Assignment Project Report

Assignment No - 2

Embedded Systems and Designs

Aniket Chelawat – 20uec019

Animesh Jain – 20uec020

Instructor: Dr. Deepak Nair

Department of Electronics and Communication Engineering
The LNMIIT Jaipur, Rajasthan

April 24, 2023

Declaration

We, as a group, hereby declare that this report is entirely our own work. We have not used any material from other sources without proper attribution, and any work contributed by individuals outside of the group has been duly acknowledged. We acknowledge and give credit to all sources that have been used in the preparation of this report.

Animesh Jain – 20UEC020

Aniket Chelawat– 20UEC019

Date of Submission:24/04/2023

Table of Contents

Abstract

Microcontrollers have revolutionized the electronics industry, providing a means of controlling the behaviour of electronic devices in a wide range of applications. These devices are widely used in embedded systems, consumer electronics, automotive systems, medical devices, and more. They are highly versatile and can perform a variety of tasks, from simple control operations to more complex signal processing tasks. As technology has advanced, microcontrollers have become more powerful and efficient, enabling developers to create smaller, more reliable, and more complex systems than ever before.

In this project, we have utilized the Tiva C board, which features a robust ARM Cortex-M4 microcontroller. This board provides a versatile and adaptable platform for developing embedded systems and is commonly employed in both academic and industrial contexts. The Tiva C board includes a range of components such as timers, ADCs, DACs, PWMs, and GPIOs that can be used for various tasks, including generating audio tones.

The aim of this project is to implement a simple audio tone generator using a Tiva C board with an internal DAC and speaker. The project involves using GPIOs and timers to generate the desired waveform and output it through the internal DAC and speaker. The implementation process involves programming the microcontroller, interfacing the hardware components, and testing the output waveform using a DSO.

The project has significant implications for the development of embedded systems and can be used as a foundation for more complex audio signal processing applications. The successful implementation of the audio tone generator demonstrates the potential of microcontrollers in creating sophisticated and versatile electronic devices. This project provides a practical example of how microcontrollers can be used to control the behaviour of electronic devices and highlights the importance of understanding the role of hardware components, such as the breadboard, LED, buzzer, jumper wires, and resistors, in the development of electronic systems.

Component Name

TIVA C board is used to perform this experiment. The board contains a wide range of components that enable developers to create sophisticated and complex systems. The key components used in the Tiva C board are listed below:

1. **Microcontroller:** The TIVA C board is equipped with an ARM Cortex-M4 microcontroller, which is a powerful 32-bit processor that can run at up to 80MHz. This microcontroller provides a wide range of features, such as hardware floating-point support, digital signal processing (DSP) instructions, and a memory protection unit (MPU). Additionally, the Cortex-M4 microcontroller also supports a variety of communication protocols, such as USB, Ethernet, and CAN, making it an ideal choice for a wide range of embedded systems.

Figure 1: TIVA-C Board contains 2 TM4C123GH6PM Microcontrollers

2. **Digital-to-analog converter (DAC):** The board also has a 12-bit DAC that can output analog signals with a maximum update rate of 1Mbps. The DAC is used to convert digital signals into analog signals, which can be used to control analog devices, such as motors or speakers.

3. **General-purpose input/output (GPIO):** The TIVA C board has 43 GPIO pins that can be used for digital input and output. These pins can be configured as inputs or outputs, and they can be used to interface with other digital devices, such as sensors or LEDs. GPIO pins can also be used for communication with other devices using protocols such as I2C or SPI.

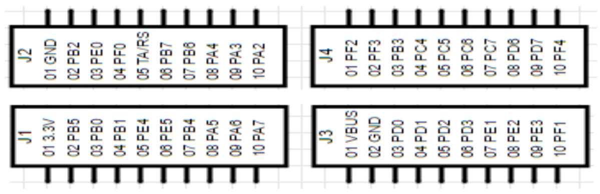


Figure 2: GPIO Pins available on the board.

4. **Timers:** The board features four 16-bit timers and two 32-bit timers that can be used for a variety of timing and control applications. Timers are used to generate precise timing intervals, such as for measuring the duration of an event or for generating a periodic signal. The timers on the TIVA C board are highly configurable, with options for selecting the timer mode, clock source, and interrupt handling.

5. **Pulse-width modulation (PWM) module:** The board also includes a PWM module that can be used to generate signals with varying pulse widths. This can be useful for controlling the brightness of LEDs or the speed of motors.

6. **Clock module:** The Tiva C board includes a clock module that generates clock signals used by the microcontroller and other components. The clock module can be configured to generate different frequencies and can be used to control the timing of the system.

7. **Reset button:** The board includes a reset button that can be used to reset the board and restart the program.

1. Components used in circuit:

2. **Bread Board:** In the context of the audio tone generator project with Tiva C, the breadboard played an essential role in connecting the LED, resistor, and buzzer to the Tiva C board. The breadboard provided a convenient way to prototype and test the circuit without the need for soldering, which allowed for quick modifications and adjustments to the circuit.

3. **Buzzer:** A buzzer is an electronic device that produces sound when an electric signal is applied to it. It consists of a piezoelectric element that vibrates at a specific frequency to generate sound. To generate an audio tone using the buzzer, we connected it to one of the GPIO pins on the Tiva C board and used the board's internal DAC to provide the signal. The buzzer was connected to the breadboard along with a resistor to limit the current flowing through it.

4. **LED:** In our project, we used the LED as an indicator to show the status of the system. We connected the LED to a GPIO pin of the Tiva C board through a current-limiting resistor to ensure that the LED does not get damaged due to excessive current. The LED was turned on and off using software commands to generate a square wave of a particular frequency, which was used to drive the tone generator.

5. **Resistor:** A resistor is an electronic component that limits the flow of electric current in a circuit. It is a passive component, meaning that it does not produce any energy but instead dissipates or converts electrical energy into heat. In our assignment, we used a resistor in conjunction with an LED to limit the current flowing through the LED and prevent it from burning out. The value of the resistor is determined based on the forward voltage of the LED and the desired current flowing through it.

6. **Jumper Wire:** Jumper wires were used to connect the LED, buzzer, and resistor on the breadboard. We also used them to connect the breadboard to the Tiva C board's GPIO pins, which provided power and ground to the circuit. Jumper wires are versatile and easy to use, and they allow for quick and flexible prototyping of circuits.

Understanding Ports

1. **RCGCGPIO:** The RCGCGPIO register provides software with the capability to enable and disable GPIO modules in Run mode. (For More Details Refer Page No: 340 pg. of the datasheet.)

2. **RCGCPWM:** The RCGCPWM register provides software the capability to enable and disable the PWM modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. (For More Details Refer Page No: 354 pg. of the datasheet.)

3. **RCC:** The bits in this register configure the system clock and oscillators. (For More Details Refer Page No: 254 pg. of the datasheet.)

4. **GPIOAFSEL:** The GPIOAFSEL register is the mode control select register. If a bit is clear, the pin is used as a GPIO and is controlled by the GPIO registers. Setting a bit in this register configures the corresponding GPIO line to be controlled by an associated peripheral. (For More Details Refer Page No: 340 pg. of the datasheet.)

5. **GPIOPCTL:** The GPIOPCTL register is used in conjunction with the GPIOAFSEL register and selects the specific peripheral signal for each GPIO pin when using the alternate function mode. (For More Details Refer Page No: 688 pg. of the datasheet.)

6. **PWM3CTL:** These registers configure the PWM signal generation blocks (PWM0CTL controls the PWM generator 0 block, and so on). The Register Update mode, Debug mode, counting mode, and Block Enable mode are all controlled via these registers. (For More Details Refer Page No: 1266 pg. of the datasheet.)

7. **PWM3LOAD:** These registers contain the load value for the PWM counter (PWM0LOAD controls the PWM generator 0 block, and so on). (For More Details Refer Page No: 1278 pg. of the datasheet.)

8. **PWM3CMPA:** These registers contain a value to be compared against the counter (PWM0CMPB controls the PWM generator 0 block, and so on). (For More Details Refer Page No: 1280 pg. of the datasheet.)

9. **PWM3GENA:** These registers control the generation of the pwmA signal based on the load and zero output pulses from the counter, as well as the compare A and compare B pulses from the comparators (PWM0GENA controls the PWM generator 0 block, and so on). (For More Details Refer Page No: 1282 pg. of the datasheet.)

10. **PWMENABLE:** This register provides a master control of which generated pwmA' and pwmB' signals are output to the MnPWMn pins. (For More Details Refer Page No: 1247 pg. of the datasheet.)

11. **RCGCTIMER:** The RCGCTIMER register provides software the capability to enable and disable 16/32-bit timer modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. (For More Details Refer Page No: 338 pg. of the datasheet.)

12. **GPTMCTL:** This register is used alongside the GPTMCFG and GMTMTnMR registers to fine-tune the timer configuration, and to enable other features such as timer stall and the output trigger. (For More Details Refer Page No: 737 pg. of the datasheet.)

13. **GPTMCFG:** This register configures the global operation of the GPTM module. The value written to this register determines whether the GPTM is in 32- or 64-bit mode (concatenated timers) or in 16- or 32-bit mode (individual, split timers). (For More Details Refer Page No: 727 pg. of the datasheet.)

14. **GPTMTAMR:** This register configures the GPTM based on the configuration selected in the GPTMCFG register. When in PWM mode, set the TAAMS bit, clear the TACMR bit, and configure the TAMR field to 0x1 or 0x2. (For More Details Refer Page No: 729 pg. of the datasheet.)

15. **GPTMTAILR:** When the timer is counting down, this register is used to load the starting count value into the timer. When the timer is counting, this register sets the upper bound for the timeout event. (For More Details Refer Page No: 756 pg. of the datasheet.)

16. **GPTMTAMATCHR:** This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode. In PWM mode, this value along with GPTMTAILR, determines the duty cycle of the output PWM signal. (For More Details Refer Page No: 758 pg. of the datasheet.)

17. **GPIOLOCK**: The GPIOLOCK register enables write access to the GPIOCR register. Writing 0x4C4F434B to the GPIOLOCK register unlocks the GPIOCR register. (For More Details Refer Page No: 684 pg. of the datasheet.)

18. **GPIOCR**: The GPIOCR register is the commit register. (For More Details Refer Page No: 685 pg. of the datasheet.)

19. **GPIOPUR**: The GPIOPUR register is the pull-up control register. When a bit is set, a weak pull-up resistor on the corresponding GPIO signal is enabled. (For More Details Refer Page No: 677 pg. of the datasheet.)

20. **GPIODEN** – The GPIODEN register is the digital enable register. (For More Details Refer Page No: 682 pg. of the datasheet.)

21. **GPIODIR** – The GPIODIR register is the data direction register. Setting a bit in the GPIODIR register configures the corresponding pin to be an output. (For More Details Refer Page No: 663 pg. of the datasheet.)

Introduction

The objective of this assignment is to design and implement a simple audio tone generator using the Tiva C board and its internal DAC and speaker. The Tiva C board is a microcontroller development board that is designed to provide a platform for building advanced embedded systems. The board features a TM4C123GH6PM microcontroller, which is a member of the ARM Cortex-M4F family of processors. This processor is known for its power and versatility, and it contains various peripherals and interfaces that can be used for developing sophisticated embedded systems.

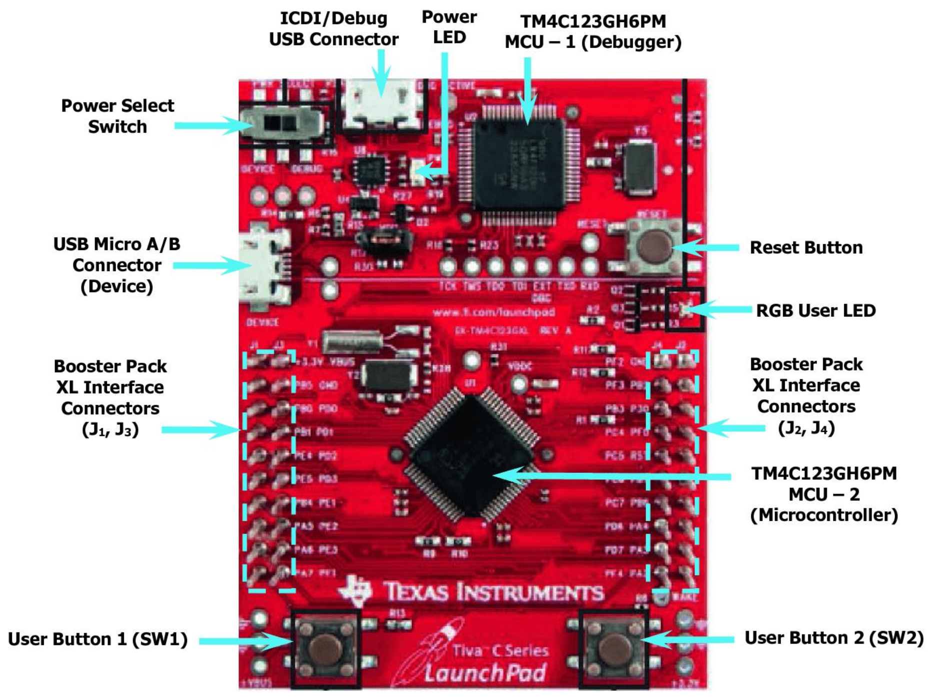

ICDI/Debug USB Connector
Power LED
TM4C123GH6PM MCU – 1 (Debugger)
Power Select Switch
USB Micro A/B Connector (Device)
Reset Button
RGB User LED
Booster Pack XL Interface Connectors (J1, J3)
Booster Pack XL Interface Connectors (J2, J4)
TM4C123GH6PM MCU – 2 (Microcontroller)
User Button 1 (SW1)
User Button 2 (SW2)

Figure 3: Tiva-C TM4C123GH6PM

In this assignment, we will utilize the GPIOs and timers on the Tiva C board to generate an audio tone with a specific frequency and waveform. The audio tone generator is a fundamental building block of many electronic devices, such as musical instruments, audio amplifiers, and sound effect generators. By designing and implementing an audio tone generator, we will gain practical experience in embedded system design and programming, as well as in the principles of digital signal processing and waveform synthesis.

To achieve our objective, we will first study the components of the Tiva C board, including the microcontroller, the GPIOs, and the timers. We will then design and implement the software that generates the audio tone waveform. We will also implement the hardware interface between the Tiva C board and the speaker.

Once we have designed and implemented the audio tone generator, we will test and validate its functionality and performance using a digital storage oscilloscope (DSO). Through this assignment, we aim to develop a working prototype of the tone generator and to demonstrate its functionality and performance. We hope to learn about the various components of the Tiva C board, the principles of digital signal processing, and the techniques for interfacing hardware and software.

Source Code

The source code for this project involves implementing a simple audio tone generator using a Tiva C board with an internal DAC and speaker. The code involves setting up the necessary GPIO pins, configuring the timers, and generating the desired waveform using digital signal processing techniques. The source code is then uploaded onto the Tiva C board using a programming interface, such as a USB cable.

1. Code 1:

1.1 code 1:

```
// Animesh Jain - 20uec020, Aniket Chelawat - 20uec019
#include "TM4C123GH6PM.h" // Including the header file TM4C123GH6PM.h

int main()
{
    // STEP-1: Clock setting for PWM module-1 and PORT-F (since the
M1PWM6 is used which is present at PORTF2, that is why we are enabling
PORTF)

    SYSCTL->RCGCPWM |= 1<<1; // Enable clock to PWM Module-1
    SYSCTL->RCGCGPIO |= 1<<5;// Enable clock to PORTF
    SYSCTL->RCC      &= (~(1<<20));// Directly feed system clock to PWM1
module without pre-divider

    // STEP-2: Setting the PF2 pin for M1PWM6 channel output pin

    GPIOF->AFSEL |= 1<<2;// Enabling the alternate function of PF2
    GPIOF->PCTL   = 0X00000500;// Make PF2 as PWM output pin
    GPIOF->DEN   |= 1<<2// Set PF2 as Digital Pin

    // STEP-3: Setting the PWM1 Channel 6

    PWM1->_3_CTL &= (~(1<<0)); // Disabling the generator 3 counter
    PWM1->_3_CTL &= (~(1<<1));// Select the down count mode of generator
3
    PWM1->_3_LOAD  = 16000;// Set the load value for generation of 1KHz
signal
    PWM1->_3_CMPA  = 8000; // Setting the duty cycle to 50% of the load
value(1600)
    PWM1->_3_GENA |= (1<<2) | (1<<3) | (1<<7); //
    PWM1->_3_CTL  |= (1<<0); // Enable Generator 3 counter
    PWM1->ENABLE  |= 1<<6; // Enable PWM1 channel 6 output

while(1)  // infinite while loop for regular working
    {  }
}
```

1.2 Working of code 1:

The main goal of this code is to generate a PWM signal on Pin PF2 of Tiva C board using M1PWM6 channel. The PWM signal is generated with a frequency of 1kHz and a duty cycle of 50%.

The first step of the code is to configure the clock for the PWM module-1 and PORT-F. The clock is enabled for PWM module-1 and PORT-F, and the system clock is directly fed to PWM1 module without pre-divider. Then, PF2 pin is configured as the PWM output pin by enabling its alternate function, setting it as a PWM output pin, and making it a digital pin.

In the next step, PWM1 Channel 6 is configured. Generator 3 counter is disabled, and down count mode is selected. The load value for the generation of 1kHz signal is set to 16000. The duty cycle is set to 50% of the load value (8000). The output is configured as a PWM signal with high and low pulses for one cycle of the signal. Generator 3 counter is enabled, and PWM1 channel 6 output is enabled.

The code then enters an infinite while loop to keep generating the PWM signal continuously.

Overall, this code sets up the Tiva C board to generate a 1kHz PWM signal with a 50% duty cycle on Pin PF2 using M1PWM6 channel. It demonstrates how to configure the clock, GPIOs, and PWM channels of the Tiva C board using registers. This code can be modified to generate different frequencies and duty cycles of the PWM signal, making it a useful foundation for further embedded system development.

2. Code 2:

2.1 Code 2:

```
// Animesh Jain - 20uec020, Aniket Chelawat - 20uec019
#include "TM4C123GH6PM.h" // Including the header file TM4C123GH6PM.h

#define PWM_SIGNAL_TIME_PERIOD 16000  // PWM Signal Time period in clock
cycles    (Calculated Using = Required Period / clock Period)
#define PWM_SIGNAL_DUTY_CYCLE  8000    // PWM Signal Time duty cycle in
clock cycles (50 % of the period of PWM signal)

int main()
{
    // STEP-1 Enabling the system clock to PORTF and Internal General
Purpose Timer

    SYSCTL->RCGCGPIO  |= 1<<5; // Enable clock to PORTF
    SYSCTL->RCGCTIMER |= 1<<1; // Enable system clock to TIMER-1 module
(T1CCP0 -> Timer-A of Timer-1 module)

    // STEP-2 Setting the Alternate function of PF2 and setting it as
Digital output pin

    GPIOF->AFSEL |= 1<<2;// Enabling the alternate function of PF2
    GPIOF->PCTL   = 0x00000700; // Alternate function --> T1CCP0
    GPIOF->DEN   |= 1<<2; // Set PF2 as Digital Pin
    GPIOF->DIR   |= 1<<2; // Set PF2 as Output Pin

    // STEP-3 Configuring the Timer-1 to work in PWM Mode and generate
PWM signals

    TIMER1->CTL     &= ~(1<<0); // Disable Timer 1
    TIMER1->CFG      = 0x04; // Set Timer 1 to 16-bit mode
    TIMER1->TAMR     = 0x0A; // Set Timer 1A to PWM mode
    TIMER1->TAILR       =  PWM_SIGNAL_TIME_PERIOD  - 1;  // Setting the
starting count value into the timer
    TIMER1->TAMATCHR = PWM_SIGNAL_DUTY_CYCLE  - 1; // Setting the duty
cycle of the output PWM Signal
    TIMER1->CTL     |= (1<<0);  // Enable Timer 1

while(1){}     // Infinitely running While Loop

    // We can also vary the duty cycle of the generated PWM signal by
maintaining a for loop inside the while loop and continuously
    // changing the value of TAMATCHR register.
}

// OUTPUT----> PWM Signal of 1KHz frequency is generated
```

2.2 Working of code 2:

This code is written to generate a Pulse Width Modulated (PWM) signal with a frequency of 1 kHz using the Tiva C Launchpad microcontroller board. The code is written in C programming language, and it uses the header file TM4C123GH6PM.h to access the microcontroller's register addresses.

The first step in the code is to enable the system clock to the PORTF and internal general-purpose timer of the microcontroller. This is done by setting the RCGCGPIO and RCGCTIMER bits in the SYSCTL register. Once the clock is enabled, the code moves to the second step of setting the alternate function of PF2 as the output pin for the PWM signal.

The third step in the code is to configure Timer-1 to work in PWM mode and generate the PWM signal. The TIMER1->CTL register is first disabled, and the TIMER1->CFG register is set to 16-bit mode. The TIMER1->TAMR register is then set to PWM mode, and the TIMER1->TAILR and TIMER1->TAMATCHR registers are set to generate the desired frequency and duty cycle of the PWM signal. Finally, the TIMER1->CTL register is enabled to start the timer.

The code then enters an infinite while loop to keep generating the PWM signal continuously. To change the duty cycle of the generated PWM signal, a for loop can be maintained inside the while loop by continuously changing the value of the TIMER1->TAMATCHR register.

Overall, this code generates a PWM signal with a frequency of 1 kHz and a duty cycle of 50% using Timer-1 of the Tiva C Launchpad microcontroller board.

Implementation on Hardware

1. Connections on breadboard

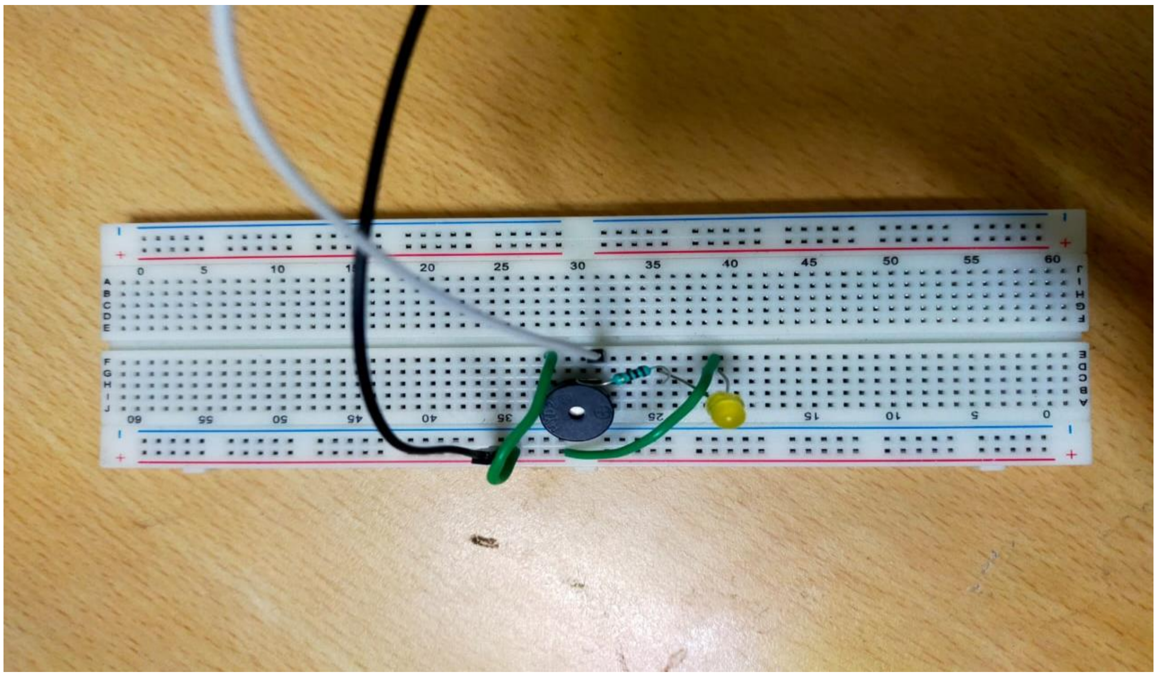
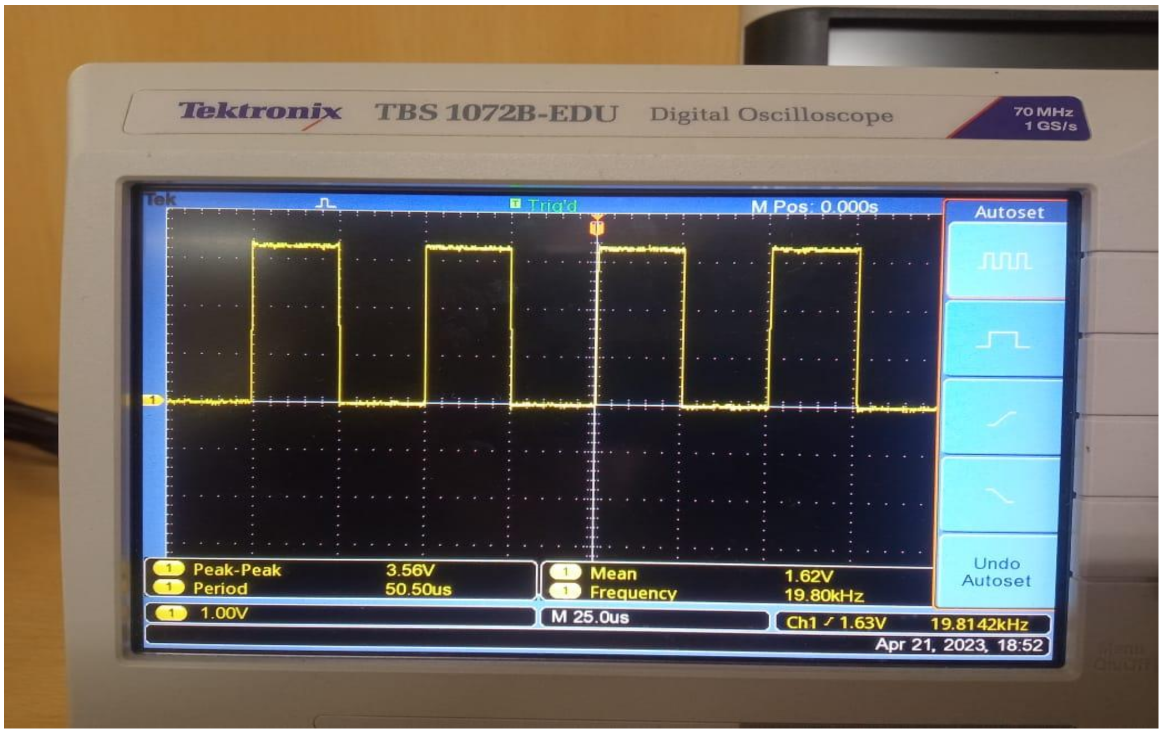


Figure 4: Connections on breadboard

2. Generated Pulse wave on DSO



```
Peak-Peak   3.56V      Mean        1.62V
Period      50.50us    Frequency   19.80kHz
1.00V                  M 25.0us    Ch1 ∕ 1.63V   19.8142kHz
                                   Apr 21, 2023, 18:52
```

Figure 5: Generated Wave

3. Complete setup

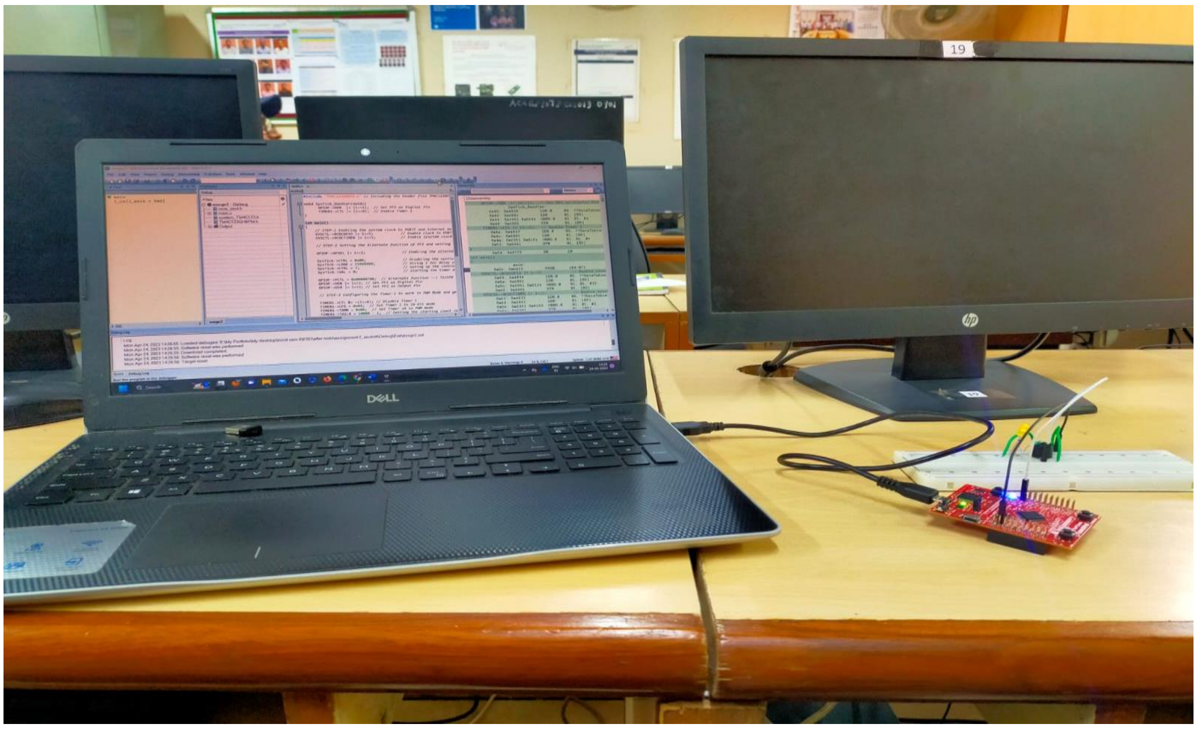


Figure 6: Setup with breadboard and Tiva-c board

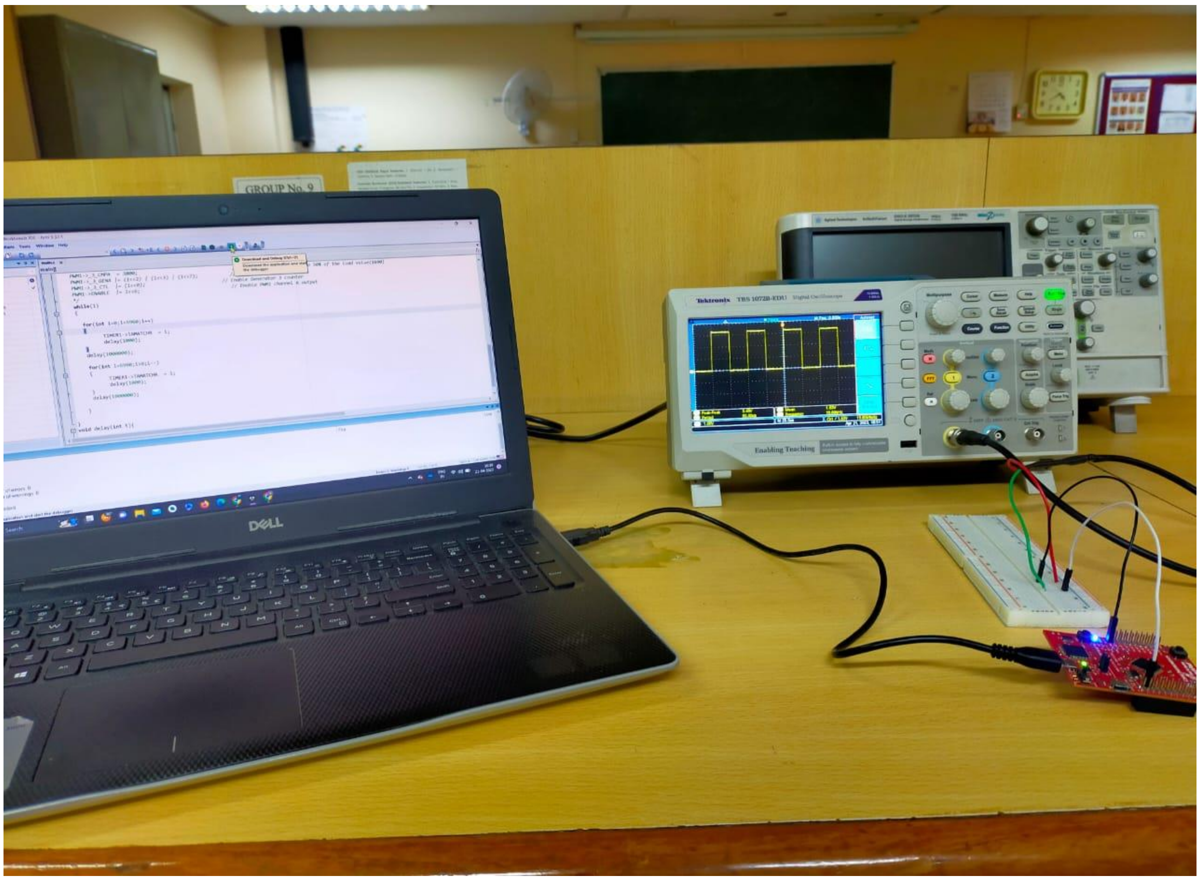


Figure 7: Setup with breadboard, Tiva-c board and DSO

Conclusion

In conclusion, we have successfully generated a PWM signal of 10kHz frequency using the Tiva C board and TM4C123GH6PM microcontroller. We started by enabling the system clock to PORTF and the internal General Purpose Timer, and then configured Timer-1 to work in PWM mode and generate PWM signals. We set the PWM signal time period and duty cycle according to our desired frequency and duty cycle using the appropriate registers, instead of Timer-1 we can also use PWM module and it’s channels to produce PWM waves. Finally, we verified the output by connecting an oscilloscope to the output pin and observing the waveform.

Overall, this project demonstrates the versatility and power of the Tiva C board for developing embedded systems with precise timing and control. The ability to generate PWM signals is essential for a wide range of applications, including motor control, power electronics, and LED lighting. With the knowledge gained from this project, we can further explore the capabilities of the Tiva C board and microcontroller and apply them to various real-world applications.

Bibliography

[1] Embedded Systems, ARM. (n.d.). Tiva™ TM4C123GH6PM Microcontroller Data Sheet. Retrieved from https://www.ti.com/lit/ds/symlink/tm4c123gh6pm.pdf.

[2] Embedded Systems, ARM. (n.d.). Tiva™ TM4C123GH6PM Microcontroller User's Guide. Retrieved from https://www.ti.com/lit/ug/spmu296/spmu296.pdf.

[3] Simon, D. E. (1999). An Embedded Software Primer. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

[4] https://www.youtube.com/watch?v=psY7JjcP-6I