CECS 346

Final Project

05 - 18 - 2018

Professor: Min He

By

KIAN SOURESRAFIL

UMAR KHAN

In this project we designed an autonomous stepper robot that performs operations with the stepper motors, IR sensor, and power supply circuit. In this project we design a simple smart home garage door that opens or closes based on various inputs (sensors & interrupt controller).

INTRODUCTION

This major objective of this project is learn how to use a stepper motor, learn to use an obstacle avoidance sensor, and learn how to build to build a stepper motor car with the onboard push button controls. An embedded system was built for this project involving state machine. This was converted in to the C language and then transformed onto the board using the uKeil tool. The components used in this project were a TM4C123GXL, microcontroller, an IR sensor, three capacitors which included two 470µF, and a 1µF, one 5V regulator (NTE1951), 9V battery, and two stepper motors, 28BYJ-48.

For the gsarge functionality. An embedded system was built for this project involving state machine. This was converted in to the C language and then transformed onto the board using the uKeil tool. The components used in this project were a TM4C123GXL, microcontroller, an IR sensor, and a stepper motor, 28BYJ-48.

OPERATION

The autonomous stepper robot operates autonomously in the following way:

1. Use the onboard push button SW1 to start the Robot, after it is started, the Robot will move forward 720.
2. Then turn left 90.
3. Move forward again until the obstacle avoidance sensor senses obstacle within 15cm, the Robot car will then stop.
4. When the obstacle is removed, the Robot will keep moving forward 360 and then stop.
5. Use the onboard push button SW2 to start the Robot, this time after it is started, the Robot will move backward 360.
6. Then turn right 90.
7. Moving forward again 720 and stop.

The operation for the garage is as follow.

A simplified smart house is designed with a stepper motor to simulate a garage door, an onboard push button to simulate a doorbell button, two onboard LEDs to simulate three lights in different room, and an obstacle avoidance sensor to detect any obstacle approaching the house. The system initially starts at the green light on with stepper motor pointing downwards until there is an obstacle detected or the button is pressed.

When the obstacle avoidance sensor detects an obstacle moving into a distance range of 15 cm, your embedded system will turn off the green LED and flash red LED with a frequency of 2Hz, and the stepper motor will turn to point upwards. The red LED will keep flashing until the stepper motor point upward. Then the blue LED will be turned on.

When the obstacle avoidance sensor detects an obstacle moving away from a distance range of 15 cm, the blue LED will be turned off, flash red LED until the distance is greater than 15 cm, stepper motor will turn and point back to downwards position. The red LED will be turned off, green LED will be turned on.

When the obstacle avoidance sensor does not detect any obstacle approaching, press one of the onboard push button will toggle garage door open/close. When the garage door is open, turn on the blue LED; when the garage door is moving, flash red LED with a frequency of 2Hz; when the garage door is closed, turn on the green LED.

THEORY

In this project, GPIO port B, E, and F were defined and then initialized. Port E is initialized and the first bit on this port is connected with the IR sensor. Port E handler is used to sense the object. Once an object is detected, the stepper motors stops until the object is removed. Port F is used to initiate the functionality of the switches. SW1 starts the robot and performs the functions for forward operations and operation related to the IR sensor. SW2 performs the backward operations, the turn, and then moves forward. Port B has been initialized for the stepper motors functionality.

SOFTWARE DESIGN

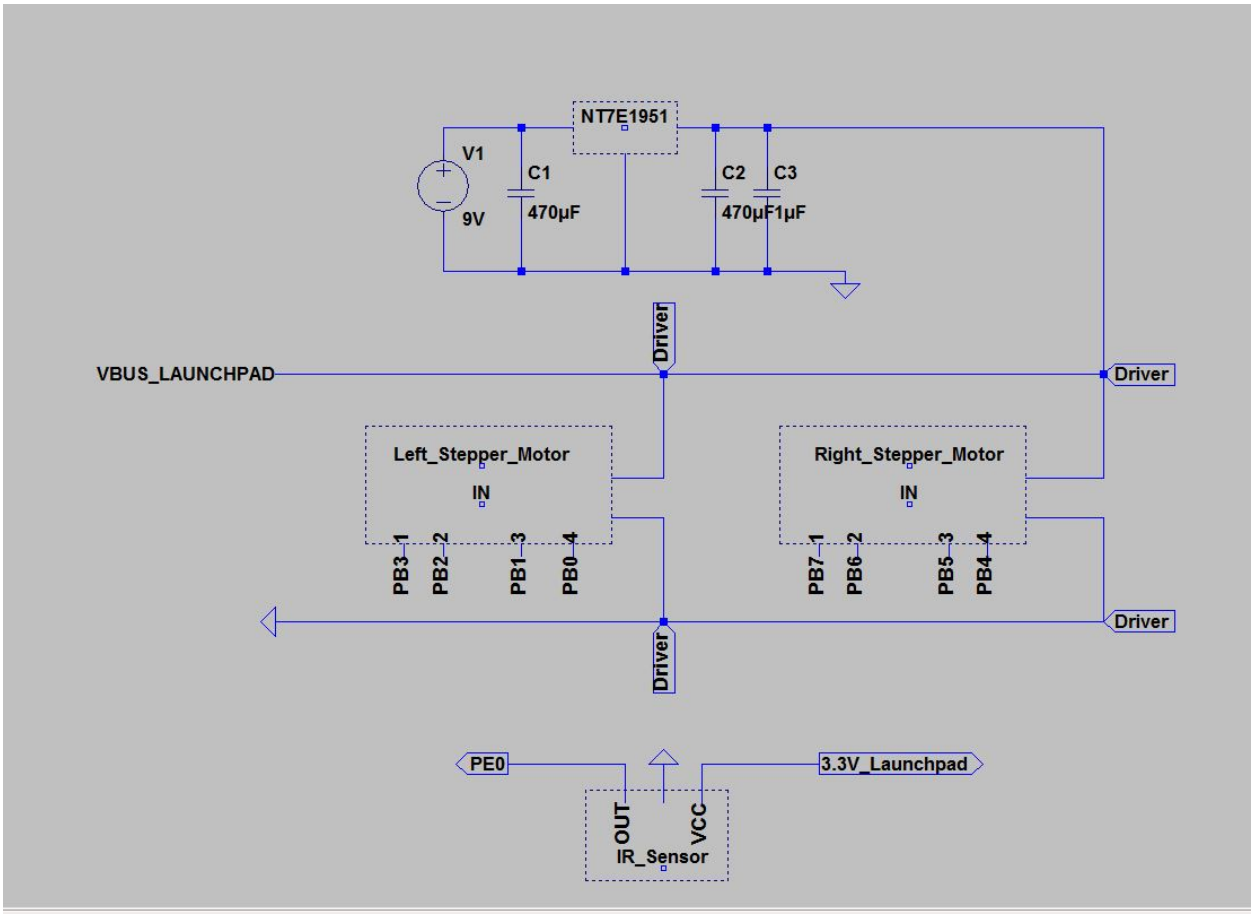
The software was built on the starter project “Project 2” and “Project 3”. The struct was used to determine the states and the ports were defined and then initialized within the main. Once the states were determined on the FSM table, they were just converted to c language to have the logic working on the board.

CONCLUSION

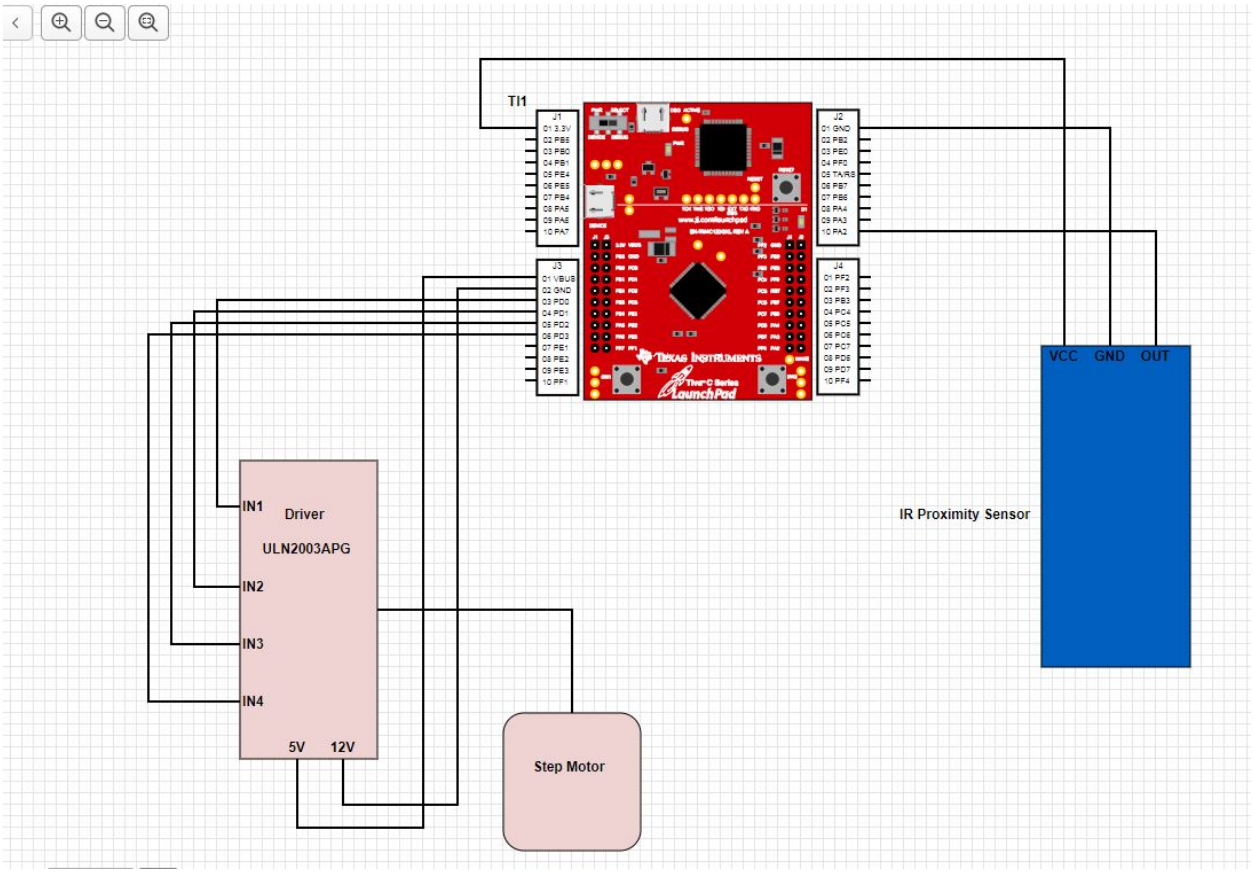
We made the board looking at the schematic that we made. We had some issues in the connections but after troubleshooting, all of them were fixed. The uKeil tool gave us some trouble but was easily fixed by looking in to the debugging documentation. In the end, we were able to complete the project, first tested it on the simulator and then worked on TM4C123 while the connections were made on the breadboard.

SCHEMATIC

Autonomous Car



Smart Home Garage



VIDEO LINK

https://drive.google.com/open?id=1q50v8QIwf25MQ_zcrWD7jAlytY7hg6C7

SOFTWARE CODE

```
//Code for the Garage


// This is your first program to run on the LaunchPad
// You will run this program without modification as your Lab 2
// If the left switch SW1 is
//      not pressed the LED toggles blue-red
//      pressed the LED toggles blue-green


// 0.Documentation Section
// main.c
// Runs on LM4F120 or TM4C123
// Lab2_HelloLaunchPad, Input from PF4, output to PF3,PF2,PF1 (LED)
// Authors: Daniel Valvano, Jonathan Valvano and Ramesh Yerraballi
// Date: January 15, 2016


// LaunchPad built-in hardware
// SW1 left switch is negative logic PF4 on the Launchpad
// SW2 right switch is negative logic PF0 on the Launchpad
// red LED connected to PF1 on the Launchpad
// blue LED connected to PF2 on the Launchpad
// green LED connected to PF3 on the Launchpad


// 1. Pre-processor Directives Section
// Constant declarations to access port registers using
// symbolic names instead of addresses
#define GPIO_PORTA_DATA_R       (*((volatile unsigned long *)0x400043FC))
#define GPIO_PORTA_DIR_R        (*((volatile unsigned long *)0x40004400))
```

```
#define GPIO_PORTA_AFSEL_R      (*((volatile unsigned long *)0x40004420))
#define GPIO_PORTA_PUR_R        (*((volatile unsigned long *)0x40004510))
#define GPIO_PORTA_DEN_R        (*((volatile unsigned long *)0x4000451C))
#define GPIO_PORTA_LOCK_R       (*((volatile unsigned long *)0x40004520))
#define GPIO_PORTA_CR_R         (*((volatile unsigned long *)0x40004524))
#define GPIO_PORTA_AMSEL_R      (*((volatile unsigned long *)0x40004528))
#define GPIO_PORTA_PCTL_R       (*((volatile unsigned long *)0x4000452C))
#define GPIO_PORTA_IS_R         (*((volatile unsigned long *)0x40004404))
#define GPIO_PORTA_IBE_R        (*((volatile unsigned long *)0x40004408))
#define GPIO_PORTA_IEV_R        (*((volatile unsigned long *)0x4000440C))
#define GPIO_PORTA_IM_R         (*((volatile unsigned long *)0x40004410))
#define GPIO_PORTA_RIS_R        (*((volatile unsigned long *)0x40004414))
#define GPIO_PORTA_ICR_R        (*((volatile unsigned long *)0x4000441C))

#define GPIO_PORTD_DATA_BITS_R  ((volatile unsigned long *)0x40007000)
#define GPIO_PORTD_DATA_R       (*((volatile unsigned long *)0x400073FC))
#define GPIO_PORTD_DIR_R        (*((volatile unsigned long *)0x40007400))
#define GPIO_PORTD_IS_R         (*((volatile unsigned long *)0x40007404))
#define GPIO_PORTD_IBE_R        (*((volatile unsigned long *)0x40007408))
#define GPIO_PORTD_IEV_R        (*((volatile unsigned long *)0x4000740C))
#define GPIO_PORTD_IM_R         (*((volatile unsigned long *)0x40007410))
#define GPIO_PORTD_RIS_R        (*((volatile unsigned long *)0x40007414))
#define GPIO_PORTD_MIS_R        (*((volatile unsigned long *)0x40007418))
#define GPIO_PORTD_ICR_R        (*((volatile unsigned long *)0x4000741C))
#define GPIO_PORTD_AFSEL_R      (*((volatile unsigned long *)0x40007420))
#define GPIO_PORTD_DR2R_R       (*((volatile unsigned long *)0x40007500))
#define GPIO_PORTD_DR4R_R       (*((volatile unsigned long *)0x40007504))
#define GPIO_PORTD_DR8R_R       (*((volatile unsigned long *)0x40007508))
#define GPIO_PORTD_ODR_R        (*((volatile unsigned long *)0x4000750C))
#define GPIO_PORTD_PUR_R        (*((volatile unsigned long *)0x40007510))
#define GPIO_PORTD_PDR_R        (*((volatile unsigned long *)0x40007514))
```

```
#define GPIO_PORTD_SLR_R       (*((volatile unsigned long *)0x40007518))
#define GPIO_PORTD_DEN_R       (*((volatile unsigned long *)0x4000751C))
#define GPIO_PORTD_LOCK_R      (*((volatile unsigned long *)0x40007520))
#define GPIO_PORTD_CR_R        (*((volatile unsigned long *)0x40007524))
#define GPIO_PORTD_AMSEL_R     (*((volatile unsigned long *)0x40007528))
#define GPIO_PORTD_PCTL_R      (*((volatile unsigned long *)0x4000752C))
#define GPIO_PORTD_ADCCTL_R    (*((volatile unsigned long *)0x40007530))
#define GPIO_PORTD_DMACTL_R    (*((volatile unsigned long *)0x40007534))

#define GPIO_PORTF_DATA_R      (*((volatile unsigned long *)0x400253FC))
#define GPIO_PORTF_DIR_R       (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_AFSEL_R     (*((volatile unsigned long *)0x40025420))
#define GPIO_PORTF_PUR_R       (*((volatile unsigned long *)0x40025510))
#define GPIO_PORTF_DEN_R       (*((volatile unsigned long *)0x4002551C))
#define GPIO_PORTF_LOCK_R      (*((volatile unsigned long *)0x40025520))
#define GPIO_PORTF_CR_R        (*((volatile unsigned long *)0x40025524))
#define GPIO_PORTF_AMSEL_R     (*((volatile unsigned long *)0x40025528))
#define GPIO_PORTF_PCTL_R      (*((volatile unsigned long *)0x4002552C))

#define GPIO_PORTF_IS_R        (*((volatile unsigned long *)0x40025404))
#define GPIO_PORTF_IBE_R       (*((volatile unsigned long *)0x40025408))
#define GPIO_PORTF_IEV_R       (*((volatile unsigned long *)0x4002540C))
#define GPIO_PORTF_IM_R        (*((volatile unsigned long *)0x40025410))
#define GPIO_PORTF_RIS_R       (*((volatile unsigned long *)0x40025414))
#define GPIO_PORTF_ICR_R       (*((volatile unsigned long *)0x4002541C))
#define NVIC_EN0_R             (*((volatile unsigned long *)0xE000E100))  // IRQ 0 to 31 Set
Enable Register
#define NVIC_PRI7_R            (*((volatile unsigned long *)0xE000E41C))  // IRQ 28 to 31
Priority Register
#define NVIC_PRI0_R            (*((volatile unsigned long *)0xE000E400))  // IRQ 28 to 31
Priority Register
```

```
#define NVIC_PRI3_R             (*((volatile unsigned long *)0xE000E40C))  // IRQ 28 to 31
Priority Register
#define NVIC_SYS_PRI3_R         (*((volatile unsigned long *)0xE000ED20))  // Sys. Handlers 12
to 15 Priority
#define NVIC_ST_CTRL_R          (*((volatile unsigned long *)0xE000E010))
#define NVIC_ST_RELOAD_R        (*((volatile unsigned long *)0xE000E014))
#define NVIC_ST_CURRENT_R       (*((volatile unsigned long *)0xE000E018))
#define SYSCTL_RCGC2_R          (*((volatile unsigned long *)0x400FE108))
```

```
#define NVIC_ST_CTRL_R          (*((volatile unsigned long *)0xE000E010))
#define NVIC_ST_RELOAD_R        (*((volatile unsigned long *)0xE000E014))
#define NVIC_ST_CURRENT_R       (*((volatile unsigned long *)0xE000E018))
#define NVIC_ST_CTRL_COUNT      0x00010000  // Count flag
#define NVIC_ST_CTRL_CLK_SRC    0x00000004  // Clock Source
#define NVIC_ST_CTRL_INTEN      0x00000002  // Interrupt enable
#define NVIC_ST_CTRL_ENABLE     0x00000001  // Counter mode
#define NVIC_ST_RELOAD_M        0x00FFFFFF  // Counter load value

#define SYSCTL_RCGCGPIO_R       (*((volatile unsigned long*)0x400FE608))
#define STEPPER                                                    (*((volatile unsigned
long *)0x4000703C))
#define clockwise 0        // Next index
#define counterclockwise 1 // Next index

struct State{
  unsigned long Out;     // Output
  unsigned long Next[2]; // CW/CCW
};
```

```
typedef const struct State StateType;



StateType fsm[4]={
  {12,{1,3}},
  { 6,{2,0}},
  { 3,{3,1}},
  { 1,{0,2}}
};



// 2. Declarations Section
//   Global Variables
unsigned long mode = 0;
unsigned long lightflg = 1;
unsigned long flag = 0;
unsigned long count, stepcount ;
unsigned char s; // current state
typedef const struct State StateType;

//   Function Prototypes
void PortA_Init(void);
void PortF_Init(void);
void DisableInterrupts(void);
                                                    // Disable interrupts
void EnableInterrupts(void);
                                                    // Enable interrupts
long StartCritical (void);
                                                    // previous I bit, disable interrupts
void EndCritical(long sr);
                                                    // restore I bit to previous value
```

```
void WaitForInterrupt(void);
                                                    // low power mode
void SysTick_Init(unsigned long);
void Stepper_Init(void);



// 3. Subroutines Section
// MAIN: Mandatory for a C Program to be executable
int main(void)
{
       PortA_Init();
  PortF_Init();
  SysTick_Init(30000); //8000000
                      // Call initialization of port PF4 PF2
       Stepper_Init();
  EnableInterrupts();

       // The grader uses interrupts
       GPIO_PORTF_DATA_R = 0x08;  //start at green
       while(1){
       }

}



// Subroutine to initialize port F pins for input and output
// PF4 and PF0 are input SW1 and SW2 respectively
// PF3,PF2,PF1 are outputs to the LED
// Inputs: None
// Outputs: None
// Notes: These five pins are connected to hardware on the LaunchPad
```

```
void PortF_Init(void)
{ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000020;
                              // 1) F clock
  delay = SYSCTL_RCGC2_R;
                              //    delay
  GPIO_PORTF_LOCK_R = 0x4C4F434B;
                              // 2) unlock PortF PF0
  GPIO_PORTF_CR_R |= 0x1E;
                                    //    allow changes to PF4-0 (FRIENDLY SET) GBR
  GPIO_PORTF_AMSEL_R &= ~0x1E;
                              // 3) disable analog function (FRIENDLY Clear)
  GPIO_PORTF_PCTL_R &= ~0x000FFFF0;
                              // 4) GPIO clear bit PCTL  (each pin gets assigned 4 bits)
  GPIO_PORTF_DIR_R |= 0x0E;
                                    // 5) PF4 output, PF3,PF2,PF1 input
  GPIO_PORTF_AFSEL_R &= ~0x1E;
                              // 6) no alternate function
  GPIO_PORTF_PUR_R |= 0x10;
                                    //    enable pullup resistors on PF4
  GPIO_PORTF_DEN_R = 0x1E;
                                    // 7) enable digital pins PF4-PF1



       GPIO_PORTF_IS_R &= ~0x10;
                                                  //(d) PF4 is edge-sensitive
  GPIO_PORTF_IBE_R &= ~0x10;
                                              //    PF4 is NOT both edges
  GPIO_PORTF_IEV_R &= ~0x10;
                                              //    PF4 f edge event
```

```
  GPIO_PORTF_ICR_R = 0x10;
                                          //(e) clear flag4
  GPIO_PORTF_IM_R |= 0x10;
                                          //(f) arm interrupt on PF4
       NVIC_PRI7_R = (NVIC_PRI7_R&0xFF1FFFFF)|0x00000000;
  NVIC_EN0_R = 0x40000000;
                                          //(h) enable interrupt 30 in NVIC
  EnableInterrupts();
}

void GPIOPortF_Handler(void)
{
               GPIO_PORTF_ICR_R = 0x10; //clearing the interrupt
               if(mode == 1) //turning it clockwise
               {
                       mode = 2;
               }

               else
               {
                       mode = 1;
               }



}



void PortA_Init(void)
{ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000001;
                             // 1) A clock
```

```
delay = SYSCTL_RCGC2_R;
                            //    delay
//GPIO_PORTA_LOCK_R = 0x4C4F434B;
                            // 2) unlock PortF PF0
GPIO_PORTA_CR_R |= 0x04;
                                    //    allow changes to PF4-0 (FRIENDLY SET)
GPIO_PORTA_AMSEL_R &= ~0x04;
                            // 3) disable analog function (FRIENDLY Clear)
GPIO_PORTA_PCTL_R &= ~0xFFFFFFFF;
                            // 4) GPIO clear bit PCTL  (each pin gets assigned 4 bits)
GPIO_PORTA_DIR_R &= ~0x04;
                            // 5) PF4 input, PF3,PF2,PF1 output
GPIO_PORTA_AFSEL_R &= ~0x04;
                            // 6) no alternate function
GPIO_PORTA_PUR_R |= 0x04;
                                    //    enable pullup resistors on PF4
GPIO_PORTA_DEN_R |= 0x04;
                            // 7) enable digital pins PF4-PF1


     GPIO_PORTA_IS_R &= ~0x04;
                                                //(d) PF4 is edge-sensitive
GPIO_PORTA_IBE_R |= 0x04;
                                                //    PF4 is both edges
//GPIO_PORTA_IEV_R &= ~0x04;
                                    //    PF4 f edge event
GPIO_PORTA_ICR_R = 0x04;
                                         //(e) clear flag4
GPIO_PORTA_IM_R |= 0x04;
                                         //(f) arm interrupt on PF4
     NVIC_PRI0_R = (NVIC_PRI0_R&0xFFFFFF1F)|0x00000020;
```

```
  NVIC_EN0_R = 0x00000001;
                                          //(h) enable interrupt 30 in NVIC
  EnableInterrupts();
}

void Stepper_Init(void)
{
  SYSCTL_RCGCGPIO_R |= 0x08; // 1) activate port D
  s = 0;
                                    // 2) no need to unlock PD3-0
  GPIO_PORTD_AMSEL_R &= ~0x0F;      // 3) disable analog functionality on PD3-0
  GPIO_PORTD_PCTL_R &= ~0x0000FFFF; // 4) GPIO configure PD3-0 as GPIO
  GPIO_PORTD_DIR_R |= 0x0F;                         // 5) make PD3-0 out
  GPIO_PORTD_AFSEL_R &= ~0x0F;                      // 6) disable alt funct on PD3-0
  GPIO_PORTD_DR8R_R |= 0x0F;                        // enable 8 mA drive
  GPIO_PORTD_DEN_R |= 0x0F;                         //

}
//sensor handler
void GPIOPortA_Handler(void)
{
  GPIO_PORTA_ICR_R = 0x04;

        //if it detects object, it will turn counter clockwise else clockwise
        if(GPIO_PORTA_DATA_R&0x04)
        {
                mode = 1;
        }
        while (flag == 1){
        if(GPIO_PORTA_DATA_R&0x04)
```

```
	{
		mode = 2;
	}	}

}

// Initialize SysTick with busy wait running at bus clock.
void SysTick_Init(unsigned long period){
  NVIC_ST_CTRL_R = 0;                   // disable SysTick during setup
  NVIC_ST_RELOAD_R = period - 1;  // maximum reload value
  NVIC_ST_CURRENT_R = 0;                // any write to current clears it
  NVIC_PRI3_R = (NVIC_PRI3_R&0x00FFFFFF)|0x40000000; // enable SysTick with core clock
  NVIC_ST_CTRL_R = 0x07;
}
// Time delay using busy wait.
// The delay parameter is in units of the core clock. (units of 20 nsec for 50 MHz clock)
void SysTick_Handler(void)
{
			//s = fsm[s].Next[clockwise]; // clock wise circular
			//STEPPER = fsm[s].Out; // step motor
	if(mode== 1)
	{
		count+=1; //for the lights

		if(count > 50)
		{
			GPIO_PORTF_DATA_R = GPIO_PORTF_DATA_R^0x02; //blinking the red light
			GPIO_PORTF_DATA_R &=~0x0D; //clear the bits for only red lights
			count = 0;
		}
		//rotation purposes
```

```
        if(stepcount < 4750)
        {
                s = fsm[s].Next[clockwise]; // clock wise circular
                STEPPER = fsm[s].Out; // step motor
                stepcount+=1;
        }
        if (stepcount >= 4750)
        {
                count = 0;
                GPIO_PORTF_DATA_R = 0x04; //turning blue when it stops
                flag = 1;
        }



        }

if (mode == 2)
{
        count+=1;
        if(count > 50)
        {
                //GPIO_PORTF_DATA_R = 0x00;
                GPIO_PORTF_DATA_R = GPIO_PORTF_DATA_R^0x02;
                GPIO_PORTF_DATA_R &=~0x0D; // Clear the bits so red only shows
                count = 0;
        }

        if(stepcount > 0)
        {
                s = fsm[s].Next[counterclockwise]; // clock wise circular
                STEPPER = fsm[s].Out; // step motor
```

```
                        stepcount -=1; //decrementing to 0
                }
                if (stepcount == 0)
                {
                        count = 0;
                        GPIO_PORTF_DATA_R = 0x08; //turning green when it comes to 0

                }
        }
}
```

```
// Color    LED(s) PortF
// dark     ---    0
// red      R--    0x02
// blue     --B    0x04
// green    -G-    0x08
// yellow   RG-    0x0A
// sky blue -GB    0x0C
// white    RGB    0x0E
// pink     R-B    0x06
```

```
//Code for the Stepper Robot


//Umar Khan
//Kian Souresfail
//CECS 346
//Project 3 main.c file - An Autonomous Stepper Robot
// Mode of operation:
// When the left switch SW1 is pressed the Robot follows the following path
//     1, Robot moves 720 degree forward
//     2, Turn left 90 degree
//     3, Move forward again until the obstacle avoidance sensor senses obstacle
//        within 15cm, the Robot car will then stop.
//     4, When the obstacle is removed, the Robot will keep moving forward 360? and then stop.
// When the Right switch  SW2 is pressed the Robot follows the following path
//     1, The Robot will move backward 360 degree.
//     2, Then turn right 90 degree.
//     3, Then moves forward again 720 degrees and stop.
//
// LaunchPad built-in hardware
// SW1 left switch is negative logic PF4 on the Launchpad
// SW2 right switch is negative logic PF0 on the Launchpad
// red LED connected to PF1 on the Launchpad
// blue LED connected to PF2 on the Launchpad
// green LED connected to PF3 on the Launchpad

// PD3 connected to driver for stepper motor coil A
// PD2 connected to driver for stepper motor coil A'
// PD1 connected to driver for stepper motor coil B
// PD0 connected to driver for stepper motor coil B'

#include <stdint.h>
```

```
#include "stepper.h"
#include "systick.h"
#define T1ms 16000    // assumes using 16 MHz PIOSC (default setting for clock source)
#define NVIC_EN0_R              (*((volatile unsigned long *)0xE000E100))  // IRQ 0 to 31 Set
Enable Register
#define NVIC_PRI7_R             (*((volatile unsigned long *)0xE000E41C))  // IRQ 28 to 31
Priority Register
#define GPIO_PORTF_DATA_R       (*((volatile unsigned long *)0x400253FC))
#define GPIO_PORTF_DIR_R        (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_AFSEL_R      (*((volatile unsigned long *)0x40025420))
#define GPIO_PORTF_PUR_R        (*((volatile unsigned long *)0x40025510))
#define GPIO_PORTF_DEN_R        (*((volatile unsigned long *)0x4002551C))
#define GPIO_PORTF_LOCK_R       (*((volatile unsigned long *)0x40025520))
#define GPIO_PORTF_CR_R         (*((volatile unsigned long *)0x40025524))
#define GPIO_PORTF_AMSEL_R      (*((volatile unsigned long *)0x40025528))
#define GPIO_PORTF_PCTL_R       (*((volatile unsigned long *)0x4002552C))
#define GPIO_PORTF_IS_R         (*((volatile unsigned long *)0x40025404))
#define GPIO_PORTF_IBE_R        (*((volatile unsigned long *)0x40025408))
#define GPIO_PORTF_IEV_R        (*((volatile unsigned long *)0x4002540C))
#define GPIO_PORTF_IM_R         (*((volatile unsigned long *)0x40025410))
#define GPIO_PORTF_ICR_R        (*((volatile unsigned long *)0x4002541C))
#define GPIO_PORTF_RIS_R        (*((volatile unsigned long *)0x40025414))
#define SYSCTL_RCGC2_R          (*((volatile unsigned long *)0x400FE108))

#define GPIO_PORTE_DATA_R       (*((volatile unsigned long *)0x400243FC))
#define GPIO_PORTE_DIR_R        (*((volatile unsigned long *)0x40024400))
#define GPIO_PORTE_AFSEL_R      (*((volatile unsigned long *)0x40024420))
#define GPIO_PORTE_PUR_R        (*((volatile unsigned long *)0x40024510))
#define GPIO_PORTE_DEN_R        (*((volatile unsigned long *)0x4002451C))
#define GPIO_PORTE_CR_R         (*((volatile unsigned long *)0x40024524))
#define GPIO_PORTE_AMSEL_R      (*((volatile unsigned long *)0x40024528))
```

```
#define GPIO_PORTE_PCTL_R       (*((volatile unsigned long *)0x4002452C))
#define GPIO_PORTE_IS_R         (*((volatile unsigned long *)0x40024404))
#define GPIO_PORTE_IBE_R        (*((volatile unsigned long *)0x40024408))
#define GPIO_PORTE_IEV_R        (*((volatile unsigned long *)0x4002440C))
#define GPIO_PORTE_IM_R         (*((volatile unsigned long *)0x40024410))
#define GPIO_PORTE_RIS_R        (*((volatile unsigned long *)0x40024414))
#define GPIO_PORTE_ICR_R        (*((volatile unsigned long *)0x4002441C))
#define GPIO_PORTE_MIS_R        (*((volatile unsigned long *)0x40024418))
#define NVIC_PRI1_R             (*((volatile unsigned long *)0xE000E400))

#define NVIC_SYS_PRI3_R         (*((volatile unsigned long *)0xE000ED20))  // Sys. Handlers 12
to 15 Priority
#define NVIC_ST_CTRL_R          (*((volatile unsigned long *)0xE000E010))
#define NVIC_ST_RELOAD_R        (*((volatile unsigned long *)0xE000E014))
#define NVIC_ST_CURRENT_R       (*((volatile unsigned long *)0xE000E018))

#define RED  0x02;
#define GREEN 0x08;
#define BLUE 0x04;
#define T1ms 16000



//   Function Prototypes
void EnableInterrupts(void);       // Enable interrupts
void Init_PortF(void);             // Initialize Port F for the Switches Control
void Init_PortE(void);                      // Initializing Port E for the sensors
functionality
void GPIOPortE_Handler(void);               // Port E handler for object detection
void GPIOPortF_Handler(void);               // Port F handler for Switches functionality
// global variable visible in Watch window of debugger
volatile unsigned long speed = 2; //to increase or decrease the speed of the motors(tires)
```

```
volatile unsigned long SW1;
volatile unsigned long SW2;
volatile unsigned long sensor;
volatile unsigned long object = 1;
unsigned int i;
unsigned int j;
unsigned int flag=1;



// MAIN: Mandatory for a C Program to be executable
int main(void){
  //calling the functions
        Init_PortF();
  Init_PortE();
        Stepper_Init();
  EnableInterrupts();

 GPIO_PORTF_DATA_R = GREEN;
  while(1){
              //SW1 functionality
              if ((SW1)){
          for (i=0;i<4000; i++) {
                     moveForward(speed*T1ms);
                                    }
                              //testing the exit of the loop
                                    GPIO_PORTF_DATA_R = RED;
                                     for (i=0;i<2000; i++) {
                    turnLeftForward(speed*T1ms);
                              }
                              //testing the exit of the loop
                              GPIO_PORTF_DATA_R = RED;
```

```
  while(flag ==1){
                                if (object == 1){
                                 moveForward(speed*T1ms);
                                }
                                 else {
                                        //test for the object detections
                                        GPIO_PORTF_DATA_R = BLUE;
                                        flag = 0;
                                        SysTick_Wait(100000);
                                        SysTick_Wait(100000);
                                        SysTick_Wait(100000);
                                        while(!object){
                                               //flag = 0;


                                               //flag = 1;
                                        }
                                        flag = 0;
                                 }
  }
                                        //moving the robot forward again
                                        GPIO_PORTF_DATA_R = GREEN;


                                       for (i=0;i<3700; i++) {


                                              moveForward(6*T1ms);
                                       }
                                GPIO_PORTF_DATA_R = RED;
                                SW1 = 0;
                        }


else if(SW2){
```

```
                //test for SW2 start
                GPIO_PORTF_DATA_R = RED;
                for (i=0;i<4000; i++) {
                        moveBackward(speed*T1ms);
                }
                //testing the exit of the loop
                GPIO_PORTF_DATA_R = BLUE;
                for (i=0;i<2000; i++) {
                      turnRightForward(speed*T1ms);
        }
                //testing the exit of the loop
                GPIO_PORTF_DATA_R = RED;
                for (i=0;i<4000; i++) {
                        moveForward(speed*T1ms);
                }
    SW2 =0;
  }
 }
}

//ISR handler for port F
void GPIOPortF_Handler(void){
 if (GPIO_PORTF_RIS_R&0x10){    //SW1 pressed
  GPIO_PORTF_ICR_R = 0x10;      // acknowledge flag4
  SW1 = 1;                      // set flag
 }
  else if (GPIO_PORTF_RIS_R&0x01){ // SW2 pressed
  GPIO_PORTF_ICR_R = 0x01;      // acknowledge flag0
   SW2 = 1;                     // set flag
  }
}
```

```
//ISR handler for port E
void GPIOPortE_Handler(void){
 GPIO_PORTE_ICR_R = 0x01;      // acknowledge flag
 sensor= GPIO_PORTE_DATA_R;
        if (sensor == 0x00){
          object = 0;
        }
   if (sensor == 0x01)
                        object = 1;
        }



// Subroutine to initialize port F pins for input and output
void Init_PortF(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000020;     // 1) F clock
  delay = SYSCTL_RCGC2_R;           // delay
  GPIO_PORTF_LOCK_R = 0x4C4F434B;   // 2) unlock PortF PF0
  GPIO_PORTF_CR_R = 0x1F;           // allow changes to PF4-0
  GPIO_PORTF_DEN_R = 0x1F;          // 7) enable digital pins PF4-PF0
  GPIO_PORTF_AMSEL_R = 0x00;        // 3) disable analog function
  GPIO_PORTF_PCTL_R = 0x00000000;   // 4) GPIO clear bit PCTL
  GPIO_PORTF_DIR_R = 0x0E;          // 5) PF4,PF0 input, PF3,PF2,PF1 output
  GPIO_PORTF_AFSEL_R = 0x00;        // 6) no alternate function
  GPIO_PORTF_PUR_R |= 0x11;         // enable pullup resistors on PF0, PF4
  GPIO_PORTF_IS_R &= ~0x11;         // PF4 is edge-sensitive
  GPIO_PORTF_IBE_R &= ~0x11;        // PF4 is not both edges
  GPIO_PORTF_IEV_R &= ~0x11;                //     PF4 falling edge event
  GPIO_PORTF_ICR_R = 0x11;                  // (e) clear flag4,flag0
  GPIO_PORTF_IM_R |= 0x11;                    // (f) arm interrupt on PF4,PF0
```

```
  NVIC_PRI7_R = (NVIC_PRI7_R&0xFF00FFFF)|0x00A00000; // (g) priority 5
  NVIC_EN0_R = 0x40000000;                  // (h) enable interrupt 30 in NVIC

}
// Subroutine to initialize port E pins for input and output
void Init_PortE(){volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000010;       // 1)  E clock
  delay = SYSCTL_RCGC2_R;             // delay
  GPIO_PORTE_CR_R |= 0x01;            // allow changes to PE0
  GPIO_PORTE_AMSEL_R = 0x00;          // 3) disable analog function
  GPIO_PORTE_PCTL_R = 0x00000000;     // 4) GPIO clear bit PCTL
  GPIO_PORTE_DIR_R &= ~0x01;          // 5) PE0 input
  GPIO_PORTE_AFSEL_R = 0x00;          // 6) no alternate function
  GPIO_PORTE_PUR_R &= ~0x01;          // disable pullup resistor PE0
  GPIO_PORTE_DEN_R |= 0x01;           // 7) enable digital PE0
  GPIO_PORTE_IS_R &= ~0x01;         // PE0 is edge-sensitive
  GPIO_PORTE_IBE_R |= 0x01;        // PE0 is both edges
  GPIO_PORTE_ICR_R = 0x01;      // (e) clear flag0
  GPIO_PORTE_IM_R |= 0x01;      // (f) arm interrupt on PE0
  NVIC_PRI1_R = (NVIC_PRI1_R&0xFFFFFF0F)|0x00000080; // (g) priority 4
  NVIC_EN0_R = 0x00000010;      // (h) enable interrupt 4 in NVIC
}

// Stepper.c
// Runs on LM4F120/TM4C123
// Provide functions that step the motor once clockwise, step
// once counterclockwise, and initialize the stepper motor
// interface.
// Daniel Valvano
// September 12, 2013
// Modified by Dr. Min He April 28, 2017
```

```
/* This example accompanies the book
   "Embedded Systems: Real Time Interfacing to ARM Cortex M Microcontrollers",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2015
   Example 4.1, Programs 4.4, 4.5, and 4.6
   Hardware circuit diagram Figure 4.27

 Copyright 2015 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
    as long as the above copyright notice remains
 THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
 OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
 VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
 OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/
 */

// PD3 connected to driver for stepper motor coil A
// PD2 connected to driver for stepper motor coil A'
// PD1 connected to driver for stepper motor coil B
// PD0 connected to driver for stepper motor coil B'

#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "systick.h"
struct State{
  uint8_t Out;     // Output
  uint8_t Next[2]; // CW/CCW
};
```

```
typedef const struct State StateType;


#define clockwise 0        // Next index
#define counterclockwise 1 // Next index
StateType fsm[4]={
  {12,{1,3}},
  { 6,{2,0}},
  { 3,{3,1}},
  { 1,{0,2}}
};


StateType moveBoth_fsm[4]={
   {0x1C,{1,3}},
   {0x36,{2,0}},
   {0x63,{3,1}},
   {0xC1,{0,2}}
};
StateType turnLeft_fsm[4]={
  {0x0C,{1,3}},
  {0x06,{2,0}},
  {0x03,{3,1}},
  {0x01,{0,2}}
};
StateType turnRight_fsm[4]={
  {0x10,{1,3}},
  {0x30,{2,0}},
  {0x60,{3,1}},
  {0xC0,{0,2}}
};


unsigned char s; // current state
```

```
#define STEPPER  (*((volatile uint32_t *)0x400053FC))

// Move 1.8 degrees clockwise, delay is the time to wait after each step
void Stepper_CW(uint32_t delay){
  s = fsm[s].Next[clockwise]; // clock wise circular
  STEPPER = fsm[s].Out; // step motor
  SysTick_Wait(delay);
}
// Move 1.8 degrees counterclockwise, delay is wait after each step
void Stepper_CCW(uint32_t delay){
  s = fsm[s].Next[counterclockwise]; // counter clock wise circular
  STEPPER = fsm[s].Out; // step motor
  SysTick_Wait(delay); // blind-cycle wait
}
void moveForward(uint32_t delay){
  s = moveBoth_fsm[s].Next[clockwise]; // clock wise circular
  STEPPER = moveBoth_fsm[s].Out; // step motor
  SysTick_Wait(delay);
        //SysTick_Wait(delay);
}

void moveWithDelayForward(uint32_t delay){
  s = moveBoth_fsm[s].Next[clockwise]; // clock wise circular
  STEPPER = moveBoth_fsm[s].Out; // step motor
  SysTick_Wait(5*delay);
}
void moveBackward(uint32_t delay){
  s = moveBoth_fsm[s].Next[counterclockwise]; // clock wise circular
  STEPPER = moveBoth_fsm[s].Out; // step motor
  SysTick_Wait(delay);
```

```
}

void turnRightForward(uint32_t delay){
  s = turnRight_fsm[s].Next[clockwise]; // clock wise circular
  STEPPER =  turnRight_fsm[s].Out; // step motor
  SysTick_Wait(delay);
}
void turnRightReverse(uint32_t delay){
  s = turnRight_fsm[s].Next[counterclockwise]; // clock wise circular
  STEPPER =  turnRight_fsm[s].Out; // step motor
  SysTick_Wait(delay);
}
void turnLeftForward(uint32_t delay){
  s = turnLeft_fsm[s].Next[clockwise]; // clock wise circular
  STEPPER = turnLeft_fsm[s].Out; // step motor
  SysTick_Wait(delay);
}
void turnLeftReverse(uint32_t delay){
  s = turnLeft_fsm[s].Next[counterclockwise]; // clock wise circular
  STEPPER = turnLeft_fsm[s].Out; // step motor
 SysTick_Wait(delay);
}



// Initialize Stepper interface
void Stepper_Init(void){
  SYSCTL_RCGCGPIO_R |= 0x02; // 1) activate port B
  SysTick_Init();
  s = 0;
                                    // 2) no need to unlock PB7-0
  GPIO_PORTB_AMSEL_R &= ~0xFF;      // 3) disable analog functionality on PB7-0
```

```
  GPIO_PORTB_PCTL_R &= ~0xFFFFFFFF; // 4) GPIO configure PB7-0 as GPIO
  GPIO_PORTB_DIR_R |= 0xFF;   // 5) make PD3-0 out
  GPIO_PORTB_AFSEL_R &= ~0xFF;// 6) disable alt funct on PB7-0
  GPIO_PORTB_DR8R_R |= 0xFF;  // enable 8 mA drive
  GPIO_PORTB_DEN_R |= 0xFF;   // 7) enable digital I/O on PB7-0
}

// SysTick.c
// Runs on LM4F120/TM4C123
// Provide functions that initialize the SysTick module, wait at least a
// designated number of clock cycles, and wait approximately a multiple
// of 10 milliseconds using busy wait.  After a power-on-reset, the
// LM4F120 gets its clock from the 16 MHz precision internal oscillator,
// which can vary by +/- 1% at room temperature and +/- 3% across all
// temperature ranges.  If you are using this module, you may need more
// precise timing, so it is assumed that you are using the PLL to set
// the system clock to 50 MHz.  This matters for the function
// SysTick_Wait10ms(), which will wait longer than 10 ms if the clock is
// slower.
// Daniel Valvano
// September 11, 2013

/* This example accompanies the book
   "Embedded Systems: Real Time Interfacing to Arm Cortex M Microcontrollers",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2015
   Program 2.11, Section 2.6

 Copyright 2015 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
    as long as the above copyright notice remains
 THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
```

```
 OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
 VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
 OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/
 */

#include <stdint.h>
#include "tm4c123gh6pm.h"
#define NVIC_ST_CTRL_COUNT      0x00010000  // Count flag
#define NVIC_ST_CTRL_CLK_SRC    0x00000004  // Clock Source
#define NVIC_ST_CTRL_INTEN      0x00000002  // Interrupt enable
#define NVIC_ST_CTRL_ENABLE     0x00000001  // Counter mode
#define NVIC_ST_RELOAD_M        0x00FFFFFF  // Counter load value

// Initialize SysTick with busy wait running at bus clock.
void SysTick_Init(void){
  NVIC_ST_CTRL_R = 0;                   // disable SysTick during setup
  NVIC_ST_RELOAD_R = NVIC_ST_RELOAD_M;  // maximum reload value
  NVIC_ST_CURRENT_R = 0;                // any write to current clears it
                                        // enable SysTick with core clock
  NVIC_ST_CTRL_R = NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC;
}
// Time delay using busy wait.
// The delay parameter is in units of the core clock. (units of 20 nsec for 50 MHz clock)
void SysTick_Wait(uint32_t delay){
  volatile uint32_t elapsedTime;
  uint32_t startTime = NVIC_ST_CURRENT_R;
  do{
    elapsedTime = (startTime-NVIC_ST_CURRENT_R)&0x00FFFFFF;
```

```
  }
  while(elapsedTime <= delay);
}
// Time delay using busy wait.
// This assumes 50 MHz system clock.
void SysTick_Wait10ms(uint32_t delay){
  uint32_t i;
  for(i=0; i<delay; i++){
    SysTick_Wait(500000);  // wait 10ms (assumes 50 MHz clock)
  }
}
```