Ali Asadi
Github root directory: https://github.com/aliasadivet/Lab_Submition-.git

Lab06

Date Submitted: **10/10/2019**

Task 00: Execute provided code

Youtube Link: No submition required

Task 01:

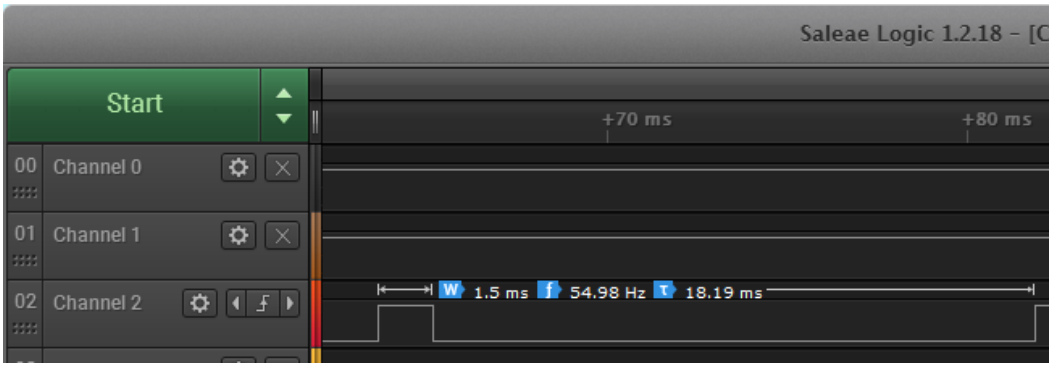
I have done this task in two ways:

1- Sweeping the servo angle between 0 and 180 degrees using buttons same as original code.
2- Sweeping the servo angle between 0 and 180 degrees continuesly.

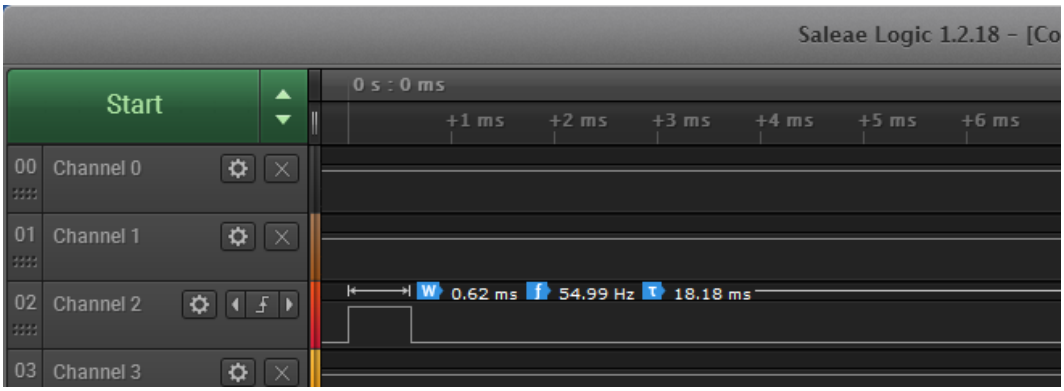
Youtube Link: https://youtu.be/61yx-yTJvI8
https://youtu.be/ZlRHjIg-osg

Modified Schematic (if applicable): N/A

PWM Measurements:

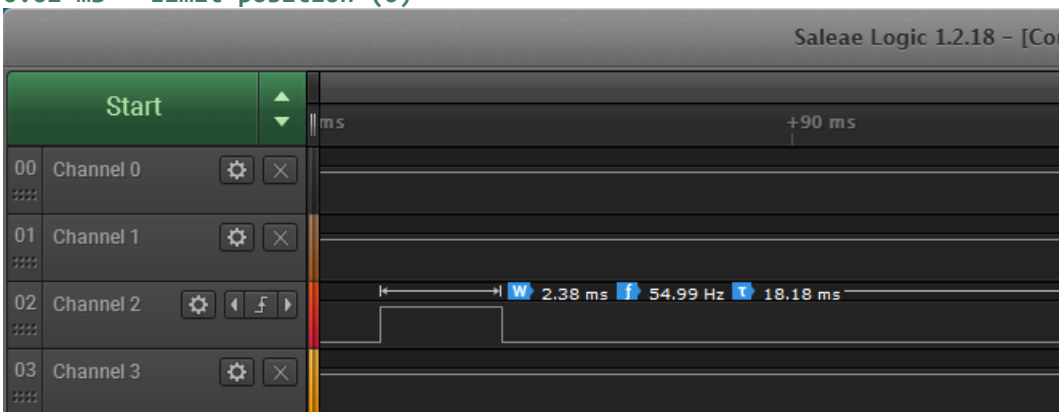


1.5 mS – Center Position (90)



Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

0.62 mS – Limit position (0)



2.38 mS – Limit position (180)

Modified Code: (using buttons)

```
// Insert code here

#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/debug.h"
#include "driverlib/pwm.h"
#include "driverlib/pin_map.h"
#include "inc/hw_gpio.h"
#include "driverlib/rom.h"

#define PWM_FREQUENCY 55

int main(void)
{
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 83;


ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
    ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0);

    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = 0;
    ROM_GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_DIR_MODE_IN);
    ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);

    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
    PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load);

    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_0);

    while(1)
    {
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0x00)
        {
            ui8Adjust--;
            if (ui8Adjust < 34)
            {
            ui8Adjust = 34;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
        }

        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_0)==0x00)
        {
            ui8Adjust++;
            if (ui8Adjust > 131)
            {
            ui8Adjust = 131;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
        }

        ROM_SysCtlDelay(100000);
    }
}
```

Modified Code: (continuesly sweeping)

```
// Insert code here
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/debug.h"
#include "driverlib/pwm.h"
```

```
#include "driverlib/pin_map.h"
#include "inc/hw_gpio.h"
#include "driverlib/rom.h"

#define PWM_FREQUENCY 55

int main(void)
{
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 83;


ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
    ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0);

    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = 0;
    ROM_GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_DIR_MODE_IN);
    ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);

    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
    PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load);

    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_0);

    while(1)
    {
        ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, 34 * ui32Load / 1000);
        ROM_SysCtlDelay(10000000);
        ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, 131 * ui32Load / 1000);
        ROM_SysCtlDelay(10000000);
    }
}
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

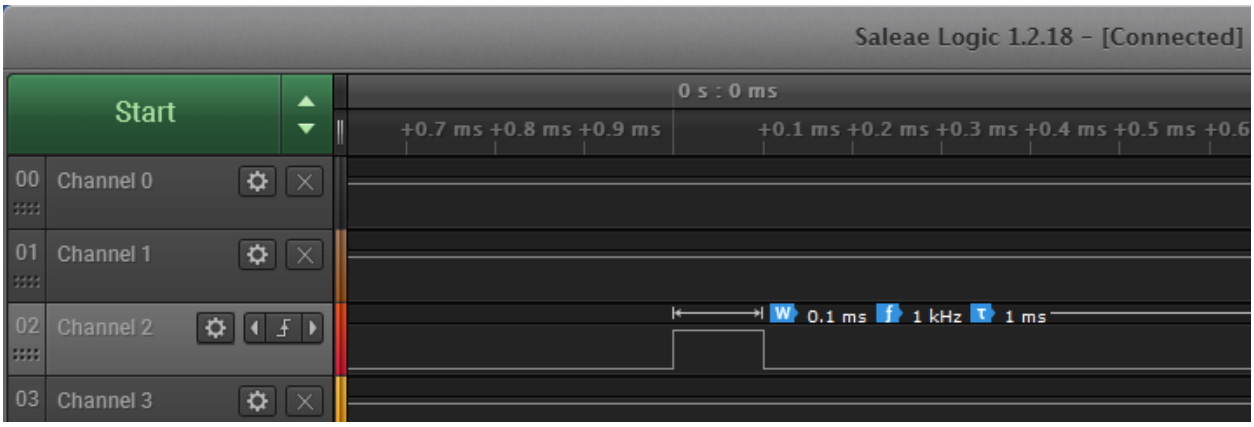
Ali Asadi
Github root directory: https://github.com/aliasadivet/Lab_Submition-.git

Task 02:

Youtube Link: https://youtu.be/dMpPdcU0Ass

Modified Schematic (if applicable): N/A

PWM Measurement:



Modified Code:

```
// Insert code here

#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/debug.h"
#include "driverlib/pwm.h"
#include "driverlib/pin_map.h"
#include "inc/hw_gpio.h"
#include "driverlib/rom.h"
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
#define PWM_FREQUENCY 1000

int main(void)
{
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;


    ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16M
    HZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1);
    ROM_GPIOPinConfigure(GPIO_PF1_M1PWM5);

    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
    PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, 1000 * ui32Load / 1000);
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_2);

    while(1)
    {
        int i;
        for( i = 900; i >= 100; i -= 25)
        {
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, i * ui32Load / 1000);
            ROM_SysCtlDelay(1000000);
        }

        for( i = 100; i < 900; i += 25)
        {
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, i * ui32Load / 1000);
            ROM_SysCtlDelay(1000000);
        }
    }
}
```

For Task 03 and 04 since we didn't have DC motor, I will submit in another report.

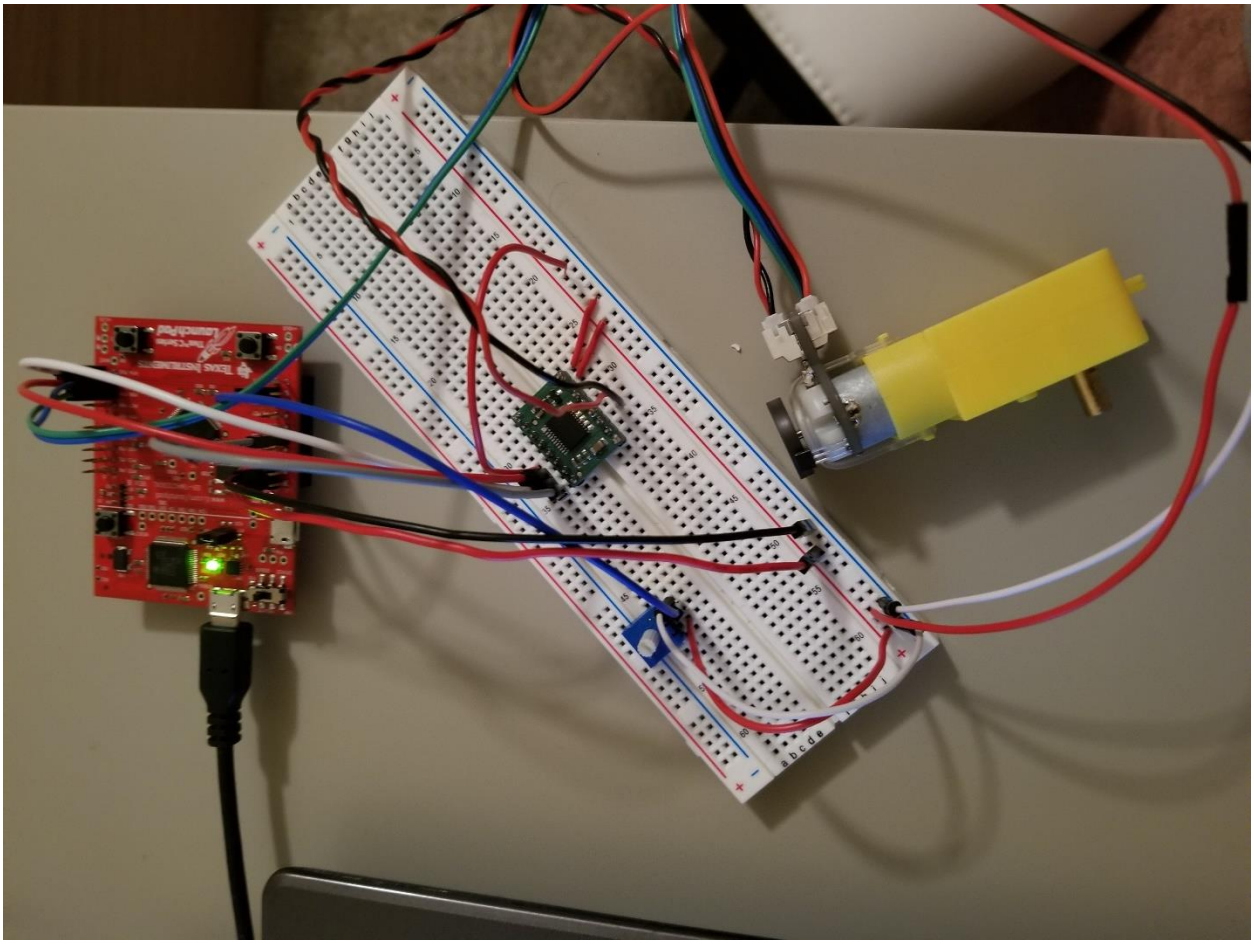
Task 03:

Youtube Link: https://youtu.be/A8wuwMj-t7A

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Ali Asadi
Github root directory: https://github.com/aliasadivet/Lab_Submition-.git

Connections:



Modified Code:

```
// Insert code here

#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
#include "driverlib/gpio.h"
#include "driverlib/debug.h"
#include "driverlib/pwm.h"
#include "driverlib/pin_map.h"
#include "inc/hw_gpio.h"
#include "driverlib/rom.h"

#define PWM_FREQUENCY 1000

int main(void)
{
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
    uint32_t ui32ADC0Value[4];
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint32_t ui32MotorPWMDutyCycle;


ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 64);
    ROM_GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_3);
    ROM_ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_CH0);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_CH0);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_CH0);
    ROM_ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_CH0|ADC_CTL_IE|ADC_CTL_END);
    ROM_ADCSequenceEnable(ADC0_BASE, 1);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_2|GPIO_PIN_3);
    ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2|GPIO_PIN_3, 4);
    ROM_GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_4);
    ROM_GPIOPinConfigure(GPIO_PE4_M1PWM2);

    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
    PWMGenConfigure(PWM1_BASE, PWM_GEN_1, PWM_GEN_MODE_DOWN);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_1, ui32Load);
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_2, 0 * ui32Load / 1000);
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_2_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_1);

    while(1)
    {
        ROM_ADCIntClear(ADC0_BASE, 1);
        ROM_ADCProcessorTrigger(ADC0_BASE, 1);

        while(!ROM_ADCIntStatus(ADC0_BASE, 1, false)) { }

        ROM_ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
        ui32MotorPWMDutyCycle = (ui32ADC0Value[0] + ui32ADC0Value[1] +
ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;

        ui32MotorPWMDutyCycle = (uint32_t)(((float)ui32MotorPWMDutyCycle / 4096.0) *
ui32Load);

        ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_2,  ui32MotorPWMDutyCycle +1);
    }
}
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Ali Asadi
Github root directory: https://github.com/aliasadivet/Lab_Submition-.git

---

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.