0xF2, 0xF4, 0xF8};

//initialized adc module

void adc\_int(void)

{

//setup and enable ADC

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) | // ADC4(PC4)

(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);

}

void adc\_read(void)

{

ADCSRA |= (1 << ADSC); //enable start conversion

while(ADCSRA & (1<< ADSC)); //wait

adc\_value = ADC; // LM35 value stored in temp.

}

int main(void)

{

adc\_int(); //initialize the adc module

while(1)

{

adc\_read();

SHIFT\_REGISTER |= (DATA | LATCH | CLOCK); //Set control pins as outputs

SHIFT\_PORT &= ~(DATA | LATCH | CLOCK); //Set control pins low

SPCR0 = (1<<SPE) | (1<<MSTR); //Start SPI as Master

SHIFT\_PORT &= ~LATCH;

//Shift in some data

SPDR0 = SEGMENT\_MAP[adc\_value % 10]; //This should light alternating LEDs

//Wait for SPI process to finish

while(!(SPSR0 & (1<<SPIF)));

//Shift in some more data since I have two shift registers hooked up

SPDR0 = 0xF8; //This should light alternating LEDs

//Wait for SPI process to finish

while(!(SPSR0 & (1<<SPIF)));

//Toggle latch to copy data to the storage register

SHIFT\_PORT |= LATCH;

SHIFT\_PORT &= ~LATCH;

SPDR0 = SEGMENT\_MAP[adc\_value /10]; //This should light alternating LEDs

//Wait for SPI process to finish

while(!(SPSR0 & (1<<SPIF)));

//Shift in some more data since I have two shift registers hooked up

SPDR0 = 0xF4; //This should light alternating LEDs

//Wait for SPI process to finish

while(!(SPSR0 & (1<<SPIF)));

//Toggle latch to copy data to the storage register

SHIFT\_PORT |= LATCH;

SHIFT\_PORT &= ~LATCH;

SPDR0 = SEGMENT\_MAP[adc\_value /100]; //This should light alternating LEDs

//Wait for SPI process to finish

while(!(SPSR0 & (1<<SPIF)));

//Shift in some more data since I have two shift registers hooked up

SPDR0 = 0xF2; //This should light alternating LEDs

//Wait for SPI process to finish

while(!(SPSR0 & (1<<SPIF)));

//Toggle latch to copy data to the storage register

SHIFT\_PORT |= LATCH;

SHIFT\_PORT &= ~LATCH;

//Loop forever

}

**Task 3:**

//Define MCU CPU Freq for the time delay functions

#ifndef F\_CPU

#define F\_CPU 16000000UL

#endif

#include <stdlib.h>

#include <string.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include "ds18b20.h"

#include "uart.h"

int main(void) {

char printbuff[100];

double d = 0;

//init uart

USART\_Init( 9600);

//init interrupt

sei();

while(1) {

d = ds18b20\_gettemp();

*dtostrf*(d, 10, 3, printbuff);

USART\_SendString("Temperature: ");

USART\_SendString(printbuff);

USART\_SendString("\r\n");

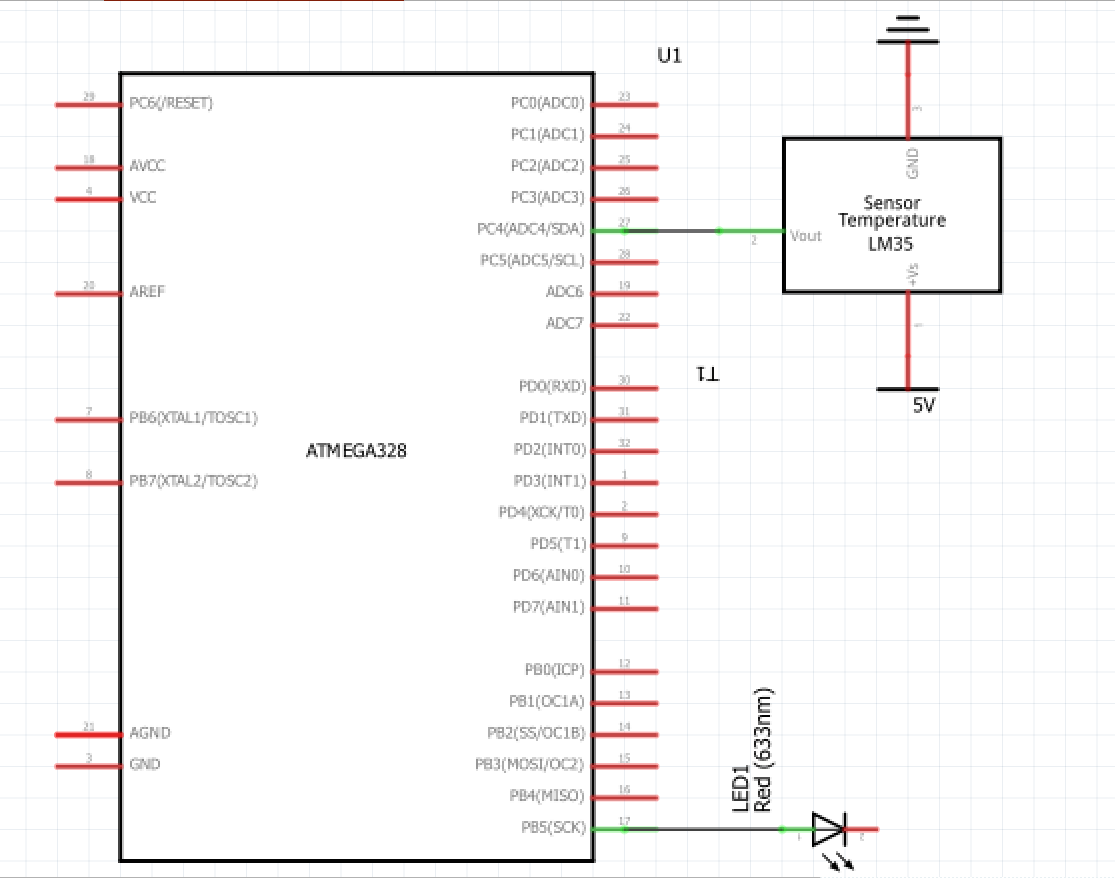
*\_delay\_ms*(500);

}

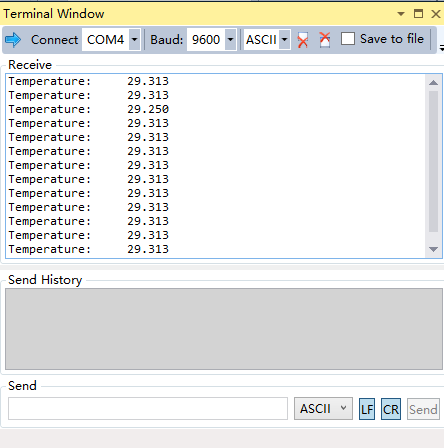
return 0;

}

1. **SCHEMATICS**

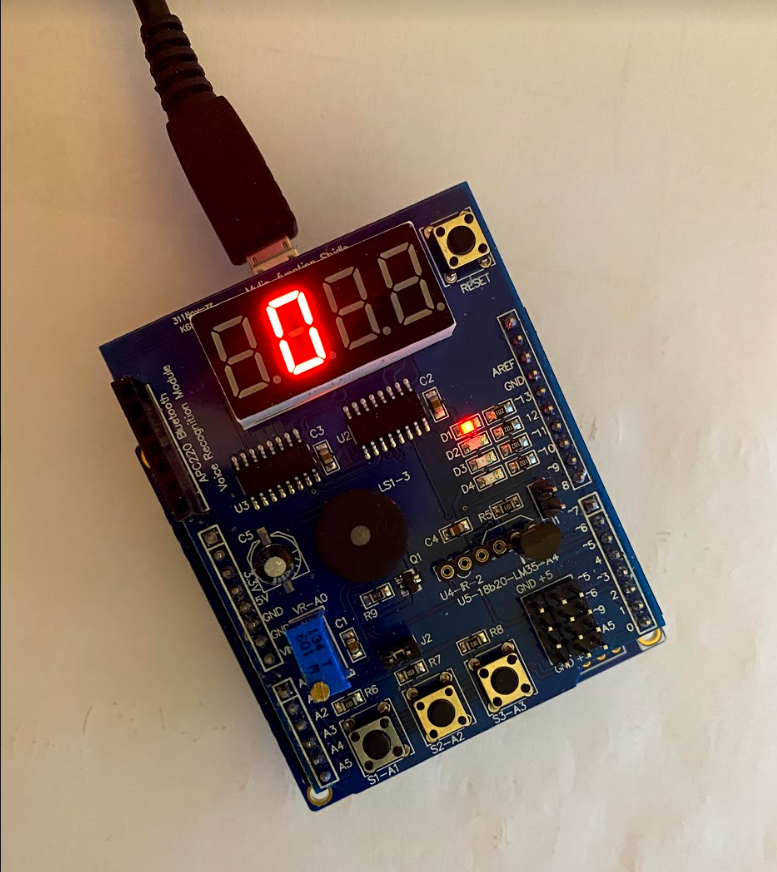


1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

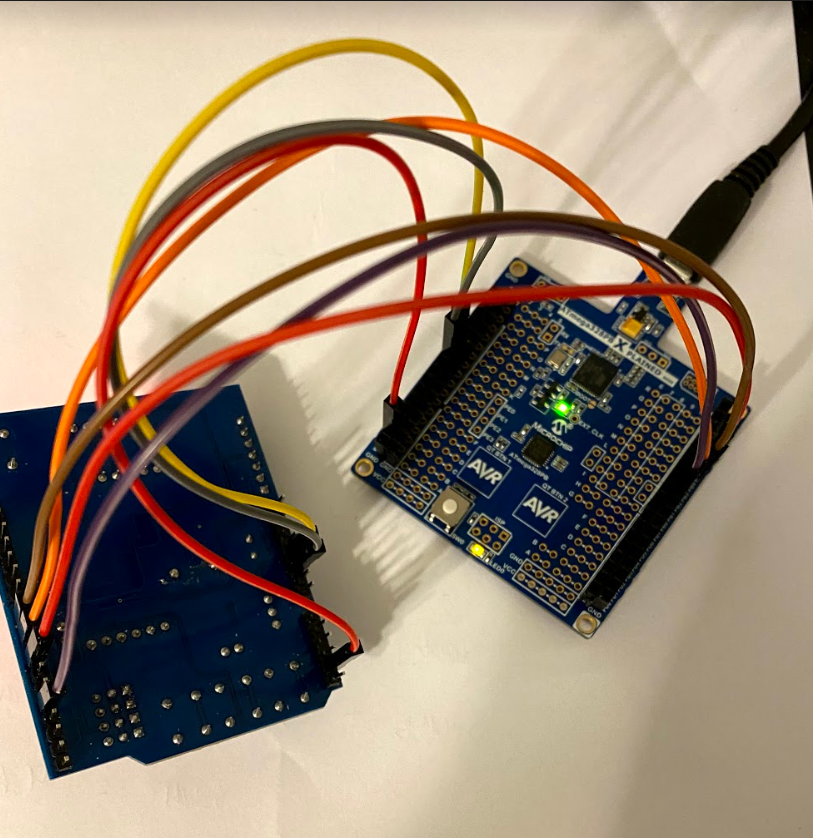


1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

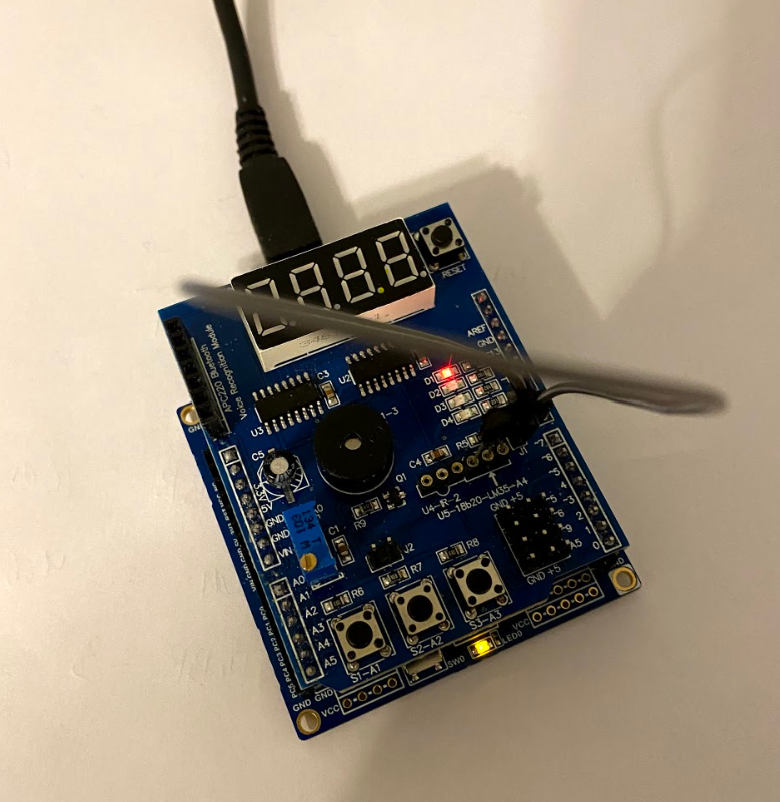
Task 1:



Task 2:



Task 3:



1. **VIDEO LINKS OF EACH DEMO**

Task 1: <https://youtu.be/KI4gFGT-7sg>

Task 2: <https://youtu.be/A_D5ACGhk4k>

Task 3: <https://youtu.be/2aFyYCPRc7M>

1. **GITHUB LINK OF THIS DA**

**Student Academic Misconduct Policy**

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

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

Xianjie Cao