Ivan Soto
Github root directory: https://github.com/sotoi2/Class3.0.4

Date Submitted:

Task 01:

Youtube Link: https://www.youtube.com/watch?v=4_HSrc0m4tg

Modified Schematic (if applicable):

Modified Code:

```
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/fpu.h"
#include "driverlib/sysctl.h"
#include "driverlib/rom.h"

#define TARGET_IS_BLIZZARD_RB1

#ifndef M_PI
#define M_PI                    3.14159265358979323846
#endif

#define SERIES_LENGTH 100

float gSeriesData[SERIES_LENGTH];

int32_t i32DataCount = 0;

int main(void)
{
    float fRadians;

    ROM_FPULazyStackingEnable();
    ROM_FPUEnable();

    ROM_SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
SYSCTL_OSC_MAIN);

    fRadians = ((2 * M_PI) / SERIES_LENGTH);

    while(i32DataCount < SERIES_LENGTH)
    {
        gSeriesData[i32DataCount] = sinf(fRadians * i32DataCount);
```

```
        i32DataCount++;
    }

    while(1)
    {
    }
}
```

Task 02:

Youtube Link: https://www.youtube.com/watch?v=KSHmncGi6hg

Modified Schematic (if applicable):

Modified Code:

```
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/fpu.h"
#include "driverlib/sysctl.h"
#include "driverlib/rom.h"

#define TARGET_IS_BLIZZARD_RB1
// used for assigning radians value
#ifndef M_PI
#define M_PI                    3.14159265358979323846
#endif

#define SERIES_LENGTH 100

float gSeriesData[SERIES_LENGTH];

int32_t i32DataCount = 0;

int main(void)
{
    float fRadians;
    //enable fpu calculations
    ROM_FPULazyStackingEnable();
    ROM_FPUEnable();
    // set clock
    ROM_SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
SYSCTL_OSC_MAIN);
    // set value for radians

    fRadians = ((2 * M_PI) / SERIES_LENGTH);
    // count for 100 times
    while(i32DataCount < SERIES_LENGTH)
```

```
    {
        // equation to graph / create
        gSeriesData[i32DataCount] = sinf( fRadians * i32DataCount + 50*i32DataCount)
+ 0.5*cosf(fRadians * i32DataCount + 200*i32DataCount);

        i32DataCount++;
    }

    while(1)
    {
    }
}
```