Lab#03

Traffic Light Controller using FSM

By
Hitesh Vijay Oswal
Ankit Vilasrao Komawar

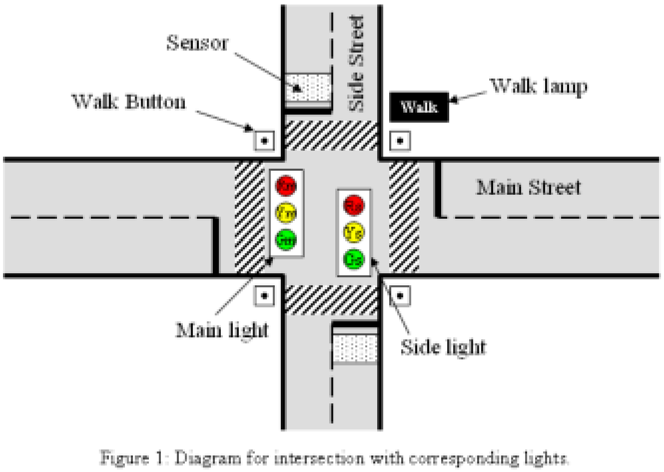
EECE 544: Digital Systems Design

Date: 10/16/2015

I. INTRODUCTION

1. **Objective:**

To develop a Traffic Light Controller for 2 – 2 way streets using Finite State Machine(FSM) in C Language on Tiva Board. The System has 5 inputs in total, out of which 4 inputs represents walk signal request and 1 input represents Side Street signal request. Main Street has highest priority.



Figure#1[1]: Traffic Signal System

2. **Basic Theory:**

The system is designed using C language. Finite State Machine (FSM) is used to design the system. The Delay between to states is provided using SysTick. Linked List Structure is used in C Language to represent single State. A Structure includes Outputs to be given on 2 ports, delay of the present state and pointer array of next states depending upon the Interrupts occurred. A pointer is used to point the present state. This pointer is updated to change the state.

The controller used in the Tiva board is TM4C123G. This controller works on 80MHz clock frequency, which is obtained using PLL with the input frequency of 16MHz. It has 6 Ports and verity of different functions. The development tool used is Keil µVision MDK-ARM.

II. BODY

1. **Hardware Connections:**

In this system 3 Ports are used and they are, Port B, Port E and Port F.

i. **Port B:** In this there 3 PINs are configured as output PINs. LEDs are used to represent outputs. All LEDs are connected in active high configuration. PB-3 represents Green light for Side Street, PB-4 represents Yellow light for Side Street and PB-5 represents Red light for Side Street.

ii. **Port E:** In this there 4 PINs are configured as output PINs and 2 PINs are configured as Input PINs. LEDs are used to represent outputs and normally open switches are used to get inputs. All LEDs are connected in active high configuration and Switches are connected in Negative logic configuration. PE-3 represents Green light for Main Street, PE-4 represents Yellow light for Main

Street and PE-5 represents Red light for Main Street. PE-2 represents Green light for Walk Signal. 2 walk signal request inputs are given on PE-1 and PE-0. These PINs have active Internal Pull-up resistors.

iii. **Port F:** In this there 3 PINs are configured as Input PINs. Normally open switches are used to get inputs. All the switches are connected in Negative logic configuration. All the 3 PINs have active Internal Pull-up resistors. PINs PF-0 and PF-1 are used for walk signal request inputs and PF-2 is used for Side Street Signal Request (Sensor).

2. PLL Configuration[2]:

XTAL	*Crystal Freq (MHz)*		*XTAL*	*Crystal Freq (MHz)*
0x0	Reserved		0x10	10.0 MHz
0x1	Reserved		0x11	12.0 MHz
0x2	Reserved		0x12	12.288 MHz
0x3	Reserved		0x13	13.56 MHz
0x4	3.579545 MHz		0x14	14.31818 MHz
0x5	3.6864 MHz		0x15	16.0 MHz
0x6	4 MHz		0x16	16.384 MHz
0x7	4.096 MHz		0x17	18.0 MHz
0x8	4.9152 MHz		0x18	20.0 MHz
0x9	5 MHz		0x19	24.0 MHz
0xA	5.12 MHz		0x1A	25.0 MHz
0xB	6 MHz (reset value)		0x1B	Reserved
0xC	6.144 MHz		0x1C	Reserved
0xD	7.3728 MHz		0x1D	Reserved
0xE	8 MHz		0x1E	Reserved
0xF	8.192 MHz		0x1F	Reserved

Table 4.9a. XTAL field used in the SYSCTL_RCC_R register of the TM4C123.

Table#1: XTAL Field used in the SYSCTL_RCC_R

Address	26-23	22	13	11	10-6	5-4	Name
$400FE060	SYSDIV	USESYSDIV	PWRDN	BYPASS	XTAL	OSCSRC	SYSCTL_RCC_R
$400FE050					PLLRIS		SYSCTL_RIS_R
	31	30	28-22	13	11	6-4	
$400FE070	USERCC2	DIV400	SYSDIV2	PWRDN2	BYPASS2	OSCSRC2	SYSCTL_RCC2_R

Table 4.9b. Main clock registers for the TM4C123.

Table#2: Main Clock register

i. Use RCC2 because it provides for more options.
ii. The first step is set BYPASS2 (bit 11). At this point the PLL is bypassed and there is no system clock divider.
iii. The second step is to specify the crystal frequency in the four XTAL bits using the code in Table 4.9. The OSCSRC2 bits are cleared to select the main oscillator as the oscillator clock source.
iv. The third step is to clear PWRDN2 (bit 13) to activate the PLL.
v. The fourth step is to configure and enable the clock divider using the 7-bit SYSDIV2 field. If the 7-bit SYSDIV2 is n, then the clock will be divided by n + 1. To get the desired 80 MHz from the 400 MHz PLL, we need to divide by 5. So, we place a 4 into the SYSDIV2 field.
vi. The fifth step is to wait for the PLL to stabilize by waiting for PLLRIS (bit 6) in the SYSCTL_RIS_R to become high.
vii. The last step is to connect the PLL by clearing the BYPASS2 bit.

3. SysTick:

SysTick is used to provide delay between the two state transitions. It is configured as follows:

Address	31-24	23-17	16	15-3	2	1	0	Name
$E000E010	0	0	COUNT	0	CLK_SRC	INTEN	ENABLE	NVIC_ST_CTRL_R
$E000E014	0	24-bit RELOAD value						NVIC_ST_RELOAD_R
$E000E018	0	24-bit CURRENT value of SysTick counter						NVIC_ST_CURRENT_R

Table 4.10. SysTick registers.

Table#3[3]: SysTick Registers

i. Disable the timer by resetting (writing 0) Enable (Bit-0 of (NVIC_ST_CTRL_R)).
ii. Write the Reload Value to Reload Register. This value is calculated using the following formulae:

$$\text{Reload Value} = \text{Time} \times \text{Frequency}$$

iii. Clear the current register by writing any value to it.
iv. Enable the timer by setting (writing 1) Enable (Bit-0 of (NVIC_ST_CTRL_R)).

4. Interrupts:

Pressing of any switch will interrupt the controller. Following Steps are done to initialize the interrupts:

i. Clear the GPIO_PORTx_IS_R for edge trigger interrupt
ii. Clear GPIO_PORTx_IBE_R for interrupt on single edge i.e. either rising or falling edge.
iii. Clear GPIO_PORTx_IEV_R for falling edge interrupt.
iv. Set the bits of GPIO_PORTx_IM_R for the respective PIN(s) to Arm Interrupts on the PINs.
v. Set GPIO_PORTx_ICR_R of the respective PIN(s) to CLEAR RIS bit of respective PIN(s)
vi. Set respective bit of NVIC_EN0 or NVIC_EN1 to enable interrupts on the respective Ports.

Address	31	30	29-7	6	5	4	3	2	1	0	Name
0xE000E100	G	F	...	UART1	UART0	E	D	C	B	A	NVIC_EN0_R
0xE000E104			...						UART2	H	NVIC_EN1_R

Table 9.3. The LM3S/TM4C NVIC interrupt enable registers.

Table#4[4]: NVIC Interrupt Enable Registers

vii. Set the interrupt priority in NVIC_PRIx_R register according to Table#5.
viii. Enable Global Interrupt by calling EnableInterrupts().

Address	31 – 29	23 – 21	15 – 13	7 – 5	Name
0xE000E400	GPIO Port D	GPIO Port C	GPIO Port B	GPIO Port A	NVIC_PRI0_R
0xE000E404	SSI0, Rx Tx	UART1, Rx Tx	UART0, Rx Tx	GPIO Port E	NVIC_PRI1_R
0xE000E408	PWM Gen 1	PWM Gen 0	PWM Fault	I2C0	NVIC_PRI2_R
0xE000E40C	ADC Seq 1	ADC Seq 0	Quad Encoder	PWM Gen 2	NVIC_PRI3_R
0xE000E410	Timer 0A	Watchdog	ADC Seq 3	ADC Seq 2	NVIC_PRI4_R
0xE000E414	Timer 2A	Timer 1B	Timer 1A	Timer 0B	NVIC_PRI5_R
0xE000E418	Comp 2	Comp 1	Comp 0	Timer 2B	NVIC_PRI6_R
0xE000E41C	GPIO Port G	GPIO Port F	Flash Control	System Control	NVIC_PRI7_R
0xE000E420	Timer 3A	SSI1, Rx Tx	UART2, Rx Tx	GPIO Port H	NVIC_PRI8_R
0xE000E424	CAN0	Quad Encoder 1	I2C1	Timer 3B	NVIC_PRI9_R
0xE000E428	Hibernate	Ethernet	CAN2	CAN1	NVIC_PRI10_R
0xE000E42C	uDMA Error	uDMA Soft Tfr	PWM Gen 3	USB0	NVIC_PRI11_R
0xE000ED20	SysTick	PendSV	--	Debug	NVIC_SYS_PRI3_R

Table 9.2. The LM3S/TM4C NVIC registers. Each register is 32 bits wide. Bits not shown are zero.

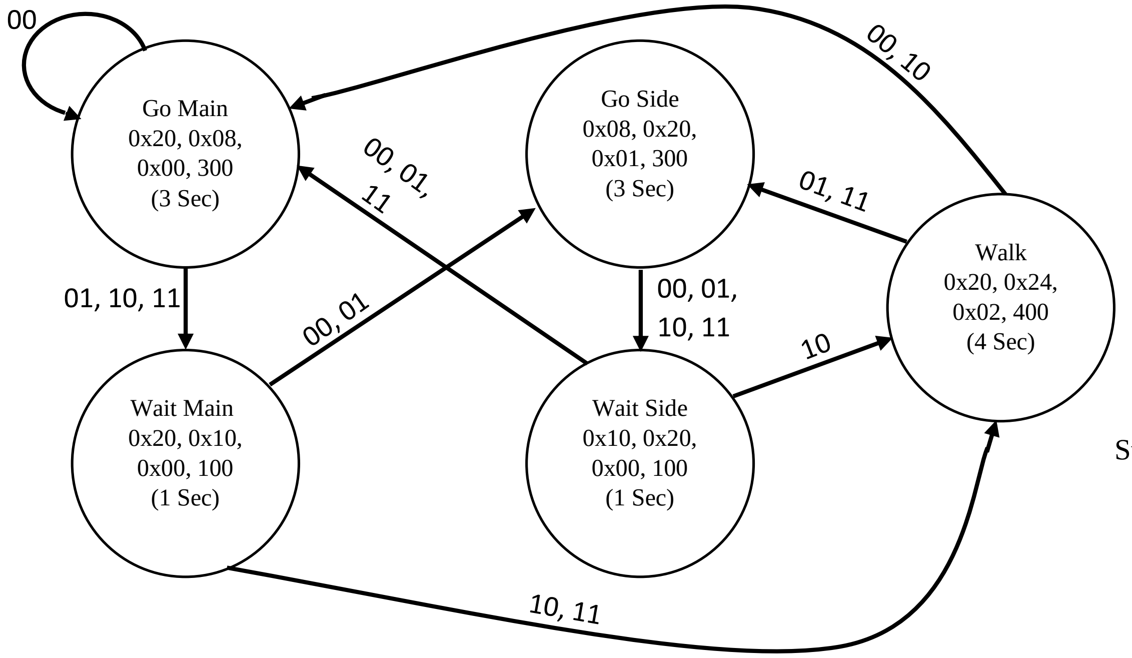
Table#5[5]: NVIC registers. Each register is 32 bits wide. Bits not shown are zero.

5. Finite State Machine(FSM) State Table and State Graph:

INPUT / Present State and output	00	01	10	11
	Next State			
Go main 0x20, 0x08, 0x00, 300 (3 Sec)	Go main	Wait Main	Wait Main	Wait Main
Wait Main 0x20, 0x10, 0x00, 100 (1 Sec)	Go Side	Go Side	Walk	Walk
Go Side 0x08, 0x20, 0x01, 300 (3 Sec)	Wait Side	Wait Side	Wait Side	Wait Side
Wait Side 0x10, 0x20, 0x00, 100 (1 Sec)	Go main	Go main	Walk	Go main
Walk 0x20, 0x24, 0x02, 400 (4 Sec)	Go main	Go Side	Go main	Go Side

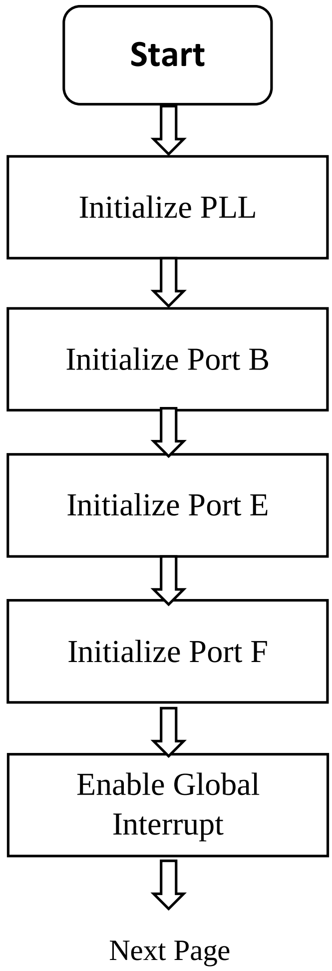
Table#6: State Table

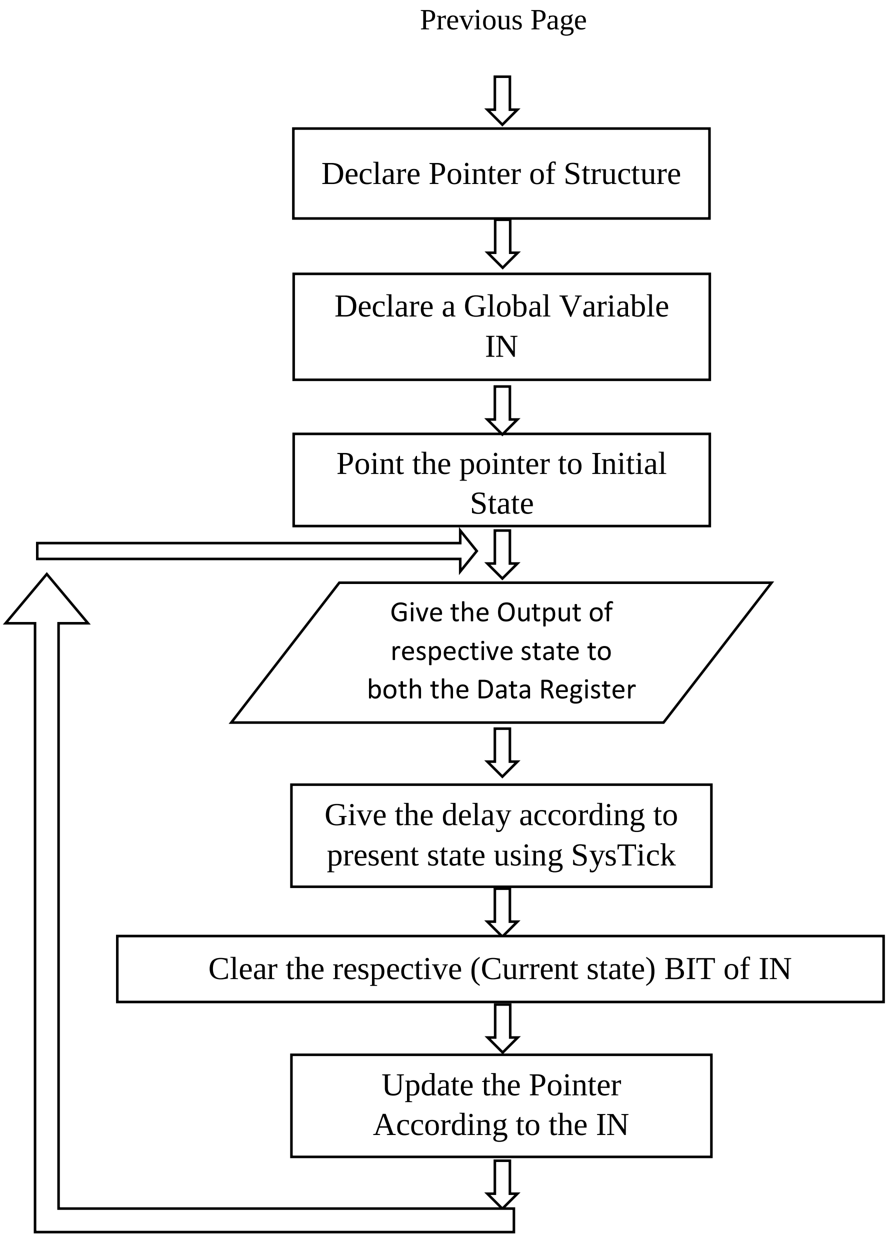
NOTE: In this Experiment a delay of 10ms is initialized and called multiple times to obtain required delay.



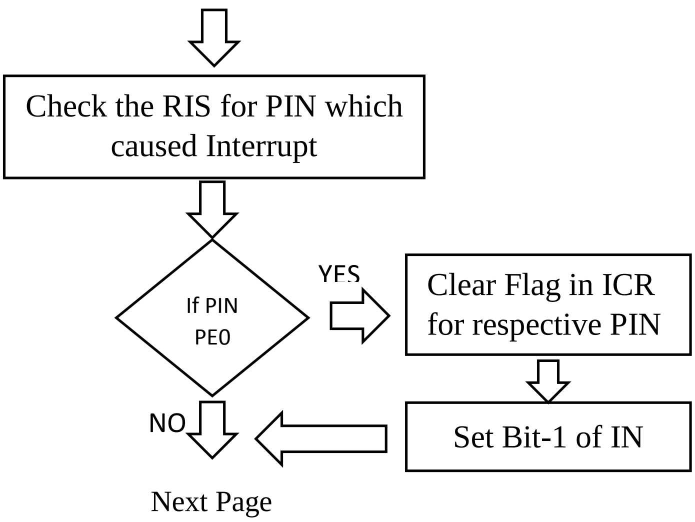
Figure#2:
State Graph

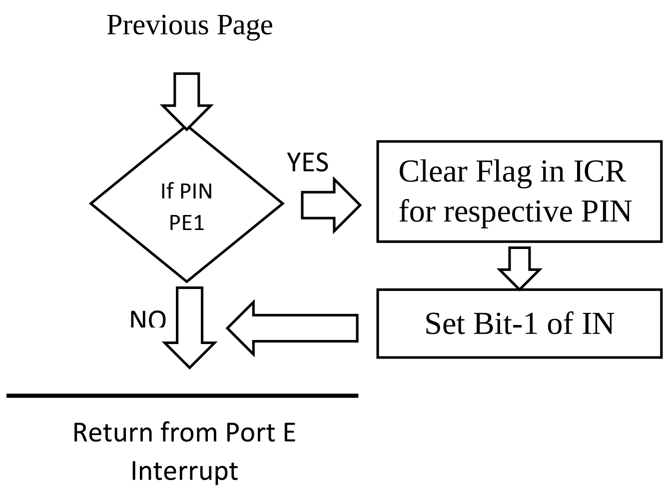
6. Flow Chart:

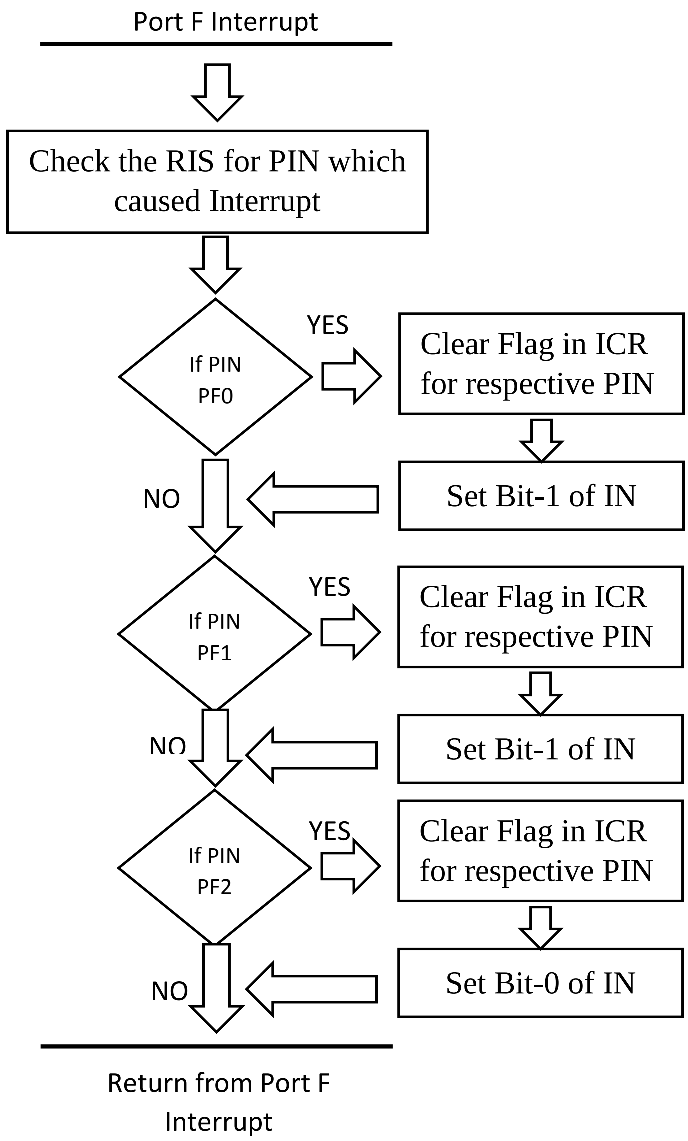




Port E Interrupt




Previous Page
If PIN PE1
YES
Clear Flag in ICR for respective PIN
Set Bit-1 of IN
NO
Return from Port E Interrupt


Port F Interrupt
Check the RIS for PIN which caused Interrupt
If PIN PF0
YES
Clear Flag in ICR for respective PIN
Set Bit-1 of IN
NO
If PIN PF1
YES
Clear Flag in ICR for respective PIN
Set Bit-1 of IN
NO
If PIN PF2
YES
Clear Flag in ICR for respective PIN
Set Bit-0 of IN
NO
Return from Port F Interrupt

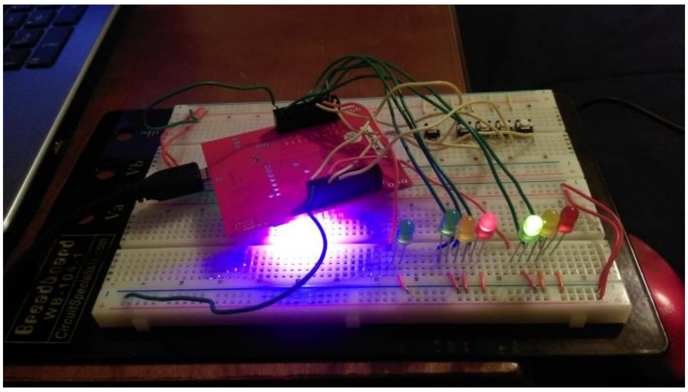
III. Conclusion

In this experiment we learnt about Finite State Machine (FSM) and its implementation using C language. We also learnt about initialization and implementation SysTick, PLL and Edge triggered interrupts.

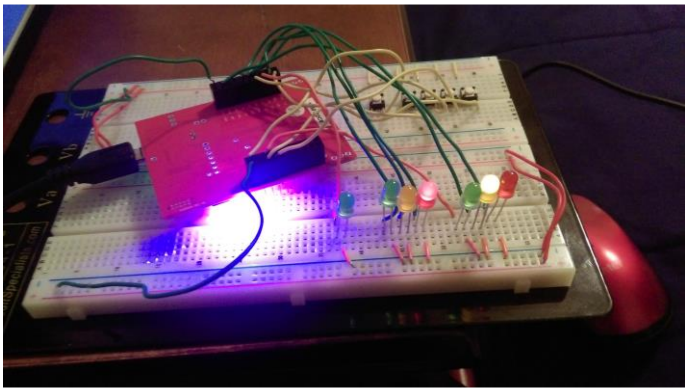
The System was Designed and Developed successfully.

Following are the Images of outputs during different states:

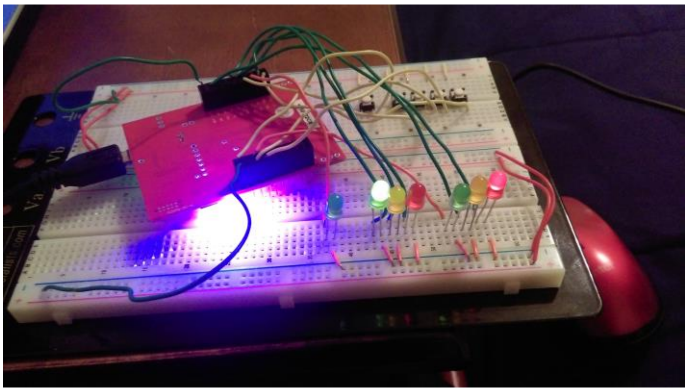
1. Go Side



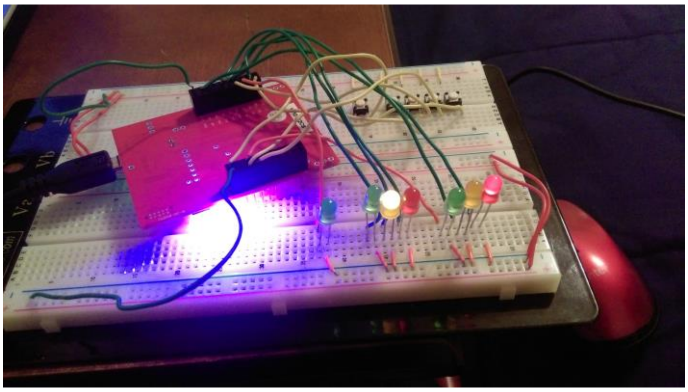
2. Wait Side



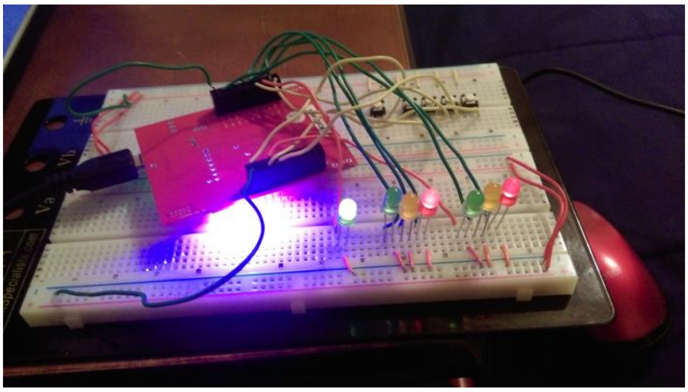
3. Go Main



4. Wait Main



5. Walk



IV. Reference

1. **Figure#1:** F15LAB03 question provided.
2. **PLL Configuration:** Jonathan Valvano. Embedded Systems (Introduction to Arm\xae Cortex\u2122-M Microcontrollers) (Kindle Locations 6577-6584). Jonathan Valvano.
3. **Table#3:** Jonathan Valvano. Embedded Systems (Introduction to Arm\xae Cortex\u2122-M Microcontrollers) (Kindle Locations 6727-6730). Jonathan Valvano.
4. **Table#4:** NVIC interrupt enable registers. Jonathan Valvano. Embedded Systems (Introduction to Arm\xae Cortex\u2122-M Microcontrollers) (Kindle Location 12845). Jonathan Valvano.
5. **Table#5:** NVIC registers. Jonathan Valvano. Embedded Systems (Introduction to Arm\xae Cortex\u2122-M Microcontrollers) (Kindle Location 12812). Jonathan Valvano.