

# Design Assignment 2C

---

Student Name: Allis Hierholzer

Student #: 2000160356

Student Email: hierholz@unlv.nevada.edu

Primary Github address: [https://github.com/acexhp/submission\\_da.git](https://github.com/acexhp/submission_da.git)

Directory: Repository/cpe301/DesignAssignment/DA2C

## Task:

The goal of the assignment is use GPIO and delays using Timers and Interrupts:

1. Implement Design Assignment 2A using Timer 0 – normal mode. Count OVF occurrence if needed. Do not use interrupts.
2. Implement Design Assignment 2A using TIMERO\_OVF\_vect interrupt mechanism in normal mode.
3. Implement Design Assignment 2A using TIMERO\_COMPA\_vect interrupt mechanism in CTC mode.

## Submission:

The following are required for successful completion of the design assignment:

- a. AVR C code that has been compiled and working for all four tasks. Verify the period and duty cycle of the waveforms in simulation and emulation.
- b. The C code should be well documented with explanation of every instruction.
- c. A word document that contains the code with comments, complete schematics, that includes the AVR, components connected on the breadboard and LED should be included. Follow the template provided.
- d. A snapshot of the board with connected components and a video of the complete LED bar blink sequence should be recorded and uploaded to Youtube and the link to be provided for each task.
- e. The git directory should have DA2\DA2T1, DA2\DA2T2, ... folders, with one doc file and video link file.

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

- ATMEGA328P XPLAINED MINI
- MULTIFUNCTION SHIELD
- ATMEL STUDIO 7.0
- Oscilloscope

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

### 1A

```
#define F_CPU 16000000UL //clock runs at 16 MHz
#include <avr/io.h>
#include<util/delay.h>

int main()
{
    int overflow = 0; //initialize overflow
    DDRB |= (1 << DDB2); //PB2 as output
    TCCR0A = 0;
    TCNT0 = 0x00; //start timer
    TCCR0B = (1 << CS02) | (1 << CS00); //pre-scaler = 1024

    while (1){
        while ((TIFR0 & 0x01) == 0); //detects overflow
        TCNT0 = 0x00; //resets counter
        TIFR0 = 0x01; //reset overflow flag
        overflow++; //inc overflow
        if (overflow <= 26) //led on
            PORTB = (0 << DDB2);
        else PORTB = (1 << DDB2);
        if (overflow == 44) {
            overflow = 0; //turns off led
        }
    }
}
```

### 1B

```
#define F_CPU 16000000UL //clock runs at 16 MHz
#include <avr/io.h>
#include<util/delay.h>

int overflow = 0; //initialize overflow

int main()
{
    DDRB |= (1<<2); //set PORTB2 as output
    PORTB |= (1<<2); //Turn LED off
    DDRC &= (0<<2); // set PORTC1 for input
    PORTC |= (1<<2); // enable pull-up

    TCCR0A = 0;
    TCCR0B = (1 << CS02) | (1 << CS00); //pre-scaler = 1024

    //when the PINC is pressed, LED pulses
    while (1) {
```

```

        if (!(PINC & (1<<PINC1)))
        {
            overflow = 0;
            TCNT0 = 0;
        }

        while ((TIFR0 & 0x01) == 0); //detects overflow
        TCNT0 = 0x00; //resets counter
        TIFR0 = 0x01; //reset overflow flag
        overflow++; //inc overflow
        if (overflow <= 69) //led turns on
            PORTB = (0 << DDB2);
        else PORTB = (1 << DDB2); //led off
    }

    return 0;
}

```

## 2A

```

#define F_CPU 16000000UL //clock runs at 16 MHz
#include <avr/io.h>
#include<avr/interrupt.h>

int overflow = 0; //initialize overflow

int main(void)
{
    DDRB |= (1 << DDB2); //PB2 as output
    TIMSK0 |= (1 << TOIE0); //enables interrupt
    TCNT0 = 0; //start counter
    sei(); //enables interrupt
    TCCR0B = (1 << CS02) | (1 << CS00); //pre-scaler = 1024
    while (1)
    {
        //timer0 overflow interrupt

        while ((TIFR0 & 0x01) == 0); //detects overflow
        TCNT0 = 0x00; //resets counter
        TIFR0 = 0x01; //reset overflow flag
        overflow++; //inc overflow
        if (overflow <= 13) //led on (13 instead of 26 because...
            PORTB = (0 << DDB2); //... overflow is being doubled)
        else PORTB = (1 << DDB2);
        if (overflow == 22) {
            overflow = 0; //turns off led
        }
    }
}

```

## 2B

```

#define F_CPU 16000000UL //clock runs at 16 MHz
#include <avr/io.h>
#include<avr/interrupt.h>

int overflow = 0; //initialize overflow

```

```

int main(void)
{
    DDRB |= (1 << DDB2);           //PB2 as output
    TIMSK0 |= (1 << TOIE0);        //enables interrupt
    TCNT0 = 0;                     //start counter
    sei();                         //enables interrupt
    TCCR0B = (1 << CS02) | (1 << CS00); //pre-scaler = 1024
    while (1)
    {

    }
}

ISR (TIMER0_OVF_vect)             //timer0 overflow interrupt
{
    while (1) {
        if (!(PINC & (1<<PINC1)))
        {
            overflow = 0;
            TCNT0 = 0;
        }

        while ((TIFR0 & 0x01) == 0); //detects overflow
        TCNT0 = 0x00;                //resets counter
        TIFR0 = 0x01;                //reset overflow flag
        overflow++;                  //inc overflow
        if (overflow <= 69)           //led turns on
            PORTB = (0 << DDB2);
        else PORTB = (1 << DDB2);    //led off
    }
}

```

### 3A

```

#define F_CPU 16000000UL           //clock runs at 16 MHz
#include <avr/io.h>
#include<avr/interrupt.h>

int overflow = 0;                  //initialize overflow

int main(void)
{
    DDRB |= (1 << DDB2);           //PB2 as output
    TCNT0 = 0;                     //start counter
    OCR0A = 255;                   //load compare reg value
    TCCR0A |= (1 << WGM01);        //set to ctc mode
    TIMSK0 |= (1 << OCIE0A);       //set interrupt on compare match
    TCCR0B = (1 << CS02) | (1 << CS00); //pre-scaler = 1024
    sei();                         //enables interrupt
    while (1)
    {

    }
}

ISR (TIMER0_COMP_vect)            //timer0 overflow interrupt
{
    while ((TIFR0 & 0x02) == 0);   //detects overflow
    TCNT0 = 0x00;                  //resets counter
}

```

```

        TIFR0 = 0x02;
        overflow++;
        if (overflow <= 13)
            PORTB = (0 << DDB2);
        else PORTB = (1 << DDB2);
        if (overflow == 22) {
            overflow = 0;
        }
    }
}

```

### 3B

```

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>

int overflow = 0;

int main(void)
{
    DDRB |= (1 << DDB2);
    TCNT0 = 0;
    OCR0A = 255;
    TCCR0A |= (1 << WGM01);
    TIMSK0 |= (1 << OCIE0A);
    TCCR0B = (1 << CS02) | (1 << CS00);
    sei();
    while (1)
    {

    }
}

ISR (TIMER0_COMPA_vect)
{
    while (1) {
        if (!(PINC & (1<<PINC1)))
        {
            overflow = 0;
            TCNT0 = 0;
        }

        while ((TIFR0 & 0x02) == 0);
        TCNT0 = 0x00;
        TIFR0 = 0x02;
        overflow++;
        if (overflow <= 69)
            PORTB = (0 << DDB2);
        else PORTB = (1 << DDB2);
    }
}

```

```

//reset overflow flag
//inc overflow
//led on (13 instead of 26 because...
//... overflow is being doubled)

//turns off led

```

```

//clock runs at 16 MHz

```

```

//initialize overflow

```

```

//PB2 as output
//start counter
//load compare reg value
//set to ctc mode
//set interrupt on compare match
//pre-scaler = 1024
//enables interrupt

```

```

//timer0 overflow interrupt

```

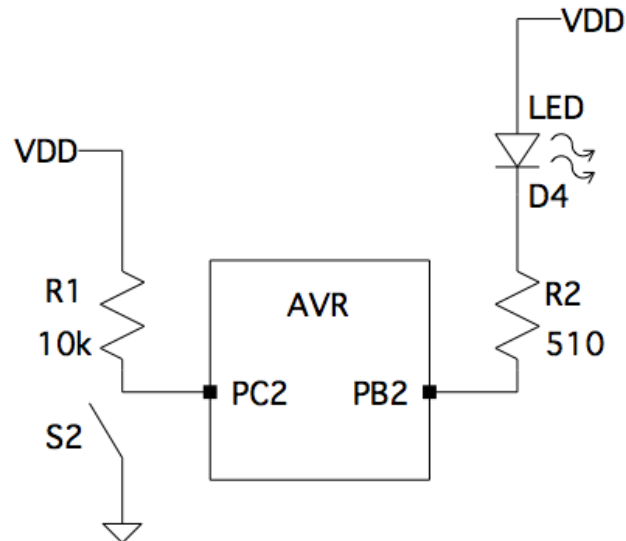
```

//detects overflow
//resets counter
//reset overflow flag
//inc overflow
//led turns on

//led off

```

### 3. SCHEMATICS



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

#### 1A

The screenshot shows the Atmel Studio IDE with the main.c file open. The code defines a 16 MHz clock and includes the AVR I/O and delay headers. The main function initializes an overflow counter, configures PB2 as an output, and starts a timer with a pre-scaler of 1024. The timer's overflow flag (TIFR0) is used to toggle the LED (D4) on and off. The I/O window on the right shows the status of the I/O ports, with PORTB (0x25) set to 0x00 and DDRB (0x24) set to 0x04.

```
1a main.c X
main int main()
/*
 * 1a.c
 */

#define F_CPU 16000000UL //clock runs at 16 MHz
#include <avr/io.h>
#include<util/delay.h>

int main()
{
    int overflow = 0; //initialize overflow
    DDRB |= (1 << DDB2); //PB2 as output
    TCCR0A = 0;
    TCNT0 = 0x00; //start timer
    TCCR0B = (1 << CS02) | (1 << CS00); //pre-scaler = 1024

    while (1){
        while ((TIFR0 & 0x01) == 0); //detects overflow
        TCNT0 = 0x00; //resets counter
        TIFR0 = 0x01; //reset overflow flag
        overflow++; //inc overflow
        if (overflow <= 26) //led on
            PORTB = (0 << DDB2);
        else PORTB = (1 << DDB2);
        if (overflow == 44) {
            overflow = 0; //turns off led
        }
    }
}
```

I/O

Name	Address	Value	Bits
PINB	0x23	0x00	00000000
DDRB	0x24	0x04	00000000
PORTB	0x25	0x00	00000000

The screenshot displays the AVR Studio IDE with the C code for the LED blink program. The code is as follows:

```

/*
 * 1b.c
 */

#define F_CPU 16000000UL           //clock runs at 16 MHz
#include <avr/io.h>
#include<util/delay.h>

int overflow = 0;                //initialize overflow

int main()
{
    DDRB |= (1<<2);               //set PORTB2 as output
    PORTB |= (1<<2);              //Turn LED off
    DDRC &= (0<<2);              // set PORTC1 for input
    PORTC |= (1<<2);              // enable pull-up

    TCCR0A = 0;
    TCCR0B = (1 << CS02) | (1 << CS00); //pre-scaler = 1024

    //when the PINC is pressed, LED pulses
    while (1) {
        if (!(PINC & (1<<PINC1)))
        {
            overflow = 0;
            TCNT0 = 0;

            while ((TIFR0 & 0x01) == 0); //detects overflow
            TCNT0 = 0x00;                //resets counter
            TIFR0 = 0x01;                //reset overflow flag
            overflow++;                  //inc overflow
            if (overflow <= 76)          //led turns on
                PORTB = (0 << DDB2);
            else PORTB = (1 << DDB2);   //led off
        }

        return 0;
    }
}

```

The I/O window on the right shows the hardware components of the AVR microcontroller. The I/O Port (PORTB) is highlighted in blue. The I/O window also displays the address and value of the I/O ports and the status of the bits.

Name	Address	Value	Bits
PINB	0x23	0x00	00000000
DDRB	0x24	0x04	00000100
PORTB	0x25	0x00	00000000

[illegible]

## 2A

The image shows an AVR IDE window with a C program for an AVR microcontroller. The code defines a timer overflow interrupt that toggles an LED. The IDE also displays a list of AVR peripherals on the right.

```
main.c 2a
main int main(void)
/*
 * 2a.c
 */

#define F_CPU 16000000UL //clock runs at 16 MHz
#include <avr/io.h>
#include<avr/interrupt.h>

int overflow = 0; //initialize overflow

int main(void)
{
    DDRB |= (1 << DDB2); //PB2 as output
    TIMSK0 |= (1 << TOIE0); //enables interrupt
    TCNT0 = 0; //start counter
    sei(); //enables interrupt
    TCCR0B = (1 << CS02) | (1 << CS00); //pre-scaler = 1024
    while (1)
    {
    }
}

ISR (TIMER0_OVF_vect) //timer0 overflow interrupt
{
    while ((TIFR0 & 0x01) == 0); //detects overflow
    TCNT0 = 0x00; //resets counter
    TIFR0 = 0x01; //reset overflow flag
    overflow++; //inc overflow
    if (overflow <= 13) //led on (13 instead of 26 because...
    PORTB = (0 << DDB2); //... overflow is being doubled)
    else PORTB = (1 << DDB2);
    if (overflow == 22) {
        overflow = 0; //turns off led
    }
}
```

Peripheral List:

Name	Value
ADC (DAC)	Analog Comparator (AC)
ADC (DAC)	Analog-to-Digital Convert...
CPU	CPU Registers (CPU)
EEPROM	EEPROM (EEPROM)
External Interrupts (EXINT)	External Interrupts (EXINT)
I/O	I/O Port (PORTB)
I/O	I/O Port (PORTC)
I/O	I/O Port (PORTD)
Serial Peripheral Interface (...)	Serial Peripheral Interface (...)
Timer/Counter, 16-bit (TC1)	Timer/Counter, 16-bit (TC1)

Name	Address	Value	Bits
I/O PINB	0x23	0x00	00000000
I/O DDRB	0x24	0x04	00000100
I/O PORTB	0x25	0x00	00000000



**2B**

The image shows a code editor with C code for an AVR microcontroller. The code defines a clock frequency of 16 MHz and includes headers for I/O and interrupts. It initializes an overflow flag and sets up a timer overflow interrupt. The main function configures DDRB for output, enables the timer interrupt, and enters a loop where it checks for timer overflow and toggles PORTB pin 2 (PB2) as an LED.

```

#define F_CPU 16000000UL           //clock runs at 16 MHz
#include <avr/io.h>
#include<avr/interrupt.h>

int overflow = 0;                  //initialize overflow

int main(void)
{
    DDRB |= (1 << DDB2);           //PB2 as output
    TIMSK0 |= (1 << TOIE0);        //enables interrupt
    TCNT0 = 0;                     //start counter
    sei();                         //enables interrupt
    TCCR0B = (1 << CS02) | (1 << CS00); //pre-scaler = 1024
    while (1)
    {
        //timer0 overflow interrupt
        while (1) {
            if (!(PINC & (1<<PINC1)))
            {
                overflow = 0;
                TCNT0 = 0;
            }

            while ((TIFR0 & 0x01) == 0); //detects overflow
            TCNT0 = 0x00;                 //resets counter
            TIFR0 = 0x01;                 //reset overflow flag
            overflow++;                   //inc overflow
            if (overflow <= 69)           //led turns on
                PORTB = (0 << DDB2);
            else PORTB = (1 << DDB2);    //led off
        }
    }
}

```

On the right, the 'I/O' peripheral view shows the following components:

- Analog Comparator (AC)
- Analog-to-Digital Convert...
- CPU Registers (CPU)
- EEPROM (EEPROM)
- External Interrupts (EXINT)
- I/O Port (PORTB)** (Selected)
- I/O Port (PORTC)
- I/O Port (PORTD)
- Serial Peripheral Interface (...)
- Timer/Counter, 16-bit (TC1)

The detailed view of the selected I/O Port (PORTB) shows the following registers and their bit states:

Name	Address	Value	Bits
PINB	0x23	0x98	Bit 7: 1, Bit 6: 1, Bit 5: 1, Bit 4: 1, Bit 3: 0, Bit 2: 0, Bit 1: 0, Bit 0: 0
DDRB	0x24	0x04	Bit 7: 0, Bit 6: 0, Bit 5: 0, Bit 4: 0, Bit 3: 0, Bit 2: 1, Bit 1: 0, Bit 0: 0
PORTB	0x25	0x00	Bit 7: 0, Bit 6: 0, Bit 5: 0, Bit 4: 0, Bit 3: 0, Bit 2: 0, Bit 1: 0, Bit 0: 0

### 3A

main.c 3a

main int main(void)

```
/*
 * 3a.c
 */

#define F_CPU 16000000UL //clock runs at 16 MHz
#include <avr/io.h>
#include<avr/interrupt.h>

int overflow = 0; //initialize overflow

int main(void)
{
    DDRB |= (1 << DDB2); //PB2 as output
    TCNT0 = 0; //start counter
    OCR0A = 255; //load compare reg value
    TCCR0A |= (1 << WGM01); //set to ctc mode
    TIMSK0 |= (1 << OCIE0A); //set interrupt on compare match
    TCCR0B = (1 << CS02) | (1 << CS00); //pre-scaler = 1024
    sei(); //enables interrupt
    while (1)
    {
    }
}

ISR (TIMER0_COMPA_vect) //timer0 overflow interrupt
{
    while ((TIFR0 & 0x02) == 0); //detects overflow
    TCNT0 = 0x00; //resets counter
    TIFR0 = 0x02; //reset overflow flag
    overflow++; //inc overflow
    if (overflow <= 13) //led on (13 instead of 26 because...
    PORTB = (0 << DDB2); //... overflow is being doubled)
    else PORTB = (1 << DDB2);
    if (overflow == 22) {
        overflow = 0; //turns off led
    }
}
```

I/O

Filter:

	Name	Value
+	Analog Comparator (AC)	
+	Analog-to-Digital Convert...	
+	CPU Registers (CPU)	
+	EEPROM (EEPROM)	
+	External Interrupts (EXINT)	
I/O	I/O Port (PORTB)	
I/O	I/O Port (PORTC)	
I/O	I/O Port (PORTD)	
+	Serial Peripheral Interface (...)	
+	Timer/Counter, 16-bit (TC1)	

	Name	Address	Value	Bits
I/O	PINB	0x23	0x00	□□□□□□□□
I/O	DDRB	0x24	0x04	□□□□□■□□
I/O	PORTB	0x25	0x00	□□□□□□□□

The screenshot displays the AVR Studio IDE with a C program for an AVR microcontroller. The code is divided into three main sections: header files, initialization, and the main function.

```

main
{
    /*
     * 3b.c
     */

    #define F_CPU 16000000UL //clock runs at 16 MHz
    #include <avr/io.h>
    #include <avr/interrupt.h>

    int overflow = 0; //initialize overflow

    int main(void)
    {
        DDRB |= (1 << DDB2); //PB2 as output
        TCNT0 = 0; //start counter
        OCR0A = 255; //load compare reg value
        TCCR0A |= (1 << WGM01); //set to ctc mode
        TIMSK0 |= (1 << OCIE0A); //set interrupt on compar
        TCCR0B = (1 << CS02) | (1 << CS00); //pre-scaler = 1024
        sei(); //enables interrupt
        while (1)
        {
        }
    }

    ISR (TIMER0_COMPA_vect) //timer0 overflow interrupt
    {
        while (1) {
            if (!(PINC & (1<<PINC1)))
            {
                overflow = 0;
                TCNT0 = 0;
            }

            while ((TIFR0 & 0x02) == 0); //detects overflow
            TCNT0 = 0x00; //resets counter
            TIFR0 = 0x02; //reset overflow flag
            overflow++; //inc overflow
            if (overflow <= 69) //led turns on
                PORTB = (0 << DDB2);
            else PORTB = (1 << DDB2); //led off
        }
    }
}

```

On the right side, the I/O peripheral window is open, showing a list of peripherals and a table of I/O registers.

**I/O Peripherals List:**

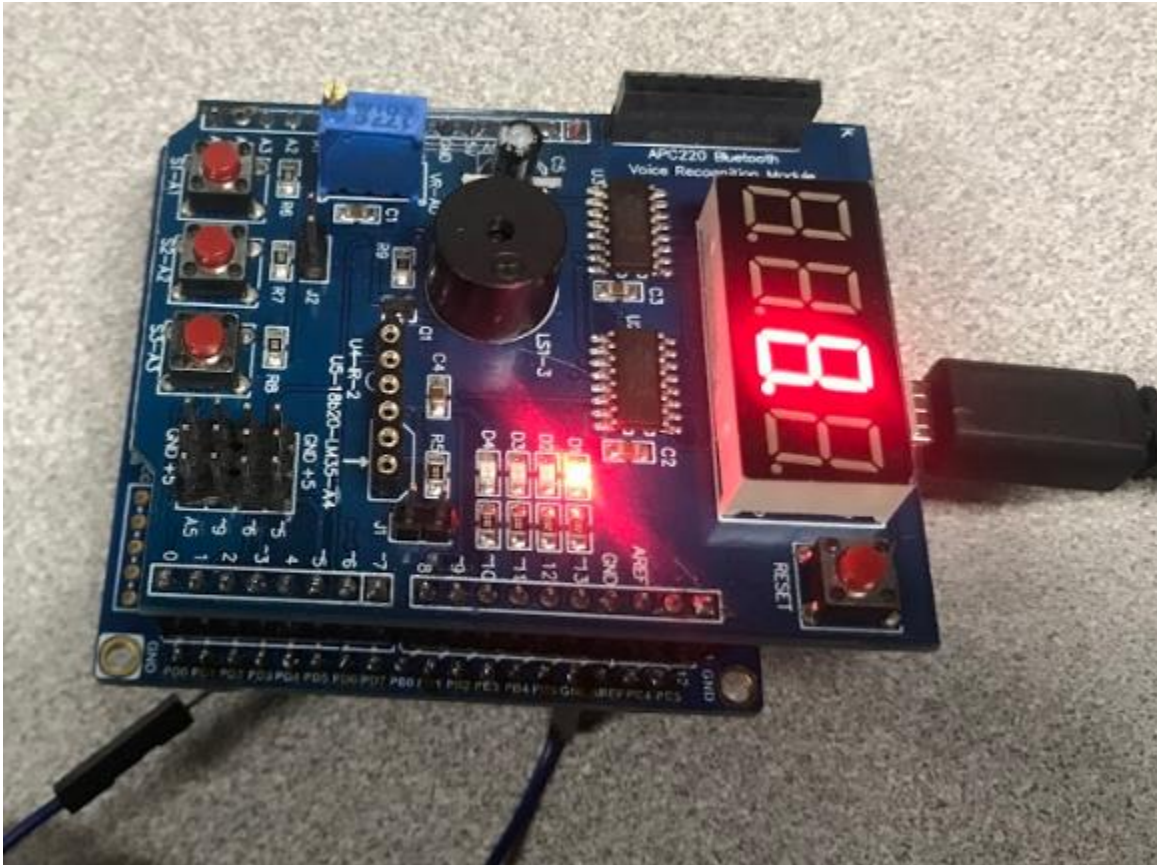
- ADC (Analog Comparator (AC))
- ADC (Analog-to-Digital Convert...)
- CPU Registers (CPU)
- EEPROM (EEPROM)
- External Interrupts (EXINT)
- I/O Port (PORTB) - **Selected**
- I/O Port (PORTC)
- I/O Port (PORTD)
- Serial Peripheral Interface (...)
- Timer/Counter, 16-bit (TC1)

**I/O Register Table:**

Name	Address	Value	Bits
PINB	0x23	0x00	00000000
DDRB	0x24	0x04	00000001
PORTB	0x25	0x00	00000000

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

The board set up is the same for all exercises



6. VIDEO LINKS OF EACH DEMO

1A

<https://youtu.be/7cCjv1m9Di4>

1B

<https://youtu.be/z2lXkrFeqw0>

2A

<https://youtu.be/m-iaM6LmcC8>

2B

<https://youtu.be/j0gw1afVVrg>

3A

<https://youtu.be/hrxYflyjL0o>

3B

<https://youtu.be/MIzcfgecc1g>

**7. GITHUB LINK OF THIS DA**

[https://github.com/acexhp/submission\\_da.git](https://github.com/acexhp/submission_da.git)

**Student Academic Misconduct Policy**

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

*"This assignment submission is my own, original work".*  
Allis Hierholzer