

Embedded System Design Project Report

Final Report

Whac-A-Mole Game

A report submitted in part fulfilment Embedded System
Design course

3rd Year (6th Semester) in ECE

Supervisor: Dr. Deepak Nair



Designers: i) Gopal Agarwal (Roll No: 18uec115)

**Department of Electronics and Communication Engineering
The LNMIIT, Jaipur, Rajasthan, India**

September 14, 2021

Declaration

This report has been prepared on the basis of our own work and designs developed under Embedded System Design Course. Where other published and unpublished source materials have been used, these have been acknowledged.

Students Name:

1) Gopal Agarwal, Roll No: 18uec115

Date of Submission: 18/04/2021

Signature of Supervisor/ Supervisors: _____

Name of Supervisor/ Supervisors: Dr. Deepak Nair

Table of Contents

Abstract	4
Chapter 1: Introduction.....	5
1.1 Schematic	5
1.2 Main Components.....	6
Chapter 2: System Block Diagram	9
Chapter 3: Flow Chart and Code.....	11
3.1 Flow Chart.....	11
3.2 Component Used.....	11
Chapter 4: Source Code	12
Chapter 5: Observation	15
Chapter 6: Future Work	16
References.....	17

Table of Figures & Tables

Figure 1:	Whac a Mole Game	4
Figure 2:	Whac Controller Circuit Diagram.....	5
Figure 3:	TM4C123GH6PM High Level Block Diagram.....	6
Figure 4:	System Block Diagram	9
Figure 5:	Flow Chart	10
Figure 6:	Circuit Image	14
Table 1:	Absolute Maximum Rating	7
Table 2:	Recommended DC Operating Conditions	7
Table 3:	LED DC Operating Conditions	7
Table 4:	Push Button DC Operating Conditions	7
Table 5:	Component List	10

Abstract

This project is based on a popular game “Whac-A-Mole”. We used LED’s and Switches to design this game using TM4C123GH6PM Microcontroller.

The objective of this game is very simple a mole comes out from a random hole and within limited time player needs to hit it with the hammer. The Same logic is applied in our project.

- 8 Blue LED’s are used which represents the moles and correspondingly 8 switches are used to catch the LED.
- LED will glow randomly.
- LED will glow for 1sec and within this time user needs to catch it to win the game.
- On-Board Green led will glow to show user wins and On-Board Red led will glow to show user lose.
- SysTick Timer is used to generate 1sec time period.
- We have used the TM4C123GH6PM microcontroller as the controller for this game.

This Project has led to a better understanding of TM4C123GH6PM microcontroller and the basics of designing an electronic product.



Figure 1: Whac a Mole Game

Chapter 1: Introduction

“Whac-A-Mole” is an easy and engaging game. In this game there is matrix of holes and in any one random hole a mole appears and the objective of this game is to smash that mole using hammer before it gets in the hole to get the points.

I have designed this game using the LED's and Switches. TM4C123GH6PM microcontroller is used to control the working of this game.

8 LED's matrix are used to represent the moles and correspondingly 8 switches are used to catch that LED. As soon as the game starts a random LED will glow for 1sec and within this time user needs to press the corresponding switch to make the LED off and win the game.

If the user presses the wrong switch or run out of time i.e. not being able to press the switch within time, then that LED will automatically get off after 1sec and the user lose the game.

Every successful try makes the On-board Green LED glow to represent that user “WINS” and every wrong attempt will make On-Board Red LED glow to represent that user “LOSE”.

Now 1sec delay is given before the next random LED glow and again the user needs to catch that LED by pressing the correct switch. There are two stages namely Stage 1 and stage 2.

In stage 1 only 1 random LED glow and it continues for next 10 turns then Stage 2 gets start and in this stage 2 LED get ON simultaneously and it continues for next 10 turns.

Then game gets over and final score will be displayed on the LCD.

1.1 Schematic

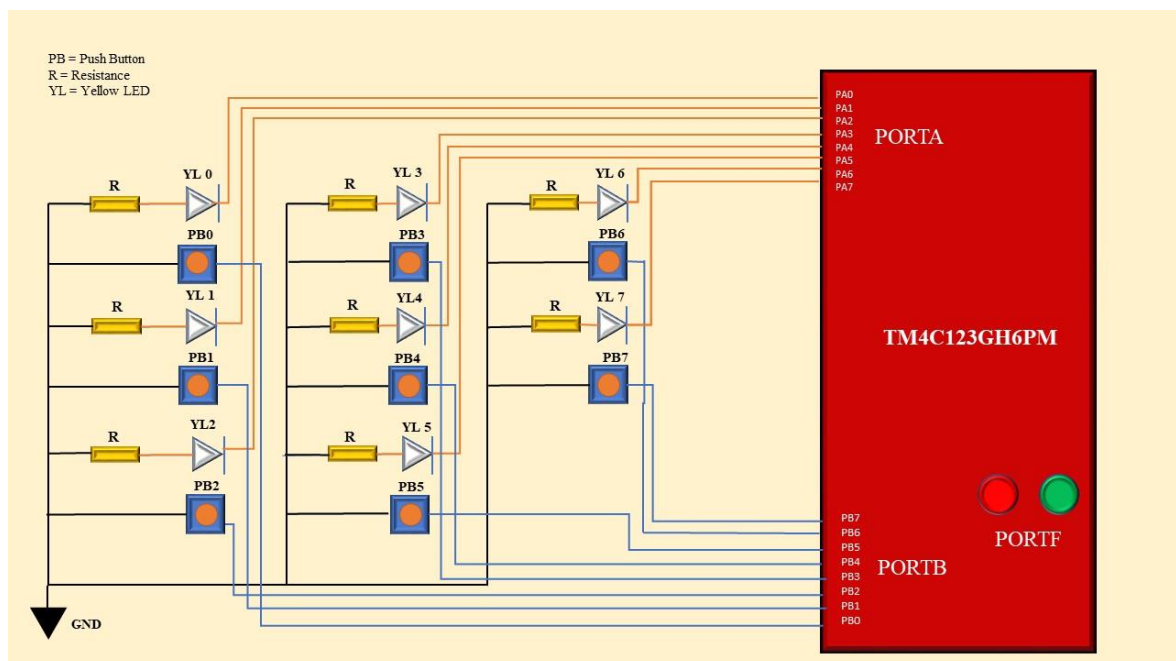


Figure 2: Whac Controller Circuit Diagram

We have used TM4C123GH6PM Microcontroller, LED's, Push Switches, Resistors and IAR software IDE to program it.

1.2 Main Components

1. TM4C123GH6PM TI Microcontroller

A. General Specification

- It is Texas Instruments Tiva™ C Series microcontroller based on 32-bit ARM Cortex M4 Architecture.
- It offers 80Mhz Cortex-M with FPU, multiple programmable peripherals and comprehensive libraries of software tools to minimize the design-cycle time.
- It uses ARM's Thumb®-compatible Thumb-2 instruction set to reduce memory requirements.
- Suitable for high performance, significant control processing and even for low power applications.
- Outstanding Processing performance and fast interrupt handling capacity.
- Enhanced system debug with extensive breakpoint capabilities.
- Ultra-low power consumption with integrated sleep modes.
- Following is the block diagram to get an overview of the integrated devices:

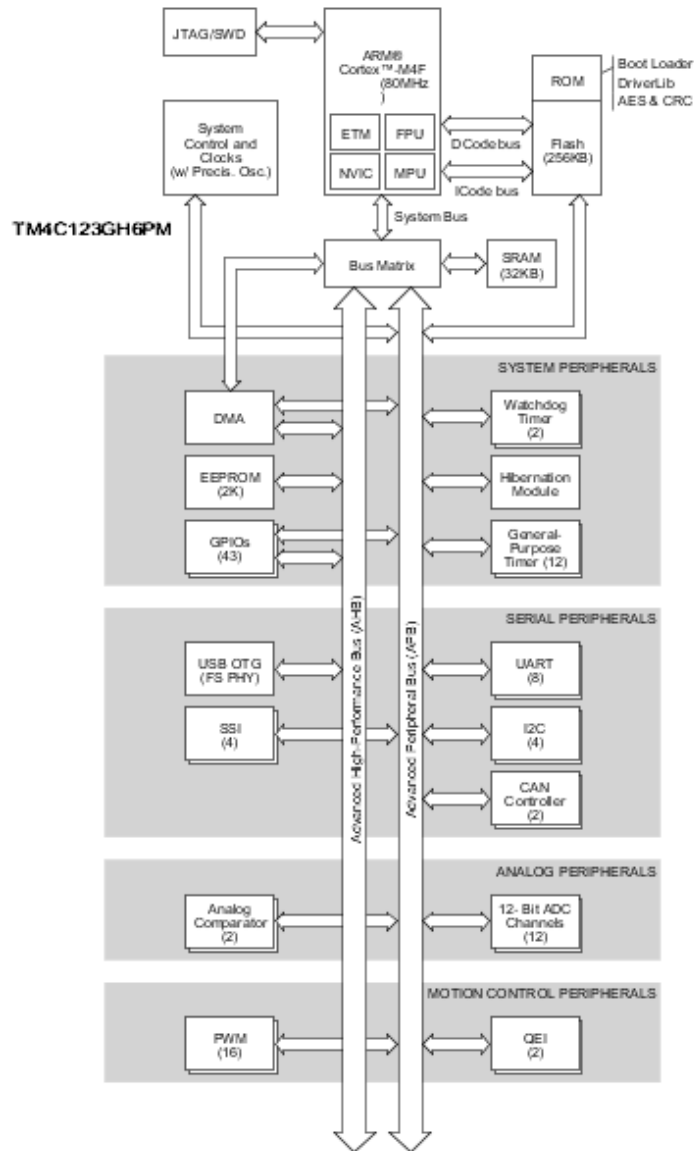


Figure 3: TM4C123GH6PM High Level Block Diagram

B. Electrical Characteristics

Parameter	Parameter Name	Value		Unit
		Min	Max	
V_{DD}	V_{DD} supply voltage	0	4	V
V_{IN_GPIO}	Input voltage on GPIOs, regardless of whether the -0.3 5.5 V microcontroller is powered	-0.3	5.5	V
	Input voltage for PD4, PD5, PB0 and PB1 when -0.3 $V_{DD} + 0.3$ V configured as GPIO	-0.3	$V_{DD} + 0.3$	V
I_{GPIO_MAX}	Maximum current per output pin	-	25	mA

Table 1 Absolute Maximum Ratings

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{DD}	V_{DD} supply voltage	3.15	3.3	3.63	V
VOH	GPIO high-level output voltage	2.4	-	-	V
VOL	GPIO low-level output voltage	-	-	0.4	V
IOH	High-level source current, VOH=2.4 V				

	2-mA Drive	2.0	-	-	mA
--	------------	-----	---	---	----

Table 2: Recommended DC Operating Conditions

2. LED

- Calculation for Series Resistance with LED

$$R = \frac{V_{oh} - V_{led}}{I}$$

- V_{oh} (GPIO high level output voltage) = 2.4V
- V_{led} = Drop voltage across LED
- I (GPIO output current) = 2mA
- R1 (Yellow LED) =
- 150 ohms

Color	Parameter Name	Value	Unit
Yellow	Voltage Drop	2.1	V

Table 3: LED Drop Voltage

3. Push Buttons

- Electrical Characteristics

Parameter	Parameter Name	Max	Unit
V_{PB}	Voltage	12	V
I_{PB}	Current	100	mA
P_{PB}	Power Rating	1.2	W

Table 4: Push Button DC Operating Conditions

4. Software

a. Coding for LED's

For programming GPIO's connected to LED's first they are provided with clock to enable them using RCGCGPIO register.

Then Digitally Enable the GPIO Ports using GPIODEN register and lastly set pins as input or output as required using GPIODIR register.

Sequence of registers used: RCGCGPIO → GPIODEN → GPIODIR.

b. Coding for Switches

For programming the GPIO's connected to switch matrix first they are provided with clock to enable them using RCGCGPIO register. Some GPIO pins are locked by default for protection purpose from accidental programming for so we used lock and commit Register to unlock them.

After that we enable internal pull up resistor using RCGCPUR register and lastly make pins digitally enable using RCGCDEN register and choose pins as input using RCGCDIR register.

Sequence of registers used: RCGCGPIO → GPIOLOCK → GPIOCR → GPIOPUR → GPIODEN → GPIODIR.

Chapter 2: System Block Diagram

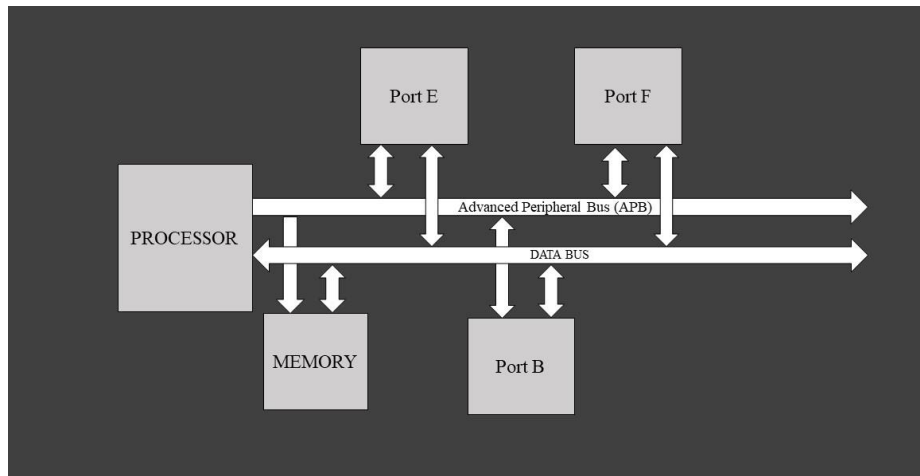


Figure 4: System Block Diagram

Details of Block Used

❖ GPIO Ports

- GPIO module consist of 6 GPIO ports namely port A, B, C, D, E and F.
- It supports up to 43 programmable input/output pins.
- 5-V tolerant input configuration
- **Port E** – Port E pins are used to connect LED's.
- **Port B** – Port B pins are used to connect switches.
- **Port F** – PF.1 and PF.3 pins are used to glow RED and GREEN LED.

❖ Memory – SRAM, FLASH, STACK memory are used in this project.

- **SRAM** -- To store variables.
- **Flash memory** -- To store program.
- **Stack memory** -- To store last address before function calls.

❖ Processor – 32-bit ARM® Cortex™-M4F processor with 80-MHz operation clock speed.

Chapter 3: Flow Chart and Code

3.1 Flow Chart

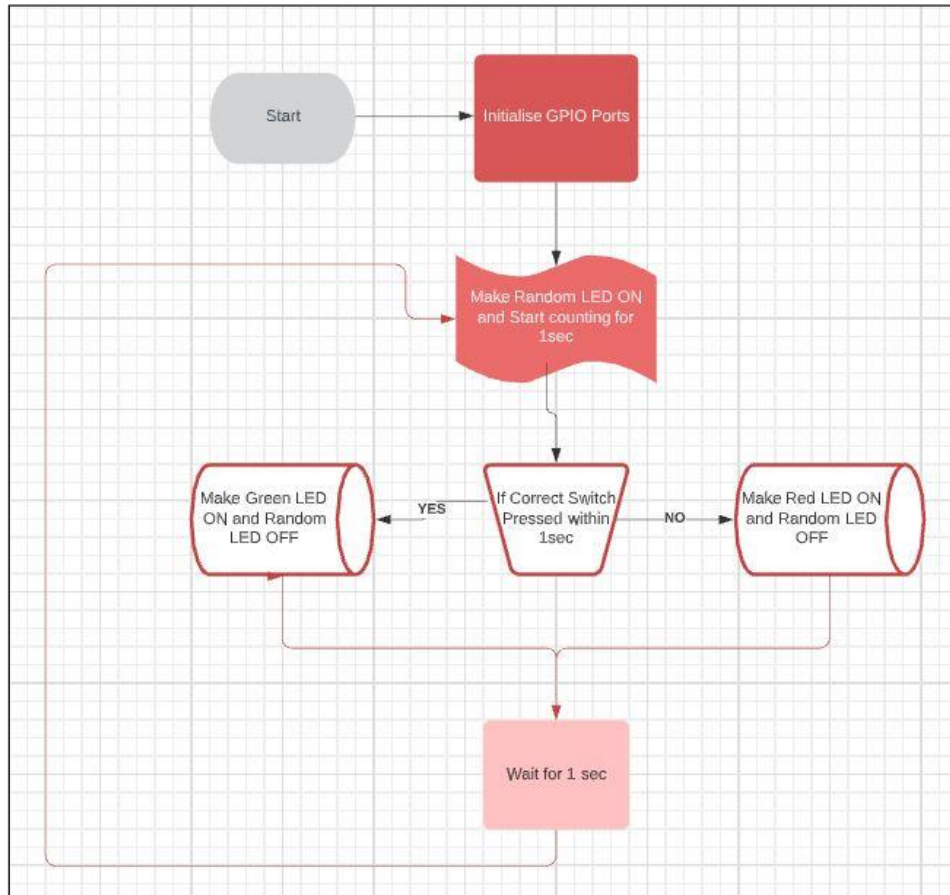


Figure 5: Flow Chart

3.2 Component Used

Name	Quantity
TM4C123GH6PM	1
Yellow LED	8
Push Button	8
Resistance (150 ohm)	8
Breadboard	1

Table 5: Component List

Chapter 4: Source Code

- **Main File.c**

```

/* Whac-A-Mole Game Main Source File

ESD Course Project
Developer: Gopal Agarwal & Ayush mangal
Created: April 17, 2021

Hardware Resources:
- TM4C123GH6PM Board
- Switches and LED's

*/

#include "TM4C123GH6PM.h"    // Standard library for TM4C123GH6PM
#include "stdlib.h"          //Standard library for C used to access rand() function
#include "my_header_file.h"  // User generated library

void Check_switch(unsigned int value);
void Random_LED_On(void);
void next_round(void);
int COUNT = 800000;
int STATUS = 0;

//This function is used to generate any random value in between 0-7 and the corresponding LED will
turn ON
void Random_LED_On(void)
{
    unsigned int index = rand() % 5;    // Generating a random value from 0-7
    unsigned int value = dec_to_hex(index);    // Converting random generated decimal value into
    hexadecimal using dec_to_hex user defined function

    GPIOE -> DATA = value;    // Make random LED ON
    Check_switch(value);
}

// This function is used to check the input response of the user
void Check_switch(unsigned int value)
{
    while(COUNT-->0)    // Polling technique used to check input response
    {
        if(!(GPIOB->DATA & value ) )    // If correct switch pressed
        {
            STATUS = 1;
            GPIOE -> DATA = 0x00;    // Make random LED OFF
            GPIOF -> DATA = 0X08;    // Make Green LED ON to show user win
        }
    }

    COUNT = 800000;
    next_round();
}

// This function is used to call for next round
void next_round(void)
{
    if(STATUS == 0)    // If user presses wrong switch or run out of time

```

```

    {
        GPIOE -> DATA = 0x00;    // Make Random LED OFF
        GPIOF -> DATA = 0x02;    // Make RED LED ON to show user lose
    }

    delay(1999999);                // Approx 1sec delay given for next round
    GPIOF -> DATA = 0x00;        // Make RED LED OFF
    STATUS = 0;
    Random_LED_On();
}

int main()
{
    // RCGCGPIO Register Enables Clock. Refer page:- 340
    SYSCTL -> RCGCGPIO = 0x32;    // 0b0011 0010 Ports B, E, F clock Enabled

    // Port E is used to connect LED matrix.
    // DIR Register sets pins as I/O, 0->input and 1->output, Refer page:- 663
    GPIOE -> DIR = 0xFF;          // 0b1111 1111 all pins are set as output

    // DEN Register enables pins digitally Refer page:- 682
    GPIOE -> DEN = 0xFF;          // 0b1111 1111 all pins are digitally enabled

    // Port B is used to connect switch matrix.
    // LOCK register is used to unlock the pins of port E. Refer page:- 684
    GPIOB -> LOCK = 0x4C4F434B;

    // Commit Register is used to determine which bits of the GPIOAFSEL, GPIOPUR, GPIOPDR,
    and GPIODEN registers are committed. Refer Page:- 685
    GPIOB -> CR = 0xFF;           // 0b1111 1111 all 7 bits are unlocked and can be
    programmed.

    // PUR register enables internal pull up resistor Refer page:- 677
    GPIOB -> PUR = 0xFF;          // 0b1111 1111 pull up resistor enabled on all pins

    // DIR Register sets pins as I/O, 0->input and 1->output, Refer page:- 663
    GPIOB -> DIR = 0x00;          // 0b0000 0000 all pins are set as input

    // DEN Register enables pins digitally Refer page:- 682
    GPIOB -> DEN = 0xFF;          // 0b1111 1111 all pins are digitally enabled

    // Port F is used to show the output
    // DIR Register sets pins as I/O, 0->input and 1->output, Refer page:- 663
    GPIOF -> DIR = 0x0A;          // 0b0000 1010 PF.1 and PF.3 pins are set as output

    // DEN Register enables pins digitally Refer page:- 682
    GPIOF -> DEN = 0x0A;          // 0b0000 1010 PF.1 and PF.3 pins are digitally enabled

    while(1)
    {
        Random_LED_On();
    }
}

```

- **My.h**

```

#ifndef __MY_HEADER_FILE_H__
#define __MY_HEADER_FILE_H__

unsigned int dec_to_hex(unsigned int index);
void delay(int iteration);
#endif

```

- **My.c**

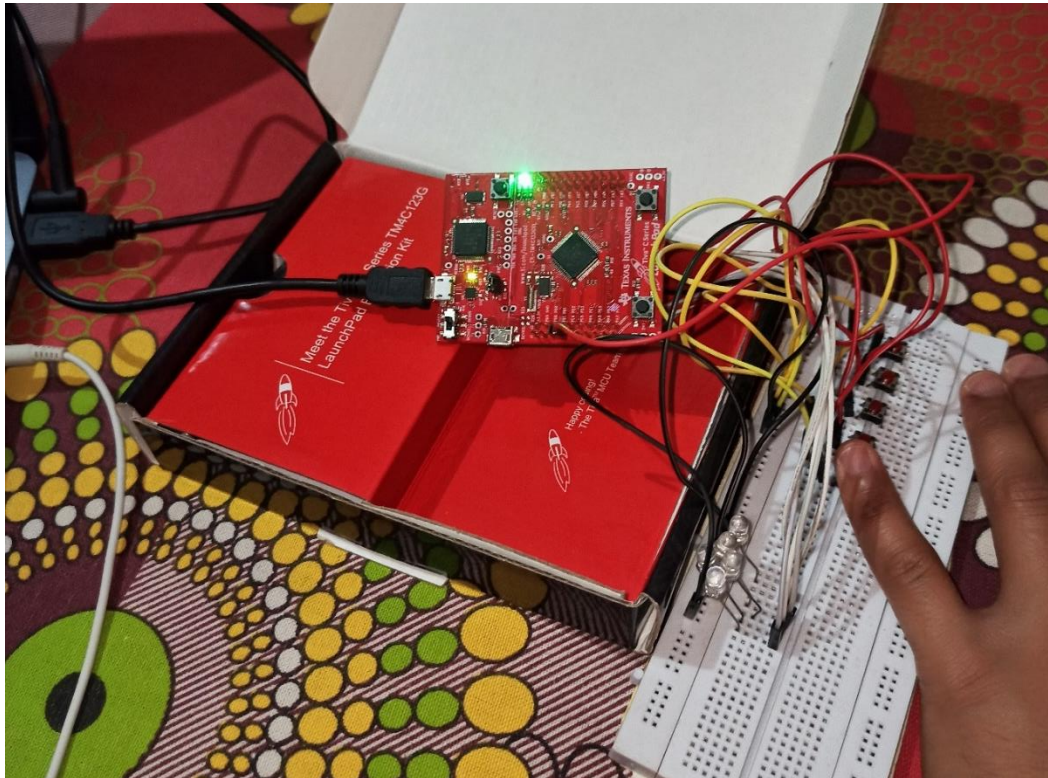
```
#include "my_header_file.h"

// Converting decimal value into hexadecimal value
unsigned int dec_to_hex(unsigned int index)
{
    unsigned int value;
    switch(index){
        case 0: value=0x01;           //0b0000 0001 0th LED will glow
            break;
        case 1: value=0x02;           //0b0000 0010 1st LED will glow
            break;
        case 2: value=0x04;           //0b0000 0100 2nd LED will glow
            break;
        case 3: value=0x08;           //0b0000 1000 3rd LED will glow
            break;
        case 4: value=0x10;           //0b0001 0000 4th LED will glow
            break;
        case 5: value=0x20;           //0b0010 0000 5th LED will glow
            break;
        case 6: value=0x40;           //0b0100 0000 6th LED will glow
            break;
        case 7: value=0x80;           //0b1000 0000 7th LED will glow
            break;
    }
    return value;
}

// Function used to generate delay
void delay(int iteration)
{
    int volatile counter;
    counter = 0;
    while(counter < iteration)
        ++counter;
}
```

Chapter 5: Observation

- Approx. 1sec delay is given before next event starts.
- Approx. 0.6sec is given to the player for pressing the switch.
- Total Power Consumption =
- Total Memory Used=



Chapter 6: **Future Work**

1. Interrupt technique can be used to utilize the microcontroller resources efficiently.
2. We can use LED embedded switches to decrease the ports used.
3. Limit for wrong attempts can set and if it exceeds game will automatically stop.
4. Different Difficulty levels can be added. Like at 2nd level simultaneously 2 LED will glow and so on.
5. LCD can be introduced to show the score, present difficulty level, wrong attempt numbers to make game more attractive.
6. Different Sounds can be added which can be played during winning or losing stage.
7. Size of LED and Switch matrix can be increased to 4*4.
8. LED's can be equipped with transparent small mole faces to make game more fancy.

References

1. TM4C123GH6PM Datasheet
<https://www.ti.com/lit/ds/spms376e/spms376e.pdf?ts=1618556278887>
2. Youtube.com. “Embedded System Designs”
https://www.youtube.com/playlist?list=PL3ZmPhf_xiU7BSoyx4vHZ6n115Wc7pCZ