Assignment On:

CSE3027.1 : Introduction to Embedded Systems

Submitted To:

Faculty Name : Rezwan Khan
Department of English,Southeast University.

Submitted By:

Name	ID	Group Name
1.Md. AL Emran	2017000000030	Alpha
2.Name: Al-Amin Hosain	2016100000142	Alpha
3.Mohammad Al Imran Sagor	2015200000062	Alpha

SOUTHEAST UNIVERSITY

Meeting the Challenges of Time

MID EXAM CODE

1.Naive Automatic Sun Tracker

```
#include <avr/io.h>

#include <util/delay.h>

#include <stdio.h>

#define F_CPU 16000000UL

#define FOSC 16000000 /**< Clock speed for UBRR calculation. refer page 179 of 328p datasheet. */

#define BAUD 9600 /**< Baud Rate in bps. refer page 179 of 328p datasheet. */

#define MYUBRR FOSC/16/BAUD-1

int result;

void USART_init(unsigned int ubrr) {

    UCSR0C = (0 << USBS0) | (3 << UCSZ00); /// Step 1. Set UCSR0C in Asynchronous mode, no parity, 1 stop bit, 8 data bits

    UCSR0A = 0b00000000; /// Step 2. Set UCSR0A in Normal speed, disable multi-proc

    UBRR0H = (unsigned char) (ubrr >> 8); /// Step 3. Load ubrr into UBRR0H and UBRR0L
    UBRR0L = (unsigned char) ubrr;

    UCSR0B = 0b00011000; /// Step 4. Enable Tx Rx and disable interrupt in UCSR0B }

int USART_send(char c, FILE *stream) {

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

    UDR0 = c; /// Step 2. Write char to UDR0 for transmission

}

int USART_receive(FILE *stream) {

    while (!(UCSR0A & (1 << RXC0)));

    return UDR0; }

void init_ADC() {

    ADMUX = 0b01000000;
```

```
	DIDR0 = 0b00000001;
	ADCSRA = 0b10000010;}
void motor1(int res) {
	DDRB = 0b00001111;
	PORTB = 0x00;
	int flag = 0;
	int count = 0;
	while (1) {
		if (res > 100) {
			PORTB = 0b00001000;
			_delay_ms(10);
			flag++;
			PORTB = 0b00000100;
			_delay_ms(10);
			flag++;
			PORTB = 0b00000010;
			_delay_ms(10);
			flag++;
			PORTB = 0b00000001;
			_delay_ms(10);
			flag++;
			if(res == flag){
				PORTB = 0x00;
				_delay_ms(1000);
				break;
			}
		}
```

```
        }
}
void motor2(int res) {
        DDRB = 0b00001111;
        PORTB = 0x00;
        int flag = 0;
        int count = 0;
        while (1) {
                if (res > 100) {
                        PORTB = 0b00001000;
                        _delay_ms(10);
                        flag++;
                        PORTB = 0b00000100;
                        _delay_ms(10);
                        flag++;
                        PORTB = 0b00000010;
                        _delay_ms(10);
                        flag++;
                        PORTB = 0b00000001;
                        _delay_ms(10);
                        flag++;
                        if(res == flag){
                                PORTB = 0x00;
                                _delay_ms(1000);
                                break;
                        }
                }
```

```
        }
}
void motorAngleMove(){
        //here 2nd motor angle move code will be written
}


int main() {
        init_ADC();
        USART_init(MYUBRR);
        stdout = fdevopen(USART_send, NULL);
        stdin = fdevopen(NULL, USART_receive);
        while (1) {
                ADCSRA |= (1 << ADSC);
                while (bit_is_set(ADCSRA, ADSC)) {;}
                result = ADC;
                printf("Result %d\n", result);
                _delay_ms(100);
                if(result > 100 && result < 500){
                        motor(result);
                        _delay_ms(5000);
                }
        }
}
```

2.Conveyor Belt

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

```
#define F_CPU 16000000UL
int main() {
   DDRB = 0b00001111;
   PORTB = 0x00;
   int i = 1;
   while (1) {
      if(i == 256)
         i = 1;
      PORTB = 0b00001000;
      _delay_ms(10);
      PORTB = 0b00000100;
      _delay_ms(10);
      PORTB = 0b00000010;
      _delay_ms(10);
      PORTB = 0b00000001;
      _delay_ms(10);
      i++;
      if(i == 128)
      _delay_ms(3000);
   }
}
```

3.A Weak Signal

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#define F_CPU 16000000UL
```

```
#define FOSC 16000000 /**< Clock speed for UBRR calculation. refer page 179 of 328p datasheet. */
#define BAUD 9600 /**< Baud Rate in bps. refer page 179 of 328p datasheet. */
#define MYUBRR FOSC/16/BAUD-1
int result;
void USART_init(unsigned int ubrr){
	UCSR0C = (0<<USBS0)|(3<<UCSZ00);
	UCSR0A = 0b00000000;/// Step 2. Set UCSR0A in Normal speed, disable multi-proc
	UBRR0H = (unsigned char)(ubrr>>8);/// Step 3. Load ubrr into UBRR0H and UBRR0L
	UBRR0L = (unsigned char)ubrr;
	UCSR0B = 0b00011000;/// Step 4. Enable Tx Rx and disable interrupt in UCSR0B
}
int USART_send(char c, FILE *stream){
	while ( !( UCSR0A & (1<<UDRE0)) )/// Step 1.  Wait until UDRE0 flag is high. Busy Waitinig
	{;}
	UDR0 = c; /// Step 2. Write char to UDR0 for transmission
}
int USART_receive(FILE *stream ){
	while ( !(UCSR0A & (1<<RXC0)) )/// Step 1. Wait for Receive Complete Flag is high. Busy waiting
		;
	return UDR0;/// Step 2. Get and return received data from buffer
}
void init_ADC(){
	ADMUX = 0b01000000;
	ADCSRA = 0b10000111;
}
```

```
int main(){
	init_ADC();
	USART_init(MYUBRR);
	stdout = fdevopen(USART_send, NULL);
		stdin = fdevopen(NULL, USART_receive);
	while(1){
		ADCSRA |= (1<<ADSC);
		while (bit_is_set(ADCSRA,ADSC)){ ; }
		result = ADC;
		printf("{\"adc0\":%d}\n",result);
		_delay_ms(100);
	}
}
```

4.Node-red Interface For Joystick Shield

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#define F_CPU 16000000UL

#define FOSC 16000000
#define BAUD 9600 /**< Baud Rate in bps. refer page 179 of 328p datasheet. */
#define MYUBRR FOSC/16/BAUD-1
int result;
void USART_init(unsigned int ubrr) {
	UCSR0C = (0 << USBS0) | (3 << UCSZ00);
```

```
	UCSR0A = 0b00000000;
	UBRR0H = (unsigned char) (ubrr >> 8);
	UBRR0L = (unsigned char) ubrr;
	UCSR0B = 0b00011000; /// Step 4. Enable Tx Rx and disable interrupt in UCSR0B
}
int USART_send(char c, FILE *stream) {
	while (!(UCSR0A & (1 << UDRE0))) {    ;        }
	UDR0 = c; /// Step 2. Write char to UDR0 for transmission
}
int USART_receive(FILE *stream) {

	while (!(UCSR0A & (1 << RXC0)))         ;
	return UDR0; /// Step 2. Get and return received data from buffer
}
void init_ADC() {
	ADMUX = 0b01000000;
	ADCSRA = 0b10000111;
}
uint16_t read_ADC(uint8_t ch) {
	ch &= 0b00000111;  // AND operation with 7
	ADMUX = (ADMUX & 0xF8) | ch; // clears the bottom 3 bits before ORing
	ADCSRA |= (1 << ADSC);
	while (ADCSRA & (1 << ADSC));
	return (ADC);
}
int main() {
	init_ADC();
```

```
USART_init(MYUBRR);
stdout = fdevopen(USART_send, NULL);
stdin = fdevopen(NULL, USART_receive);
while (1) {
	//ADC0
	result = read_ADC(0b01000000);
	printf("{\"adc0%d\":%d}\n", bit_is_set(ADMUX, 0), result);
	_delay_ms(100);
	//ADC1
	result = read_ADC(0b01000001);
	printf("{\"adc0%d\":%d}\n", bit_is_set(ADMUX, 0), result);
	_delay_ms(100);
	//ADC2
	result = read_ADC(0b01000010);
	printf("{\"adc0%d\":%d}\n", bit_is_set(ADMUX, 1), result);
	_delay_ms(100);
	//ADC3
	result = read_ADC(0b01000011);
	printf("{\"adc03\":%d}\n", result);
	_delay_ms(100);
	//ADC4
	result = read_ADC(0b01000100);
	printf("{\"adc04\":%d}\n",result);
	_delay_ms(100);
	//ADC5
	result = read_ADC(0b01000101);
	printf("{\"adc05\":%d}\n", result);
```

```
            _delay_ms(100);
        }
}
```

5.Bose Quite Control

```
#include <avr/io.h>
#include <avr/delay.h>
#define F_CPU 16000000UL
int count = 0;
int volMin = 0;
int volMax = 100;
int volume = 0;
void PINB0_play_pause(uint8_t pin) {
        if (bit_is_set(PINB, pin)) {
            PORTD ^= _BV(PORTD0);
            _delay_ms(20);
        }
}
void PINB1_skip_forward(uint8_t pin) {
        if (bit_is_set(PINB, pin)) {
            _delay_ms(20);
            count++;
            _delay_ms(20);
        }
        if (bit_is_set(PINB, pin)) {
            _delay_ms(20);
            count++;
            _delay_ms(50);
```

```
	}
	if (count == 2) {
		PORTD |= _BV(PORTD1);
		_delay_ms(100);
		PORTD &= ~_BV(PORTD1);
		count = 0;
	}
}
void PINB2_skip_backward(uint8_t pin) {
	if (bit_is_set(PINB, pin)) {
		_delay_ms(20);
		count++;
		_delay_ms(20);
	}
	if (bit_is_set(PINB, pin)) {
		_delay_ms(20);
		count++;
		_delay_ms(50);
	}
	if (bit_is_set(PINB, pin)) {
		_delay_ms(20);
		count++;
		_delay_ms(50);
	}
	if (count == 3) {
		PORTD |= _BV(PORTD2);
		_delay_ms(100);
```

```
                PORTD &= ~_BV(PORTD2);
                count = 0;
        }
}
void PINB3_fast_forward(uint8_t pin) {
        if (bit_is_set(PINB, pin)) {
                _delay_ms(20);
                count++;
                _delay_ms(20);
        }
        if (bit_is_set(PINB, pin)) {
                _delay_ms(20);
                count++;
                _delay_ms(50);
        }
        if (count >= 2 && bit_is_set(PINB, pin)) {
                PORTD |= _BV(PORTD3);
        } else {
                _delay_ms(100);
                PORTD &= ~_BV(PORTD3);
                count = 0;
        }
}
void PINB4_rewind(uint8_t pin) {
        if (bit_is_set(PINB, pin)) {
                _delay_ms(20);
                count++;
```

```
            _delay_ms(20);
        }
        if (bit_is_set(PINB, pin)) {
            _delay_ms(20);
            count++;
            _delay_ms(50);
        }
        if (bit_is_set(PINB, pin)) {
            _delay_ms(20);
            count++;
            _delay_ms(20);
        }
        if (count >= 3 && bit_is_set(PINB, pin)) {
            PORTD |= _BV(PORTD4);
        }
        else {          _delay_ms(100);
                        PORTD &= ~_BV(PORTD4);
                        count = 0;
            }
}
void PINB5_volumeUp(uint8_t pin) {
        if (bit_is_set(PINB, pin)) {
            PORTD |= _BV(PORTD5);
            volume = volume_counter(volume + 1);
            _delay_ms(200);
            PORTD &= ~_BV(PORTD5);
        }
```

```
}
void PINB6_volumeDown(uint8_t pin) {
	if (bit_is_set(PINB, pin)) {
		PORTD |= _BV(PORTD6);
		volume = volume_counter(volume - 1);
		_delay_ms(200);
		PORTD &= ~_BV(PORTD6);
	}
}
int volume_counter(int vol) {
	if (vol < volMin)
		return volMin;
	if (vol > volMax)
		return volMax;
	else	return vol;
}
int main(void) {
	DDRB &= 0x00; //Set as input
	DDRD = 0xFF; // set as output
	PORTD = 0x00;
	while (1) {
		//Press the Multi-function button
		if (bit_is_set(PINB, PINB0)) {
			PINB0_play_pause(PINB0);
		}
		//Press twice quickly.
		if (bit_is_set(PINB, PINB1)) {
```

```
                PINB1_skip_forward(PINB1);
            }
            //Press three times quickly.
            if (bit_is_set(PINB, PINB2)) {
                PINB2_skip_backward(PINB2);
            }
            //Press twice quickly and hold the second press.
            if (bit_is_set(PINB, PINB3)) {
                PINB3_fast_forward(PINB3);
            }
            //Press three times quickly and hold the third press.
            if (bit_is_set(PINB, PINB4)) {
                PINB4_rewind(PINB4);
            }
            //Press onece
            if (bit_is_set(PINB, PINB5)) {
                PINB5_volumeUp(PINB5);
            }
            //Press onece
            if (bit_is_set(PINB, PINB6)) {
                PINB6_volumeDown(PINB6);
            }
        }
}
```

6.CommandLine Interpreter CLI

```
#include <avr/io.h>
#include <stdio.h>
```

```
#include <util/delay.h>
#define F_CPU 16000000UL /**< Clock speed for delay functions. */

#define FOSC 16000000 /**< Clock speed for UBRR calculation. refer page 179 of 328p datasheet. */
#define BAUD 9600 /**< Baud Rate in bps. refer page 179 of 328p datasheet. */
#define MYUBRR FOSC/16/BAUD-1 /**< UBRR = (F_CPU/(16*Baud))-1 for asynch USART page 179 328p datasheet. Baud rate 9600bps, assuming 16MHz clock UBRR0 becomes 0x0067*/
int result;
void USART_init(unsigned int ubrr) {
	UCSR0C = (0 << USBS0) | (3 << UCSZ00);
	UCSR0A = 0b00000000; /// Step 2. Set UCSR0A in Normal speed, disable multi-proc
	UBRR0H = (unsigned char) (ubrr >> 8); /// Step 3. Load ubrr into UBRR0H and UBRR0L
	UBRR0L = (unsigned char) ubrr;
	UCSR0B = 0b00011000; /// Step 4. Enable Tx Rx and disable interrupt in UCSR0B
}
int USART_send(char c, FILE *stream) {
	while (!(UCSR0A & (1 << UDRE0))) {    ; }

	UDR0 = c; /// Step 2. Write char to UDR0 for transmission
}
int USART_receive(FILE *stream) {
	while (!(UCSR0A & (1 << RXC0)))        ;
	return UDR0; /// Step 2. Get and return received data from buffer
}
void init_ADC() {
	ADMUX = 0b01000000;
	ADCSRA = 0b10000111;
```

```
}
uint16_t analogRead(uint8_t ch) {
	ch &= 0b00000111;  // AND operation with 7
	ADMUX = (ADMUX & 0xF8) | ch; // clears the bottom 3 bits before ORing
	ADCSRA |= (1 << ADSC);
	while (ADCSRA & (1 << ADSC))   ;
	return (ADC);
}
void pinMode(uint8_t pin, uint8_t mode) {
	printf("\npin = %d\tmode = %d", pin, mode);
	if (pin >= 2 && pin <= 7 && mode == 1) {
		DDRD |= _BV(pin);
		printf("\tPIN %d is set as OUTPUT.\n", pin);
	} else if (pin >= 2 && pin <= 7 && mode == 0) {
		DDRD &= ~_BV(pin);
		printf("\tPIN %d is set as INPUT.\n", pin);
	} else if (pin >= 8 && pin <= 13 && mode == 1) {
		DDRB |= _BV(pin - 8);
		printf("\tPIN %d is set as OUTPUT.\n", pin);
	} else if (pin >= 8 && pin <= 13 && mode == 0) {
		DDRB &= ~_BV(pin - 8);
		printf("\tPIN %d is set as INPUT.\n", pin);
	}
}
void digitalRead(uint8_t pin) {
	if (pin >= 2 && pin <= 7) {
		if (bit_is_set(DDRD, pin))
```

```
                printf("\tPIN %d is set as OUTPUT.\n", pin);
        else    printf("\tPIN %d is set as INPUT.\n", pin);
    }
    if (pin >= 8 && pin <= 13) {
        if (bit_is_set(DDRB, pin - 8))
                printf("\tPIN %d is set as OUTPUT.\n", pin);
        else    printf("\tPIN %d is set as INPUT.\n", pin);
    }
}
void digitalWrite(uint8_t pin, uint8_t mode) {

    printf("\npin = %d\tmode = %d", pin, mode);
    if (pin >= 2 && pin <= 7 && mode == 1) {
        DDRD |= _BV(pin);
        printf("\tPIN %d is set as OUTPUT.\n", pin);
    } else if (pin >= 2 && pin <= 7 && mode == 0) {
        DDRD &= ~_BV(pin);
        printf("\tPIN %d is set as INPUT.\n", pin);
    } else if (pin >= 8 && pin <= 13 && mode == 1) {
        DDRB |= _BV(pin - 8);
        printf("\tPIN %d is set as OUTPUT.\n", pin);
    } else if (pin >= 8 && pin <= 13 && mode == 0) {
        DDRB &= ~_BV(pin - 8);
        printf("\tPIN %d is set as INPUT.\n", pin);
    }
}
int main() {
```

```
	init_ADC();
	USART_init(MYUBRR);
	stdout = fdevopen(USART_send, NULL);
	stdin = fdevopen(NULL, USART_receive);
	DDRD = 0xFF;
	DDRB = 0xFF;
	DDRC = 0xFF;
	PORTD = 0x00;
	PORTB = 0x00;
	PORTC = 0x00;
	unsigned char select;
	unsigned char a;
	unsigned char b;
	unsigned char m;
	uint8_t pin;
	uint8_t mode;
	while (1) {
		printf("\n\n**pinMode(),digitalWrite() and digitalRead() for pin 2-13 and
analogueRead() 0-5.\n");
		printf("Mode 0 as INPUT and 1 as OUTPUT.\n\n");
		printf("1. PinMode.\n");
		printf("2. DigitalRead.\n");
		printf("3. DigitalWrite.\n");
		printf("4. AnalogRead.\n");
		printf("\n Please select a opeation:\t");
		scanf("%c", &select);
		printf("\n\n");
```

```
switch (select) {
case '1':
        printf("PinMode:\tPIN: ");
        scanf("%c %c", &a, &b);  // input 2 value like 02 or 12 as pin
        printf("\tMODE: ");
        scanf("%c", &m);
        // ASCII value of 0 is 48
        //Converting char to uint8_t data type [value - 48]
        if (a - 48 == 0) {
                pin = b - 48;
        } else {
                pin = 10 + (b - 48);
        }
        mode = m - 48;
        pinMode(pin, mode);                break;
case '2':
        printf("DigitalRead:\tPIN: ");
        scanf("%c %c", &a, &b);
        // ASCII value of 0 is 48
        //Converting char to uint8_t data type [value - 48]
        if (a - 48 == 0) {
                pin = b - 48;
        } else {
                pin = 10 + (b - 48);
        }
        digitalRead(pin);              break;
case '3':
```

```
        printf("DigitalWrite:\tPIN: ");
        scanf("%c %c", &a, &b);
        printf("\tMODE: ");
        scanf("%c", &m);
        // ASCII value of 0 is 48
        //Converting char to uint8_t data type [value - 48]
        if (a - 48 == 0) {
                pin = b - 48;
        } else {
                pin = 10 + (b - 48);
        }
        mode = m - 48;
        digitalWrite(pin, mode);            break;
case '4':
        printf("AnalogRead:\tPIN: ");
        scanf("%c", &a); //input only one value
        pin = a - 48;
        printf("pin = %d\t",pin);
        if (pin == 0) {
                result = analogRead(0b01000000);
                printf("{\"adc0\":%d}\n",result);
                _delay_ms(100);
        }
        if (pin == 1) {
                result = analogRead(0b01000001);
                printf("{\"adc1\":%d}\n",result);
                _delay_ms(100);
```

```
                }
                if (pin == 2) {
                        result = analogRead(0b01000010);
                        printf("{\"adc2\":%d}\n",result);
                        _delay_ms(100);
                }
                if (pin == 3) {
                        result = analogRead(0b01000011);
                        printf("{\"adc3\":%d}\n", result);
                        _delay_ms(100);
                }
                if (pin == 4) {
                        result = analogRead(0b01000100);
                        printf("{\"adc4\":%d}\n",result);
                        _delay_ms(100);
                }
                if (pin == 5) {
                        result = analogRead(0b01000101);
                        printf("{\"adc5\":%d}\n",result);
                        _delay_ms(100);
                }
                break;
        default:
                break;
        }
    }
}
```