8.1 Appendix - Bill of Materials

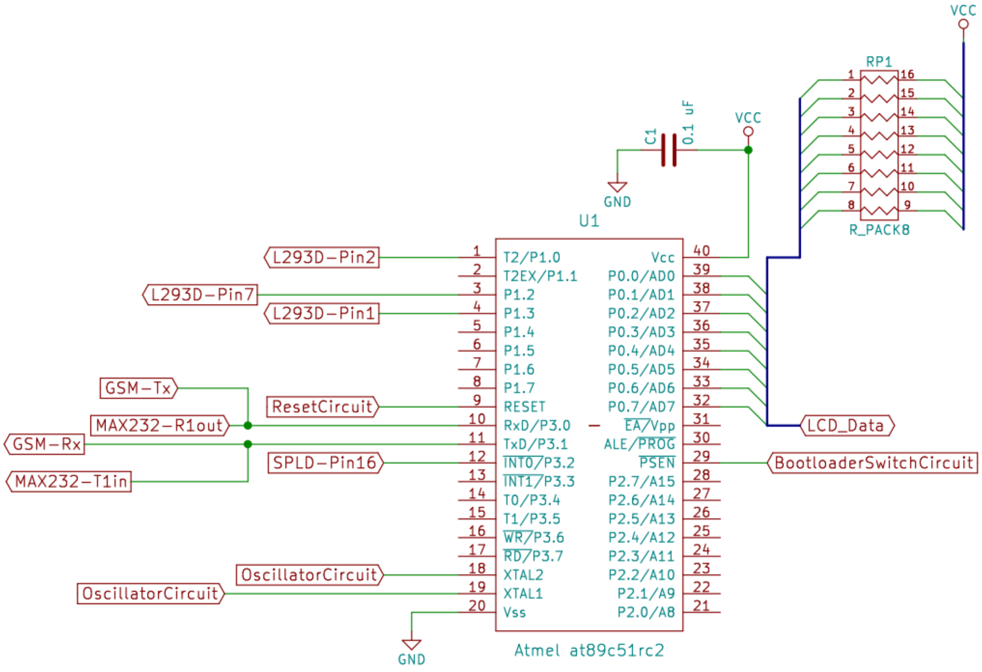
Part Description	**Source**	**Cost**
GSM module, Adafruit Sim808	Amazon https://www.amazon.com	$59.99
Prime Shipping		$8.99
GSM Quad band antenna	Amazon https://www.amazon.com	$8.95
Lithium Ion Battery	Amazon https://www.amazon.com	$14.53
USB to TTL serial cable	Amazon https://www.amazon.com	$13.95
USB to serial converter	Amazon https://www.amazon.com/	$9.99
ADC0808	TI website: http://www.ti.com/	$.5.30
Shipping		$6.99
L293D motor driver	Amazon https://www.amazon.com	$2.95
Shipping		$8.00
Accelerometer ADXL335	Borrowed from fellow student	$0.00
Flex sensor	Borrowed from BTU lab	$0.00
Jumper wires	Borrowed from ITLL	$0.00
DC motor	Amazon https://www.amazon.com	12.76
Adafruit PIR motion sensor	Amazon https://www.amazon.com	$10.97
TOTAL		**$158.07**

Notes:

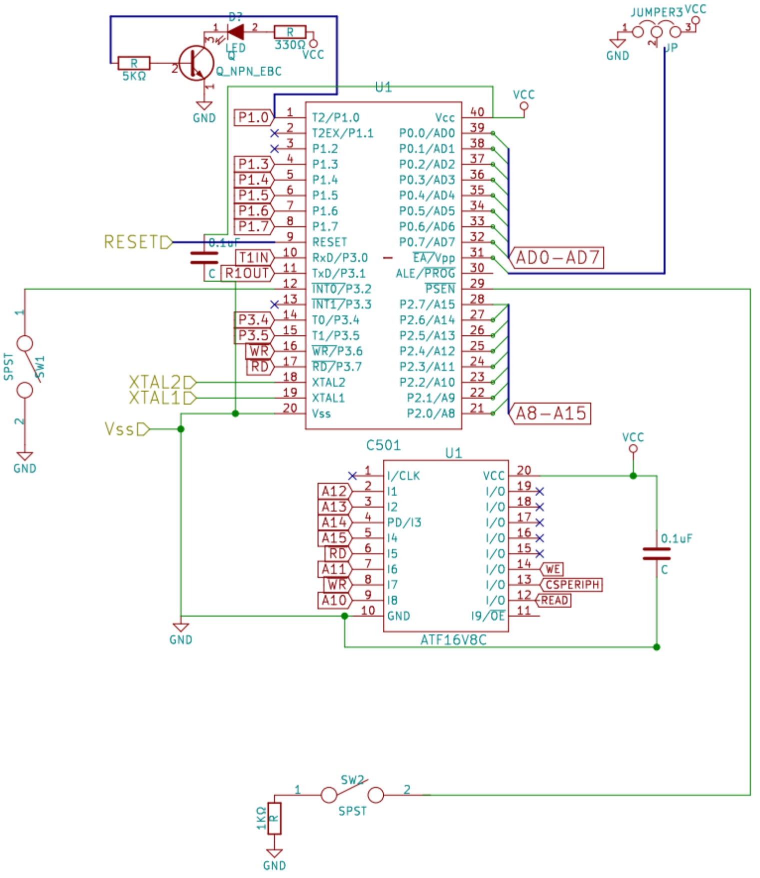
1) We had initially chosen the Sim300 GSM module. But we were not aware that the same module comes in different breakout boards. The one that we ordered from amazon:
https://www.amazon.com/gp/product/B005FXDX74/ref=od_aui_detailpages04?ie=UTF8&psc=1
This did not have any data sheet or pinout.
We had to return the module, as it did not have any documentation.
Lesson learnt: With Sim300 at the core, there are different development boards available. Need to check for relevant documentation before ordering parts.
2) Adafruit Sim808 was available at a cheaper price directly from Adafruit website, but we had to purchase it from amazon because we needed it shipped at the earliest during the Thanksgiving break.
3) We would have borrowed DC motors from ITLL, if we had known earlier that we could do that.

8.2 Appendix – Schematics

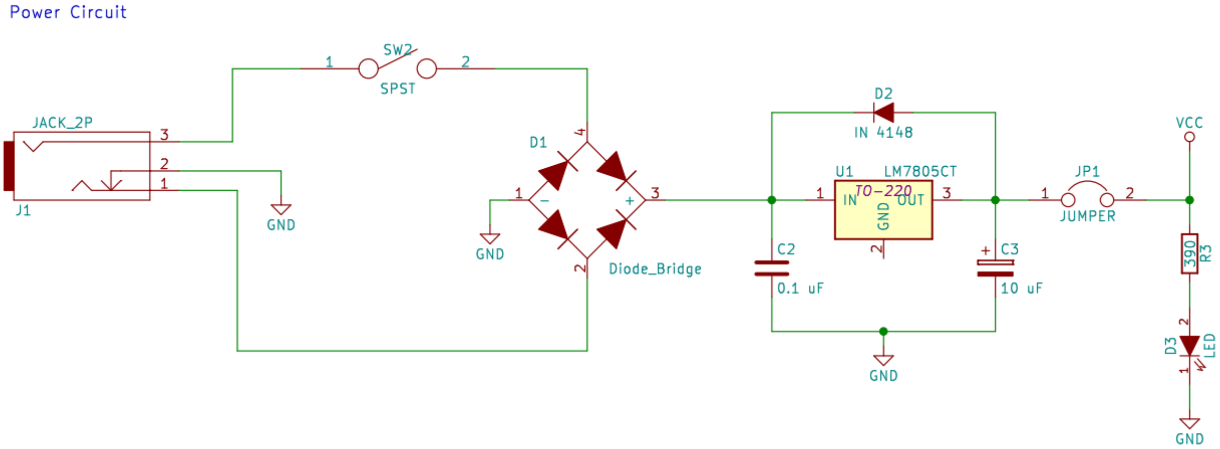
Connections to the microcontroller: Board 1



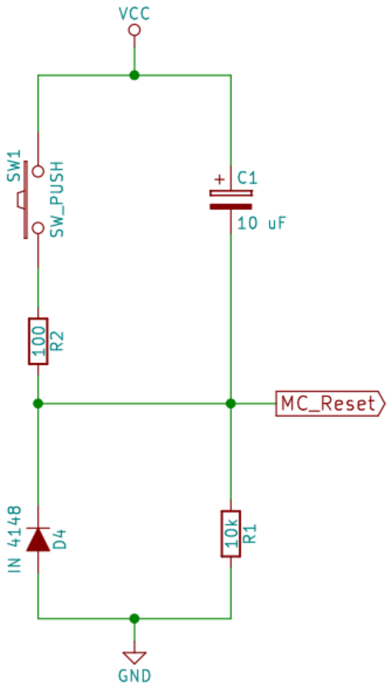
Board 2:



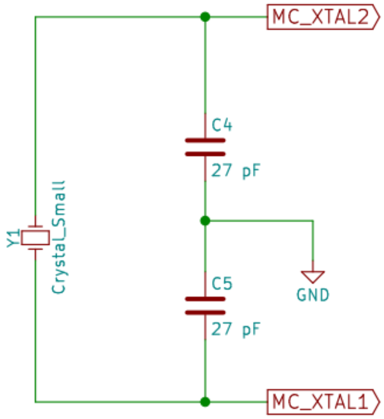
D?
LED
R
330Ω
VCC
R
5KΩ
Q
Q_NPN_EBC
GND
JUMPER3
VCC
GND
JP
U1
VCC
P1.0
P1.3
P1.4
P1.5
P1.6
P1.7
RESET
0.1uF
C
T1IN
R1OUT
T2/P1.0
T2EX/P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
RESET
RxD/P3.0
TxD/P3.1
INT0/P3.2
INT1/P3.3
T0/P3.4
T1/P3.5
WR/P3.6
RD/P3.7
XTAL2
XTAL1
Vss
Vcc
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
EA/Vpp
ALE/PROG
PSEN
P2.7/A15
P2.6/A14
P2.5/A13
P2.4/A12
P2.3/A11
P2.2/A10
P2.1/A9
P2.0/A8
AD0–AD7
A8–A15
P3.4
P3.5
WR
RD
XTAL2
XTAL1
Vss
SPST
SW1
GND
C501
U1
VCC
A12
A13
A14
A15
RD
A11
WR
A10
I/CLK
I1
I2
PD/I3
I4
I5
I6
I7
I8
GND
VCC
I/O
I9/OE
WE
CSPERIPH
READ
0.1uF
C
ATF16V8C
GND
SW2
SPST
1KΩ
R
GND



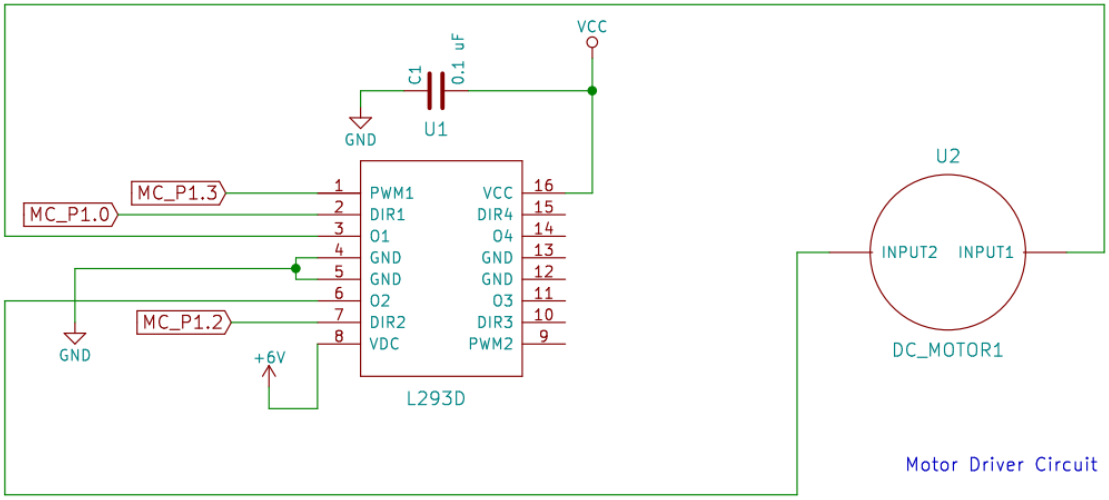
Power Circuit
SW2
SPST
JACK_2P
J1
GND
D1
GND
Diode_Bridge
D2
IN 4148
U1
LM7805CT
TO-220
IN
GND
OUT
C2
0.1 uF
C3
10 uF
GND
JP1
JUMPER
VCC
390
R3
D3
LED
GND



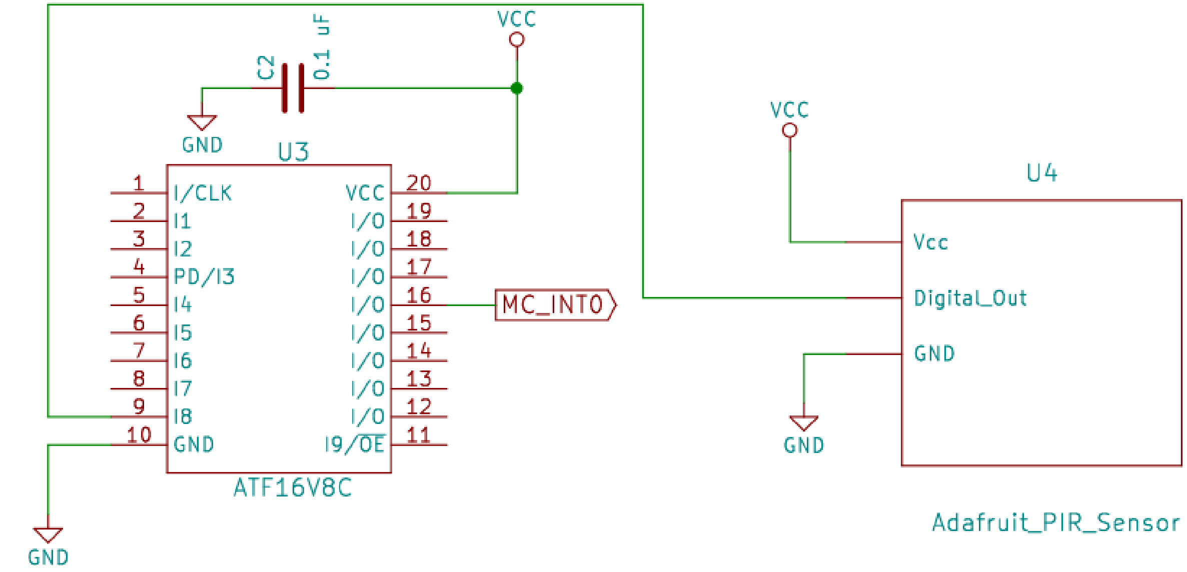
Reset Circuit



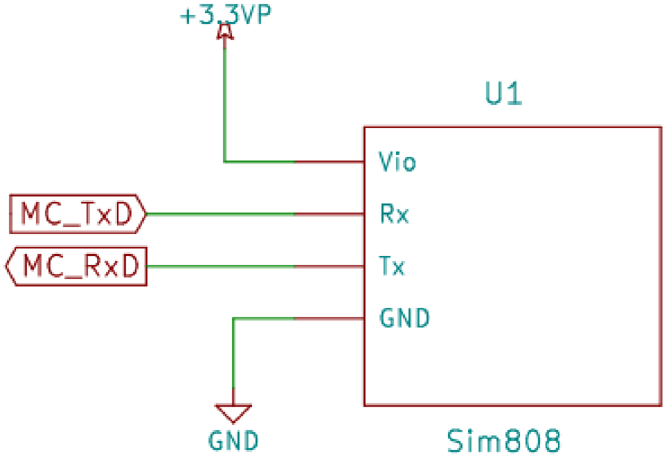
Oscillator Circuit



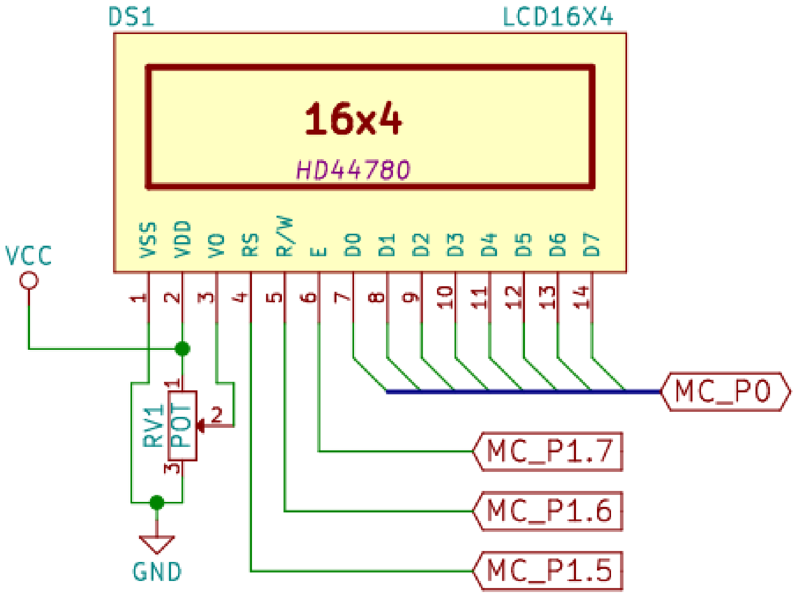
Motor Driver Circuit



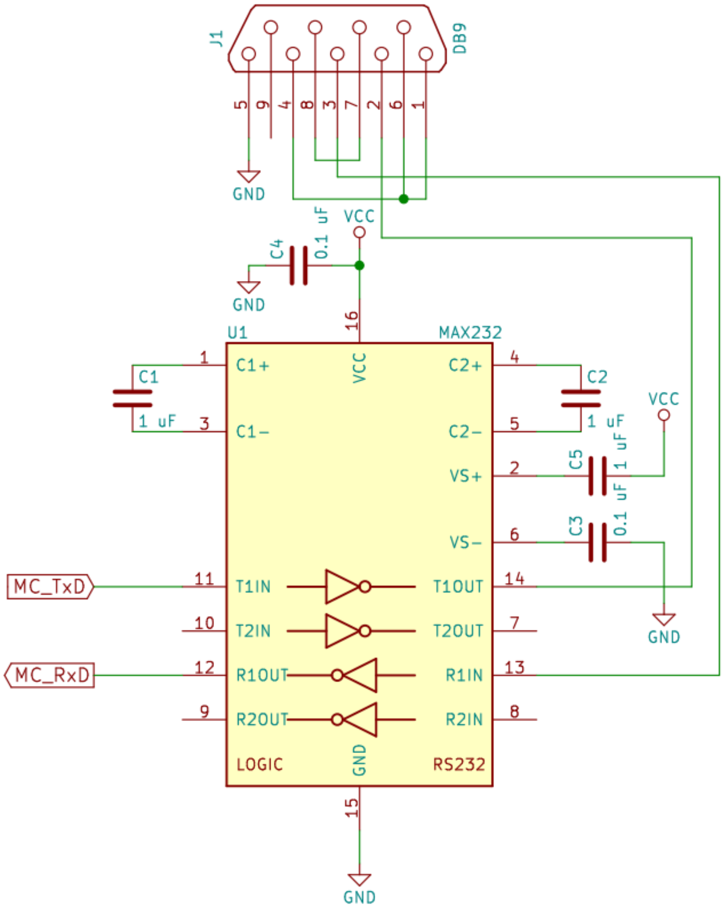
PIR Sensor Intefacing Circuit



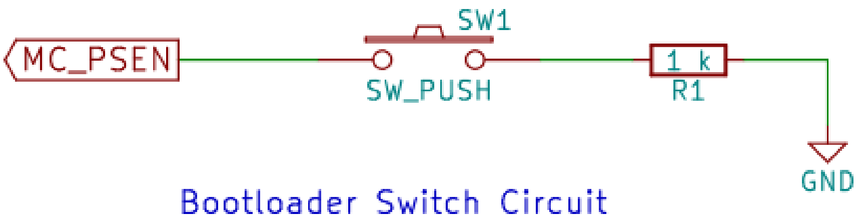
GSM Interfacing Circuit



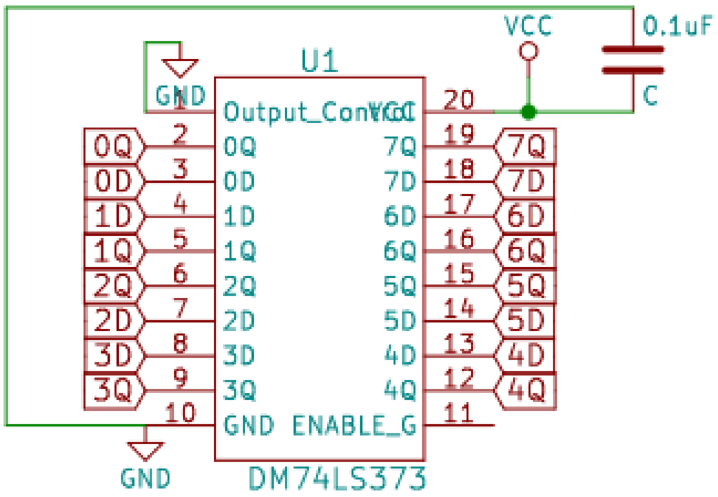
LCD Interfacing Circuit

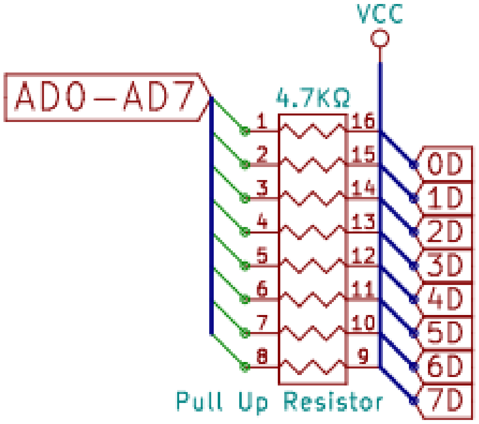


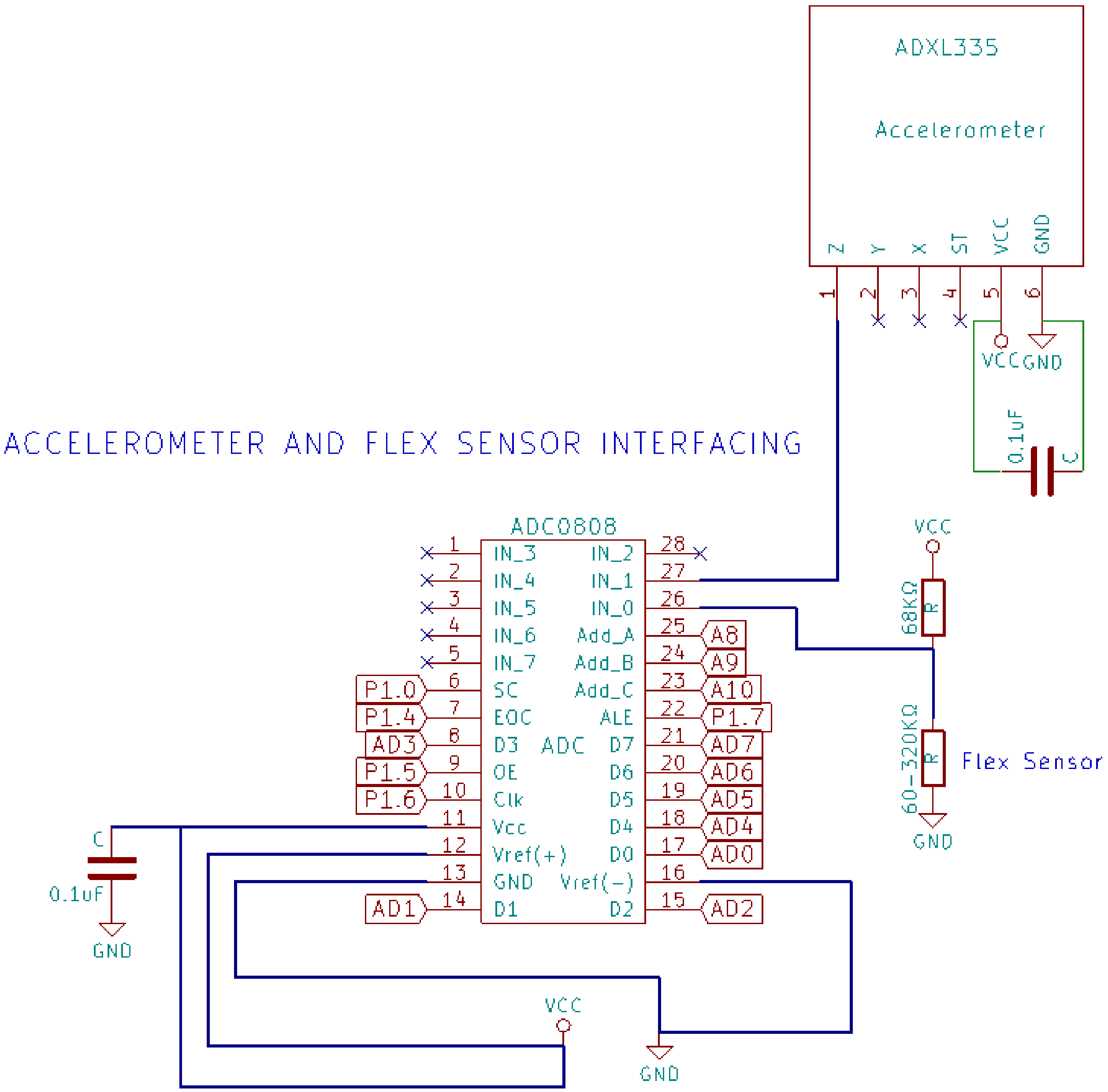
Serial Interfacing Circuit



Bootloader Switch Circuit







8.3 Appendix - Firmware Source Code:

```
/*

***********************************************************************************
 * Description  : This file contains the functions generating delays
 * Author       : Tarun
 * Date         : 22 November 2016
 * File name    : delay.c
 *

***********************************************************************************
 */
```

```
#include "delay.h"

/* Description  : This function generates a delay of 1ms
 * Input        : NA
 * Return       : NA
 */
void timer_ms()
{
    /** Timer 0, Mode 1, 16 bit timer */
    TMOD |= 0x01;

    /** initial value for 1ms */
    TH0= 0xFC;
    TL0 = 0x66;

    /** Enable global and timer 0 interrupts */
    IEN0 |= 0x82;

    /** Start timer */
    TR0 = 1;
}

/* Description  :   This function is used generate delay in us.
 *                  It genarates a approximate delay of 10us for each count,
 *                  if 5000 is passed as the argument then it generates a delay of
apprx 50ms.
 * Input        :   unsigned int representing the in us / 10
 * Return       :   NA
 */
void delay_us(unsigned int us_count)
{
    while(us_count != 0)
    {
         us_count--;
    }
}

/* Description  :   This function is used generate delay in ms.
 *                  It genarates a approximate delay of 1ms for each count,
 *                  if 1000 is passed as the argument then it generates delay of
apprx 1000ms(1sec)
 * Input        :   unsigned int representing the in ms
 * Return       :   NA
 */
void delay_ms(unsigned int ms_count)
{
    while(ms_count!=0)
    {
        delay_us(112);   //delay_us is called to generate 1ms delay
        ms_count--;
    }
}

/* Description  :   This function is used generate delay in sec .
 *                  It genarates a approximate delay of 1sec for each count,
 *                  if 10 is passed as the argument then it generates delay of
apprx 10sec
```

```
 * Input        :   unsigned int representing the in sec
 * Return       :   NA
 */
void delay_sec(unsigned char sec_count)
{
    while(sec_count!=0)
    {
        delay_ms(1000); //delay_ms is called to generate 1sec delay
        sec_count--;
    }
}
```

```
/*

************************************************************************************
 * Description  : Header File for delay.c
 * Author       : Tarun
 * Date         : 22 November 2016
 * File name    : delay.h
 *

************************************************************************************
 */

#ifndef _DELAY_H
#define _DELAY_H
#include <at89c51ed2.h>
#include <mcs51reg.h>
#include <stdio.h>
#include <stdlib.h>


/** Function prototypes */
void delay_us(unsigned int);
void delay_ms(unsigned int);
void delay_sec(unsigned char);
```

```c
void timer_ms();

#endif
/*

************************************************************************************
 * Description  : This file contains the functions for the operations to be
 *                performed on a HD44780u LCD module
 * Author       : Tarun
 * Date         : 27 October 2016
 * File name    : lcd.c
 *

************************************************************************************
 */

#include "lcd.h"

/* Description  : This function waits for the LCD to be ready again to accept
commands
 * Input        : NA
 * Return       : NA
 */
void lcdbusywait()
{
    /** Small delay to wait for the LCD operations to be complete */
    my_delay_ms(4);
}

/* Description  : This function is used to send data to the data register
 * Input        : The data to be sent
 * Return       : NA
 */
void send_data(char a)
{
    /* RS = 1. selects the data register */
    /* RW = 0, write */
    RS = 1;
    RW = 0;
    P0 = a;

    /** Enable LCD for data transfer */
    EN  = 1;
    lcdbusywait();
    EN  = 0;
}

/* Description  : This function is used to send commands to Instruction register
 * Input        : The command to be sent
 * Return       : NA
 */
void send_command(char a)
{
    /* RS = 0. selects the instruction register */
    /* RW = 0, write */
    RS = 0;
    RW = 0;
    P0 = a;

    /** Enable LCD for command transfer */
    EN  = 1;
    lcdbusywait();
    EN  = 0;
}

/* Description  : This function clears the LCD display
 * Input        : NA
 * Return       : NA
 */
void clear_display()
{
    //Command to clear display is 1
    send_command(1);
}

/* Description  : This function initializes the LCD module
 * Input        : NA
 * Return       : NA
 */
void lcdinit()
{
    send_data(0x00);
    RS = 0;
    /* RW = 1 : Read
     * RW = 0 : Write
     */
    RW = 0;
    my_delay_ms(200);

    send_command(0x30);
    my_delay_ms(50);
    send_command(0x30);
    my_delay_ms(110);
    send_command(0x30);

    /* Interface is 8 bit long, select 1 line for display */
    send_command(0x38);
    /*
     * RS  R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
     * === === === === === === === === === ===
     * 0   0   0   0   0   0   1   D   C   B
     *
     * Details
     * D: The display is ON when D = 1 and OFF when D = 0. The DD RAM contents remain unchanged.
     *
     * C: The cursor is displayed when C = 1 and is not displayed when C = 0.
     * The cursor is displayed as 5 dots in the 8th line when the 5 x 7 dot character font is selected and
     * as 5 dots in the 11th line when the 5 x 10 dot character font is selected.
     *
     * B: The character at the the cursor position blinks when B = 1.
     */
    send_command(0x0C);

    /* clear display */
    send_command(0x01);

    /* Entry Mode Set
     *
     * RS  R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
     * === === === === === === === === === ===
     * 0   0   0   0   0   0   0   1  I/D  S
     *
     * Details
     * Specifies whether to increment (I/D = 1) or decrement (I/D = 0) the address counter after subsequent
     * DD RAM read or write operations.
     *
     * If S = 1 the display will be shifted to the left (if I/D = 1) or right (if I/D = 0) on subsequent DD RAM write operations.
     * This makes it looks as if the cursor stands still and the display moves when each character is written to the DD RAM.
     * if S = 0 the display will not shift on subsequent DD RAM write operations.
     */
    send_command(0x06);
}

/* Description  : This function displays a single character on the LCD
 * Input        : The character to be displayed
 * Return       : NA
 */
void lcdputch(char a)
{
    /* Choose data register */
    RS = 1;
    RW = 0;
    send_data(a);
}

/* Description  : This function prints a string on the LCD
 * Input        : A pointer to the string to be printed
 * Return       : NA
 */
void lcdputstr(char *str)
{
    /* Loop variable */
    int i;
    /* Iterate through the string and print one character at a time,
     * stop when the \0 character is encountered
     */
    for(i = 0; str[i] != '\0'; i++)
    {
        lcdputch(str[i]);
    }
}

/* Description  : This function
 * Input        :
 * Return       : NA
 */
void shift_right()
{
    /* Cursor and Display Shift
     *
     *   RS  R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
     *   === === === === === === === === === ===
     *    0   0   0   0   0   1  S/C R/L  *   *
     *
     *  S/C R/L
     *   === ===
     *    0   0   Shifts the cursor position to the left
     *            (Address Counter is decremented by 1)
     *    0   1   Shifts the cursor position to the right
     *            (Address Counter is incremented by 1)
     *    1   0   Shifts the entire display to the left
     *            The cursor follows the display shift
     *    1   1   Shifts the entire display to the right
     *            The cursor follows the display shift
     */
    send_command(0x1C);
}



/* Description  : This function
 * Input        :
 * Return       : NA
 */
void shift_left()
{
    /* Cursor and Display Shift
     *
     *   RS  R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
     *   === === === === === === === === === ===
     *    0   0   0   0   0   1  S/C R/L  *   *
     *
     *  S/C R/L
     *   === ===
     *    0   0   Shifts the cursor position to the left
     *            (Address Counter is decremented by 1)
     *    0   1   Shifts the cursor position to the right
     *            (Address Counter is incremented by 1)
     *    1   0   Shifts the entire display to the left
     *            The cursor follows the display shift
     *    1   1   Shifts the entire display to the right
     *            The cursor follows the display shift
     */
    send_command(0x18);
}

/* Description  : This function
 * Input        :
 * Return       : NA
 */
void lcdgotoaddr(char x)
{
    send_command(0x80 + x);
}

/* Description  : This function goes to a particular cursor location
 * Input        : Line number, and Character number within the line
 * Return       : NA
 */
void lcdgotoxy(char x, char y)
{
    if (x == 1)
    {
        lcdgotoaddr(0x00 + y - 1);
    }
    else if (x == 2)
    {
        lcdgotoaddr(0x40 + y - 1);
    }
    else if (x == 3)
    {
        lcdgotoaddr(0x10 + y - 1);
    }
    else if (x == 4)
    {
        lcdgotoaddr(0x50 + y - 1);
    }
}

/* Description  : This function reads and returns the DDRAM data
 * Input        : NA
 * Return       : DDRAM data
 */
unsigned char read_ddram_data()
{
    char ddram_data;

    /* RW = 1 : Read
     * RW = 0 : Write
     */
    /* RS = 0 : Instruction Register
     * RS = 1 : Data Register
     */
     RS = 1;
     RW = 1;
     ddram_data = *(LCD_MAP);
     return ddram_data;
}

/* Description  : This function reads and returns the DDRAM address
 * Input        : NA
 * Return       : DDRAM address
 */
char read_ddram_addr()
{
    char ddram_addr;

    /* To read DDRAM address, RS = 0, and RW = 1 */
    /* Get DDRAM address */
    RS = 0;
    RW = 1;
    ddram_addr = *(LCD_MAP);

    return ddram_addr;
}

/* Description  : This function forces the LCD to print in the next line
 * Input        : NA
 * Return       : NA
 */
void goto_next_line()
{
    char addr = read_ddram_addr();

    if (addr == 0x10)
    {
        lcdgotoxy(2,1);
    }

    else if (addr == 0x50)
    {
        lcdgotoxy(3,1);
    }

    else if (addr == 0x20)
    {
        lcdgotoxy(4,1);
    }

    else if (addr == 0x61)
    {
        clear_display();
        lcdgotoxy(1,1);
    }
}

/* Description  : This function creates a custom character
 * Input        : Character code and the Eight row values
 * Return       : NA
 */
void lcd_create_custom_char(unsigned char char_code, unsigned char pixel_val[8])
{
    /** Loop variable */
    unsigned char i;

    /*******************************
    Create CGRAM address byte
    D7  D6  D5  D4  D3  D2  D1  D0
    0   1   C3  C2  C1  R3  R2  R1

    C3, C2, C1 is the character code
    R3, R2, R1 is the row number
    ********************************/

    unsigned char cgram_addr = 0;

    /** Make bit_6 1 */
    cgram_addr |= 0x40;

    /** Store the character code at D5, D4, D3 */
    cgram_addr |= (char_code << 3);

    for (i = 0; i < 8; i++)
    {
        /** Set CGRAM address */
        RS = 0;
        RW = 0;
        send_command(0x40 | cgram_addr);

        lcdbusywait();
        /* Select data register */
        RS = 1;
        RW = 0;

        send_data(pixel_val[i]);
        EN = 1;
        my_delay_ms(5);
        EN = 0;

        /* Go to the next row */
        cgram_addr++;
    }
}



/* Description  :   This function creates a custom character for representing
                    speed of motor in pictorial way
 * Input        :   Character code and the Eight row values
 * Return       :   NA
 */
void graphical_percentage_display()
{
    unsigned char pixel_val[8];

    /* Custom character 0 */
    pixel_val[0] = 0x10;
    pixel_val[1] = 0x10;
    pixel_val[2] = 0x10;
```

```
pixel_val[3] = 0x10;
pixel_val[4] = 0x10;
pixel_val[5] = 0x10;
pixel_val[6] = 0x10;
pixel_val[7] = 0x10;
lcd_create_custom_char(0, pixel_val);

/* Custom character 1 */
pixel_val[0] = 0x18;
pixel_val[1] = 0x18;
pixel_val[2] = 0x18;
pixel_val[3] = 0x18;
pixel_val[4] = 0x18;
pixel_val[5] = 0x18;
pixel_val[6] = 0x18;
pixel_val[7] = 0x18;
lcd_create_custom_char(1, pixel_val);

/* Custom character 2 */
pixel_val[0] = 0x1C;
pixel_val[1] = 0x1C;
pixel_val[2] = 0x1C;
pixel_val[3] = 0x1C;
pixel_val[4] = 0x1C;
pixel_val[5] = 0x1C;
pixel_val[6] = 0x1C;
pixel_val[7] = 0x1C;
lcd_create_custom_char(2, pixel_val);

/* Custom character 3 */
pixel_val[0] = 0x1E;
pixel_val[1] = 0x1E;
pixel_val[2] = 0x1E;
pixel_val[3] = 0x1E;
pixel_val[4] = 0x1E;
pixel_val[5] = 0x1E;
pixel_val[6] = 0x1E;
pixel_val[7] = 0x1E;
lcd_create_custom_char(3, pixel_val);

/* Custom character 4 */
pixel_val[0] = 0x1F;
pixel_val[1] = 0x1F;
pixel_val[2] = 0x1F;
pixel_val[3] = 0x1F;
pixel_val[4] = 0x1F;
pixel_val[5] = 0x1F;
pixel_val[6] = 0x1F;
pixel_val[7] = 0x1F;
lcd_create_custom_char(4, pixel_val);

/* Custom character 5 */
pixel_val[0] = 0x01;
pixel_val[1] = 0x01;
pixel_val[2] = 0x01;
pixel_val[3] = 0x01;
pixel_val[4] = 0x01;
```

```
    pixel_val[5] = 0x01;
    pixel_val[6] = 0x01;
    pixel_val[7] = 0x01;
    lcd_create_custom_char(5, pixel_val);

    /* Custom character 6 */
    pixel_val[0] = 0x00;
    pixel_val[1] = 0x00;
    pixel_val[2] = 0x00;
    pixel_val[3] = 0x00;
    pixel_val[4] = 0x00;
    pixel_val[5] = 0x00;
    pixel_val[6] = 0x00;
    pixel_val[7] = 0x00;
    lcd_create_custom_char(6, pixel_val);

    /* Custom character 6 */
    pixel_val[0] = 0x1F;
    pixel_val[1] = 0x1F;
    pixel_val[2] = 0x1F;
    pixel_val[3] = 0x1F;
    pixel_val[4] = 0x00;
    pixel_val[5] = 0x00;
    pixel_val[6] = 0x00;
    pixel_val[7] = 0x00;
    lcd_create_custom_char(7, pixel_val);
}

/* Description  : This function generates a delay of 1 ms
 * Input        : Delay needed in ms
 * Return       : NA
 */
void my_delay_ms(int a)
{
    int i, j;
    for(i = 0; i < a; i++)
    {
        for(j = 0; j < 124; j++)
        {

        }
    }
}
```

```
/*

************************************************************************************
 * Description   : Header file for lcd.c
 * Author        : Tarun
 * Date          : 27 October 2016
 * File name     : lcd.h
 *

************************************************************************************
 */
#ifndef LCD_H
#define LCD_H

#include <at89c51ed2.h>
#include <mcs51reg.h>
#include <stdio.h>
#include <stdlib.h>

/** Pin definitions for LCD data and control */
#define RS  P1_5
#define RW  P1_6
#define EN  P1_7
#define D0  P0_0
#define D1  P0_1
#define D2  P0_2
#define D3  P0_3
#define D4  P0_4
#define D5  P0_5
#define D6  P0_6
#define D7  P0_7
#define BUSY_MASK 0x80
#define LCD_MAP ((xdata unsigned char *)0x8000)

/** Function prototypes */
void lcdinit();
void lcdbusywait();
void lcdgotoaddr(char x);
void lcdgotoxy(char x, char y);
void lcdputch(char a);
void lcdputstr(char *a);
void my_delay_ms(int a);
void send_data(char a);
void send_command(char a);
void clear_display();
void shift_right();
void shift_left();
void goto_next_line();
char read_ddram_addr();
void lcd_create_custom_char(unsigned char char_code, unsigned char pixel_val[]);
unsigned char read_ddram_data();
void graphical_percentage_display();

#endif // LCD_H
```

```
/*
***********************************************************************************
 * Description  : This file contains the main function for the final project
 * Author       : Tarun
 * Date         : 22 November 2016
 * File name    : main.c
 *
***********************************************************************************
 */

#include "main.h"

/** Global variable for PIR Sensor */
unsigned char PIRFlag = 0;

_sdcc_external_startup()
{
    //Setting the RS0:RS1 bits in the AUXR register
    //bits 2 and 3, enable 1 KB of internal XRAM
    AUXR |= 0x0C;

    //Setting the baud rate to 9600
    /* Timer 1, 8 bit auto reload mode */
    TMOD = 0x20;
    TH1 = -3;
    /* Serial mode 1, 8 bit data, 1 start bit, 1 stop bit, Receive data from RxD
pin */
    SCON = 0x52;
    /* Start timer */
    TR1 = 1;

    TCON |= 0x01;

    //Note: need to return from startup
    return 0;
}

void main(void)
{
    /* A variable to store the message number */
    unsigned char message_no, message_no_1;
    unsigned char i;
    unsigned char read[17];
    PIRFlag = 0;

    /* Initialize LCD */
    lcdinit();

    /* Initialize PWM */
    init_pwm();

    /** Create custom characters for the graphical representation of speed */
    graphical_percentage_display();
```

```
lcdgotoxy(1,1);
lcdputstr("GSM interfacing");
while (1)
{
    //printf_tiny("\r\nHello");
    /* Return message will be of the format <CR><LF>+CMTI: "SM",3<CR><LF> */
    /* Carriage return */
    while(getchar() != CARRIAGE_RETURN);
        /* New Line */
        while(getchar() != NEW_LINE);

            /* Checking for +CMTI:"SM",3 */
            if(getchar() == '+')
            {
                //printf_tiny("\n\rHello1");
                if(getchar() == 'C')
                {
                    //printf_tiny("\n\rHello2");
                    if(getchar() == 'M')
                    {
                        //printf_tiny("\n\rHello3");
                        if(getchar() == 'T')
                        {
                            //printf_tiny("\n\rHello4");
                            if(getchar() == 'I')
                            {
                                //printf_tiny("\n\rHello5");
                                while(getchar() != ',');

                                /* Get the SMS number */
                                message_no = getchar();
                                message_no_1 = getchar();

                                delay_ms(10);
                                put_string("AT");
                                putchar(CARRIAGE_RETURN);
                                putchar(NEW_LINE);

                                /* Set the operating mode to sms text mode */
                                //put_string("at+cmgf=1");
                                put_string("AT+CMGF=1");
                                putchar(CARRIAGE_RETURN);
                                putchar(NEW_LINE);

                                /* Read the sms */
                                put_string("AT+CMGR=");
                                putchar(message_no);
                                if (message_no_1 != CARRIAGE_RETURN)
                                {
                                    putchar(message_no_1);
                                }
                                /* Carriage return */
                                putchar(CARRIAGE_RETURN);
                                /* New Line */
                                putchar(NEW_LINE);
```

```
                                    /* Wait for carriage return */
                                    while (getchar()!= CARRIAGE_RETURN);
                                    /* Wait for new line */
                                    while (getchar()!= NEW_LINE);
                                    while (getchar()!= CARRIAGE_RETURN);
                                    while (getchar()!= NEW_LINE);
                                    //while (getchar()!= NEW_LINE);
                                    while (getchar()!= CARRIAGE_RETURN);
                                    while (getchar()!= NEW_LINE);

                                    while (getchar()!= CARRIAGE_RETURN);
                                    while (getchar()!= NEW_LINE);
                                    while (getchar()!= CARRIAGE_RETURN);
                                    while (getchar()!= NEW_LINE);
                                    while (getchar()!= CARRIAGE_RETURN);
                                    while (getchar()!= NEW_LINE);

                                    /** Store the received text message into the
array */
                                    for (i = 0; i <= 15; i++)
                                    {
                                        read[i] = getchar();

                                        if (read[i] == CARRIAGE_RETURN)
                                        {
                                            read[i] = '\0';
                                            break;
                                        }
                                        read[16] = '\0';
                                    }
                                    lcdgotoxy(2, 1);

                                    lcdputstr("                ");
                                    delay_ms(2);
                                    lcdgotoxy(2, 1);
                                    delay_ms(2);

                                    /** Display the text message receved from GSM
module onto the LCD */
                                    lcdputstr(read);

                                    /** Decide whether to change the direction or
speed based on the sms command received  */
                                    motor_driver(read);
                                }
                            }
                        }
                    }
                }
    }
}

/** ISR for external interrupt 0 */
void external0 (void) __interrupt(0)
{
    /** Check the previous state of the motor and decide
     *  whether to turn the motor on or off */
    if (PIRFlag)
    {
        motor_on();
        PIRFlag = 0;
    }

    else if (PIRFlag == 0)
    {
        motor_off();
        PIRFlag = 1;
    }
}

void putchar(char ch)
{
    // load serial port with transmit value
    while(TI == 0);

    SBUF = ch;
    // clear TI flag
    TI=0;
}

char getchar ()
{
    // wait for character to be received, spin on RI
    while ((SCON & 0x01) == 0);

    // clear RI flag
    RI = 0;
    // return character from SBUF
    return SBUF;
}

void put_string(unsigned char *str)
{
    while(*str)
    {
        putchar(*str);
        str++;
    }
}
```

```
/*
```

```
***********************************************************************************
 * Description  : Header file for main.c
 * Author       : Tarun
 * Date         : 22 November 2016
 * File name    : main.h
 *

***********************************************************************************
 */
#ifndef _MAIN_H
#define _MAIN_H

#include <at89c51ed2.h>
#include <mcs51reg.h>
#include <stdio.h>
#include <stdlib.h>
#include "lcd.h"
#include "delay.h"
#include "pwm.h"
#include "motor.h"

/** Macros for new line and carriage return */
#define NEW_LINE 0x0a
#define CARRIAGE_RETURN 0x0d

/** Function prototypes */
void put_string(unsigned char *str);
void motor_driver(char * read);

#endif // _MAIN_H
```

```
/*

***********************************************************************************
 * Description  : This file contains the functions for the motor control operations
 * Author       : Tarun
 * Date         : 27 October 2016
 * File name    : motor.c
 *

***********************************************************************************
 */
#include "motor.h"
#include "stdlib.h"
#include "lcd.h"
#include "pwm.h"
```

```
/***********************************************************************************
***
Description    : This function is used to decipher the message received from the GSM
                 module, and decides the direction of the motor rotation and the
speed
                 of rotation of motor.

Input          : The message received from the GSM module
Output         : Various control signals to control the direction and speed of the
                 motor.
************************************************************************************
**/
void motor_driver(char * read)
{
    unsigned char duty_cycle[3];
    unsigned char duty_cycle_val;

    /*******************************DIRECTION
CONTROL******************************************/


/***********************************************************************************
*********
    Please note that:
    P1.0 (PORT 1 BIT 6 of the microcontroller) is connected to Input 1 of the L293D
motor driver
    P1.2 is connected to Input 2 of the L293D motor driver
    ---------------------------------------------
    P1.0    P1.2    Directiion of motor rotation
    ---------------------------------------------
    0       1       Counter clockwise
    1       0       Clockwise

************************************************************************************
********/


/***********************************************************************************
***
    Check if the message received is for clockwise direction control:
    If the message "cw" is received, then need to rotate the motor in clockwise
direction
    According to the table above, need to send the following signals:
    P1.0 = 1;
    P1.2 = 0;

************************************************************************************
***/
    if ((read[0] == 'c') || (read[0] == 'C'))
    {
        if (read[1] == 'w')
        {
            previous_duty_cycle = ((255 - CCAP0H) * 100) / 255;
            slow_down_motor(0);
            P1_0 = 1;
            P1_2 = 0;
            speed_up_motor(previous_duty_cycle);
            lcdgotoxy(3, 1);
            lcdputstr("                ");
            lcdgotoxy(3, 1);
            lcdputstr("Motor ON:cw:");
        }
    }


/**********************************************************************************
***
    Check if the message received is for counter-clockwise direction control:
    If the message "ccw" is received, then need to rotate the motor in counter-
clockwise
    direction.
    According to the table above, need to send the following signals:
    P1.0 = 0;
    P1.2 = 1;

***********************************************************************************
***/
    if ((read[0] == 'c') || (read[0] == 'C'))
    {
        if (read[1] == 'c')
        {
            if (read[2] == 'w')
            {
                previous_duty_cycle = ((255 - CCAP0H) * 100) / 255;
                slow_down_motor(0);
                P1_0 = 0;
                P1_2 = 1;
                speed_up_motor(previous_duty_cycle);
                lcdgotoxy(3, 1);
                lcdputstr("                ");
                lcdgotoxy(3, 1);
                lcdputstr("Motor ON:ccw:");
            }
        }
    }

    /*******************************SPEED
CONTROL******************************************/

/**********************************************************************************
****
    Please note:
    For Speed control, the first two characters of the message has to be "sc",
indicating
    speed control, followed by 2 digits that represent the desired duty cycle
    For example,
    sc50    : means 50% duty cycle
    sc95    : means 95% duty cycle

***********************************************************************************
***/
```

```
/**********************************************************************************
****
    The connections:
    The PWM output of PCA is available at P1.3
    P1.3 is connected to Enable 1, that is pin 1 of the L293D motor driver

***********************************************************************************
***/
    if ((read[0] == 's') || (read[0] == 'S'))
    {
        if (read[1] == 'c')
        {
            if ((read[2] >= '0') && (read[2] <= '9'))
            {
                /** Display left border */
                lcdgotoxy(4, 1);
                send_data(0);

                /** Display right border */
                lcdgotoxy(4, 12);
                send_data(5);

                /** Read the tens digit
                    For example, if the speed is 68%, read and store '6'
                 */
                duty_cycle[0] = read[2];

                /** Call the function to graphically disply this digit on the LCD
*/
                display_duty_cycle_tenth(read[2]);

                if ((read[3] >= '0') && (read[3] <= '9'))
                {
                    duty_cycle[1] = read[3];
                    duty_cycle[2] = '\0';

                    display_duty_cycle_ones(read[3], read[2]);

                    duty_cycle_val = atoi(duty_cycle);

                    change_duty_cycle(duty_cycle_val);

                    lcdgotoxy(3, 1);
                    lcdputstr("                ");
                    lcdgotoxy(3, 1);
                    if ((P1_0) && (!P1_2))
                    {
                        lcdputstr("Motor ON:cw:");
                    }
                    else
                    {
                        lcdputstr("Motor ON:ccw:");
                    }
                    lcdputch(read[2]);
                    lcdputch(read[3]);
                    lcdputch('%');
                }
            }
        }
    }
}

/* Description  : This function turns the motor ON
 * Input        : NA
 * Return       : NA
 */
void motor_on()
{
    /** Turn on the PWM counter */
    CCON |= 0x40;

    /** Driver will turn off motor if both the inputs are of the same polarity */
    /** Clockwise rotation by default */
    P1_0 = 1;
    P1_2 = 0;

    /** Default duty cycle is 50% */
    CCAP0H = 127;

    lcdgotoxy(3, 1);
    lcdputstr("                ");
    lcdgotoxy(3, 1);
    lcdputstr("Motor ON:cw");
}

/* Description  : This function turns the motor OFF
 * Input        : NA
 * Return       : NA
 */
void motor_off()
{
    /** Turn on the PWM counter */
    CCON &= ~0x40;

    /** Driver will turn off motor if both the inputs are of the same polarity */
    /** Clockwise rotation by default */
    P1_0 = 1;
    P1_2 = 1;
    lcdgotoxy(3, 1);
    lcdputstr("                ");
    lcdgotoxy(3, 1);
    lcdputstr("Motor OFF");
}

/* Description  :   This function displays the digit in the ones place of the duty
cycle
                    For example, consider 56% duty cycle, This function extracts
the digit
                    '6' and displays it on the LCD
 * Input        : NA
 * Return       : NA
 */
void display_duty_cycle_ones(unsigned char ones, unsigned char tenth)
```

```
{
    /* Loop variable */
    unsigned char i = 0;
    unsigned char temp[2] = {ones, '\0'};
    unsigned char temp1[2] = {tenth, '\0'};
    unsigned char ones_val = atoi(temp);
    unsigned char tenth_val = atoi(temp1);


    lcdgotoxy(4, tenth_val + 2);
    my_delay_ms(2);
    /* Logic for displaying the custom character for the ones digit */

    if (ones_val == 0)
    {
        send_data(6);
    }

    else if (ones_val <= 2)
    {
        send_data(0);
    }

    else if (ones_val <= 4)
    {
        send_data(1);
    }

    else if (ones_val <= 6)
    {
        send_data(2);
    }

    else if (ones_val <= 8)
    {
        send_data(3);
    }

    else
    {
        send_data(4);
    }
    my_delay_ms(2);
}

/************************************************************
Description:
This function takes the duty cycle value in percentage, and
it gradually slows down the motor by 5% at a time till it
reaches the desired duty cycle
************************************************************/
void slow_down_motor(float desired_duty_cycle)
{
    float assign_duty_cycle;

    assign_duty_cycle = (255 - (desired_duty_cycle * 2.55));
    while (CCAP0H < assign_duty_cycle)
    {
        CCAP0H += 12.75;
        my_delay_ms(40);
    }

    CCAP0H = (255 - (desired_duty_cycle * 2.55));
    my_delay_ms(40);
}

/***********************************************************
Description:
This function takes the duty cycle value in percentage, and
it gradually speeds up the motor by 5% at a time till it
reaches the desired duty cycle
************************************************************/
void speed_up_motor(float desired_duty_cycle)
{
    float assign_duty_cycle;

    assign_duty_cycle = (255 - (desired_duty_cycle * 2.55));

    while (CCAP0H > assign_duty_cycle)
    {
        CCAP0H -= 12.75;
        my_delay_ms(40);
    }

    CCAP0H = (255 - (desired_duty_cycle * 2.55));
    my_delay_ms(40);
}

/* Description  :   This function displays the digit in the tens place of the duty
cycle
                    For example, consider 56% duty cycle, This function extracts
the digit
                    '5' and displays it on the LCD
 * Input        : NA
 * Return       : NA
 */
void display_duty_cycle_tenth(unsigned char tenth)
{
    /* Loop variable */
    unsigned char i = 0;
    unsigned char temp[2] = {tenth, '\0'};
    unsigned char tenth_val = atoi(temp);

    lcdgotoxy(4, 2);
    my_delay_ms(2);
    lcdputstr("          ");
    my_delay_ms(2);

    for (i = 0; i < tenth_val; i++)
    {
        lcdgotoxy(4, i + 2);
        my_delay_ms(2);
        send_data(4);
```

```c
        my_delay_ms(2);
    }
}
```

```c
/*
***********************************************************************************
 * Description  : Header file for motor.c
 * Author       : Tarun
 * Date         : 22 November 2016
 * File name    : motor.h
***********************************************************************************
 */
#ifndef _MOTOR_H
#define _MOTOR_H
#include <at89c51ed2.h>
#include <mcs51reg.h>

/** Function prototypes */
void motor_driver(char * read);
void display_duty_cycle_ones(unsigned char ones, unsigned char tenth);
void display_duty_cycle_tenth(unsigned char tenth);
void slow_down_motor(float desired_duty_cycle);
void speed_up_motor(float desired_duty_cycle);
void motor_on();
void motor_off();

#endif // _MOTOR_H
```

```c
/*
***********************************************************************************
 * Description  : This file contains the functions generating PWM to drive the
motor
 * Author       : Tarun
 * Date         : 22 November 2016
 * File name    : PWM.c
 *
```

```
*************************************************************************************
 */

#include "pwm.h"
#include "motor.h"


/* Description  : This function initializes the PWM module
 * Input        : NA
 * Return       : NA
 */
void init_pwm()
{
    /* Internal clock, F / 6, No interrupt */
    CMOD = 0x00;

    /* Positive edge capture, PWM mode(Bit 1) */
    CCAPM0 = 0x42;

    /* Turn the counter on */
    CCON = 0x40;

    /* Enable PCA interrupt? */
    IEN0 |= 0xC1;

    /* Default duty cycle is 50% */
    CCAP0H = 127;

    /* By default motor is off */
    motor_off();
}

/* Description  :   This function is used to change the duty cycle of
 *                  pwm to the given % value
 * Input        :   Duty cycle value in percentage
 * Return       :   NA
 */
void change_duty_cycle(unsigned char duty_cycle)
{
    /** Local temporary variable */
    float assign_duty_cycle;

    /************************************************
    Please Note:
    255 represents 100% duty cycle
    Each percent is 2.55
    We get the input in terms of percentage, to convert
    it to a scale of 255, we multiply by 2.55
    ***********************************************/
    assign_duty_cycle = duty_cycle * 2.55;

    /** Check if the desired duty cycle is less than the
      * desired duty cycle
      */
    if (CCAP0H > assign_duty_cycle)
    {
        /** Call the function to speed up the motor */
        speed_up_motor(assign_duty_cycle);
    }

    /** Check if the desired duty cycle is greater than the
      * desired duty cycle
      */
    else if (CCAP0H < assign_duty_cycle)
    {
        /** Call the function to slow down the motor */
        slow_down_motor(assign_duty_cycle);
    }
}
```

```
/*
 
***********************************************************************************
 * Description  : Header File for pwm.c
 * Author       : Tarun
 * Date         : 22 November 2016
 * File name    : pwm.h
 *
 
***********************************************************************************
 */

#ifndef _PWM_H
#define _PWM_H
#include <at89c51ed2.h>
#include <mcs51reg.h>

float previous_duty_cycle;

/** Function prototypes */
void init_pwm();
void change_duty_cycle(unsigned char duty_cycle);

#endif // _PWM_H
```

```
/*
```

```
************************************************************************************
 * Description  : This file contains the functions for the motor control using Flex
                  sensor and accelerometer
 * Author       : Subhradeep
 * Date         : 27 October 2016
 * File name    : main-org.c
 *

************************************************************************************
/

//Including the required header files for 8051 microcontroller
#include <at89c51ed2.h>
#include <mcs51reg.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include "delays.h"

#define SC P1_0
#define EOC P1_4
#define OE P1_5
#define CLK P1_6
#define ALE P1_7
#define MOTORIN1 P3_4
#define MOTORIN2 P3_5
#define ADDRESS_A P2_0
#define ADDRESS_B P2_1
#define ADDRESS_C P2_2
#define FLEX 0
#define ACCEL 1

//Define global variables

uint8_t ADCdata;
float duty_cycle;
int multiplier;
int interrupt_counter;
int sensor;
/*****************************************************STARTUP
CODE********************************************************************/

_sdcc_external_startup()//Startup function that is executed first when the
microcontroller is turned on
{

    TI=0;        //Set TI flag to 0
    AUXR|=0x0C;  //Set the XRS1:XRS0 bit to 1 to enable 1 KB of internal extended
RAM
    TMOD=0X20;   //Enable Timer 1 Mode 2
    TH1=0xFF;    //Load TH1 with FD for 57600 baud rate
    SCON=0X50;   //Select Mode 1 for full duplex with variable baud rate
    PCON|=0x80;
    TR1=1;       //Start Timer

    return 0;
```

```
}
/********************************************************PUTCHAR
DEFINITION************************************************************/
void putchar (char c)//Definition fo putchar function to display characters on the
serial terminal
{
    SBUF = c;    //SBUF stores the character to be dsiplayed
    while (!TI); //Wait for TI flag
    TI = 0;      //Set TI flag to 0
}
/********************************************************GETCHAR
DEFINITION************************************************************/
char getchar()      //Definition of getchar function to fetch input from the user
{
    while(RI==0);   //Wait for RI flag
    RI=0;           //Set RI to 0
    return SBUF;    //Return the value in the buffer
}

/**********************************************************************************
***
Description   : This function is used to load the register values to intialize PWM
                using PCA module on P1.3 using module 0

Input         : N/A
Output        : N/A
***********************************************************************************
**/


void PWMInit()
{
    CMOD&=0x79;//Enable PCA Counter during idle mode and set Fclk/6
    CCAPM0|=0x42;// Enable PWM mode and enable comparator
    CCAP0L=duty_cycle;//Set to 0x99 which is 153 in decimal, 153/255=60% which is
the Toff so Ton is 40%
    CCAP0H=duty_cycle;//Set to 40% duty cycle initially
    CL=0x00;
    CH=0x00;
}

/**********************************************************************************
***
Description   : This function is used to turn on the PWM on P1.3

Input         : N/A
Output        : N/A
***********************************************************************************
**/

void PWMON()
{
    PWMInit();
    CCON|=0x40;//Enable PCA Counter Run Control Bit
    MOTORIN1=1;//Set motor inputs to 1 and 0 for clockwise motion
    MOTORIN2=0;
```

```
}
/***********************************************************************************
***
Description   : This function is used to turn off the PWM on P1.3

Input         : N/A
Output        : N/A
************************************************************************************
**/

void PWMOFF()
{
    CCON&=0xBF;//Disable timer
    MOTORIN1=0;//Turn off the motor inputs
    MOTORIN2=0;//Both inputs set to 0
}


/***********************************************************************************
***
Description   : This function is used to write the new duty cycle values to the
                corresponding registers CCAP0L and CCAP0H
Input         : N/A
Output        : N/A
************************************************************************************
**/


void WriteDutyCycleValues()
{
    CCAP0L=duty_cycle;//Set to 0x99 which is 153 in decimal, 153/255=60% which is
the Toff so Ton is 40%
    CCAP0H=duty_cycle;
}

/***********************************************************************************
***
Description   : This function is used to slowly decrease the duty cycle by 5% at a
                time until it reaches 5%
Input         : N/A
Output        : N/A
************************************************************************************
**/

void SlowDownMotor()
{
    do
    {
        duty_cycle+=0x0D;
        WriteDutyCycleValues();
        Delay(40);
    }while(duty_cycle<=0xF2);//Keep decrmenting the duty cycle until it reaches 5%
}

/***********************************************************************************
***
```

```
Description   : This function is used to slowly increase the duty cycle by 5% at a
                time until it reaches the desired duty cycle
Input         : target duty cycle
Output        : N/A
***********************************************************************************
**/


void SpeedUpMotor(int target_duty)
{
    do
    {
        duty_cycle-=0x0D;
        printf_tiny("Duty %d", duty_cycle);
        WriteDutyCycleValues();
        Delay(40);
    }while((duty_cycle>=(target_duty+0x0D))&&(duty_cycle>=0x0D));//Keep
incrementing the duty cycle until it hits the target duty cycle
}

/**********************************************************************************
***
Description   : This function is used to slowly decrease the duty cycle to 5% and
then
                gradually increase the duty cycle to the original value but in the
                opposite direction. It is used to reverse the motor rotation
Input         : N/A
Output        : N/A
***********************************************************************************
**/


void MotorReverse()
{
    int old_duty_cycle;
    old_duty_cycle=duty_cycle;//Store the current duty cycle valie
    SlowDownMotor();//Slow down the motor to 5% from its current duty cycle
    MOTORIN1=!(MOTORIN1);//Toggle the motor inputs to reverse the direction of
rotation
    MOTORIN2=!(MOTORIN2);
    SpeedUpMotor(old_duty_cycle);//Speed up the motor again to the original duty
cycle but opposite direction
    printf_tiny("\n\rMotor direction reversed\n\r");
}

/**********************************************************************************
***
Description   : This function is used to set a particular duty cycle for the motor
Input         : desired duty cycle
Output        : N/A
***********************************************************************************
**/


void SetSpeed(float speed)
{
    duty_cycle=speed;
    WriteDutyCycleValues();
}

/**********************************************************************************
***
Description   : This function is used to map the output of the flex sensor to 
varying 
                duty cycles after reading them from the ADC
Input         : N/A
Output        : N/A
***********************************************************************************
**/

//The ADC output for the flex sensor was intially tested to obtain the below used 
values
void ADCToMotorSpeed()
{
    int percentspeed=0;
        if(ADCdata>=80 && ADCdata<=105)
        multiplier=16;
        else if(ADCdata>=106 && ADCdata<=110)
        multiplier=15;
        else if(ADCdata>=111 && ADCdata<=115)
        multiplier=14;
        else if(ADCdata>=116 && ADCdata<=120)
        multiplier=13;
        else if(ADCdata>=121 && ADCdata<=125)
        multiplier=12;
        else if(ADCdata>=126 && ADCdata<=130)
        multiplier=11;
        else if(ADCdata>=131 && ADCdata<=135)
        multiplier=10;
        else if(ADCdata>=136 && ADCdata<=140)
        multiplier=9;
        else if(ADCdata>=141 && ADCdata<=145)
        multiplier=8;
        else if(ADCdata>=146 && ADCdata<=150)
        multiplier=7;
        else if(ADCdata>=151 && ADCdata<=155)
        multiplier=6;
        else if(ADCdata>=156 && ADCdata<=160)
        multiplier=5;
        else if(ADCdata>=161 && ADCdata<=165)
        multiplier=4;
        else if(ADCdata>=166 && ADCdata<=170)
        multiplier=3;
        else if(ADCdata>=171 && ADCdata<=175)
        multiplier=2;
        else if(ADCdata>=176)
        multiplier=1;
        percentspeed=(21-multiplier)*5;//Calculate the percentage speed based on 
the multiplier value
        SetSpeed(multiplier*12.75);//12.75 is 5% of 255 so the duty cycle is set 
accordingly
        printf("\n\r Speed updated to %d %c of duty cycle by Flex 
Sensor",percentspeed,37);
}
```

```
/*********************************************************************************
***
Description   : This function is used to map the output of the accelerometer to 
varying
                duty cycles after reading them from the ADC
Input         : N/A
Output        : N/A
**********************************************************************************
**/
//The ADC output for the accelerometer was intially tested to obtain the below used 
values
/**Since the Vref(+) and Vref(-) was 5V and 0V respectively and the output at 0g 
for the accelerometer
is close to 3V so a very small voltage range on the accelerometer is mapped to a 
wide range on the ADC*/
void ACCLToMotorSpeed()
{
        int percentspeed=0;
        if(ADCdata>=68 && ADCdata<=69)
        multiplier=18;
        else if(ADCdata>=70 && ADCdata<=71)
        multiplier=17;
        else if(ADCdata>=72 && ADCdata<=73)
        multiplier=16;
        else if(ADCdata>=74 && ADCdata<=75)
        multiplier=15;
        else if(ADCdata>=76 && ADCdata<=77)
        multiplier=14;
        else if(ADCdata>=78 && ADCdata<=79)
        multiplier=13;
        else if(ADCdata>=80 && ADCdata<=81)
        multiplier=12;
        else if(ADCdata>=82 && ADCdata<=83)
        multiplier=11;
        else if(ADCdata>=84 && ADCdata<=85)
        multiplier=10;
        else if(ADCdata>=86 && ADCdata<=87)
        multiplier=9;
        else if(ADCdata>=88 && ADCdata<=89)
        multiplier=8;
        else if(ADCdata>=90 && ADCdata<=91)
        multiplier=7;
        else if(ADCdata>=92 && ADCdata<=93)
        multiplier=6;
        else if(ADCdata>=94 && ADCdata<=95)
        multiplier=5;
        else if(ADCdata>=96 && ADCdata<=97)
        multiplier=4;
        else if(ADCdata>=98 && ADCdata<=99)
        multiplier=3;
        else if(ADCdata>=100 && ADCdata<=101)
        multiplier=2;
        else if(ADCdata>=102)
        multiplier=1;
        percentspeed=(21-multiplier)*5;//Calculate the percentage speed based on
the multiplier value
        SetSpeed(multiplier*12.75);//12.75 is 5% of 255 so the duty cycle is set
accordingly
        printf("\n\r Speed updated to %d %c of duty cycle by
Accelerometer",percentspeed,37);
}
/**********************************************************************************
***
Description   : This function starts timer0 for the first time
Input         : N/A
Output        : N/A
***********************************************************************************
**/
void timer0() //Start Timer 0
{

    TL0=0x4B;
    TH0=0xFC;
    TR0=1;
}

/**********************************************************************************
***
Description   : This function is used to intialize the various pins of the ADC
before
                a conversion is performed
Input         : N/A
Output        : N/A
***********************************************************************************
**/


void ADCInit()
{
   ADCdata=0x00;//Intialize to zero
    EOC=1;//Set End of Conversion to high
    ALE=0;//Set Address Latch Enable to low
    OE=0;//Set Output Enable to low
    SC=0;//Set start conversion to low
    P0=0xFF;
}

/**********************************************************************************
***
Description   : This function is used to generate a clock pulse for the ADC to
sample
                data
Input         : N/A
Output        : N/A
***********************************************************************************
**/

void ADCClk()
{
    CLK=1;//Toggle the clock pin high and low after a short delay
    FastDelay(1);
    CLK=0;
    FastDelay(1);

}

/**********************************************************************************
***
Description   : This is the ISR for timer 0 running in mode 1. The code inside the
ISR
                is executed every 1 second. Every time the code is executed, the
ADC
                reads the values from the flex sensor or accelerometer depending
upon
                which one is currently activated and then the duty cycle is
adjusted
                based on the output from the ADC
Input         : N/A
Output        : N/A
***********************************************************************************
**/

void timer0interrupt() __interrupt(1) // ISR for the Timer0 mode1
{//Sensor value 0 indicates flex sensor
    if(interrupt_counter==20 && sensor==FLEX)//Execute the ISR code once every
second since the timer is set for 50ms so 20 iterations equal 1 second
    {
        interrupt_counter=0;//Variable used to track the number of times the ISR is
entered
        ADDRESS_A=0;//Set the address pins of the ADC to 000 for selecting input
channel 0
        ADDRESS_B=0;
        ADDRESS_C=0;

        TR0=0;
        TL0=0xFC; //Load 0x4BFC to obtain timing of 50 ms.
        TH0=0x4B;

        TR0=1;
        ALE=0;//Latch the address by providing low to high pulse
        ADCClk();
        FastDelay(5);
        ALE=1;
        ADCClk();
        FastDelay(5);

        ADCClk();
        SC=0;//Start conversion by providing low to high pulse to the SC pin
        ADCClk();
        FastDelay(5);
        ADCClk();
        SC=1;
        ADCClk();
        FastDelay(5);

        ALE=0;
        ADCClk();
        FastDelay(5);
        SC=0;
        ADCClk();

        while (EOC == 0)//Poll for EOC to go high indicating that conversion is
complete
        {
        ADCClk();
        }

        OE=1;
        ADCClk();
        FastDelay(5);

        ADCdata=P0;//Store the output in a variable
        ADCClk();
        OE=0;
        ADCClk();

        ADCToMotorSpeed();//Update the duty cycle of the motor based on the current
reading
    }
    else if(interrupt_counter==20&&sensor==ACCEL)
    {
        interrupt_counter=0;//Variable used to track the number of times the ISR is
entered
        ADDRESS_A=1;//Set the address pins of the ADC to 001 for selecting input
channel 1
        ADDRESS_B=0;
        ADDRESS_C=0;
        TR0=0;
        TL0=0xFC; //Load 0x4BFC to obtain timing of 50 ms.
        TH0=0x4B;
        TR0=1;
        ALE=0;//Latch the address by providing low to high pulse
        ADCClk();
        FastDelay(5);
        ALE=1;
        ADCClk();
        FastDelay(5);

        ADCClk();
        SC=0;//Start conversion by providing low to high pulse to the SC pin
        ADCClk();
        FastDelay(5);
        ADCClk();
        SC=1;
        ADCClk();
        FastDelay(5);

        ALE=0;
        ADCClk();
        FastDelay(5);
        SC=0;
        ADCClk();

        while (EOC == 0)//Poll for EOC to go high indicating that conversion is
complete
```

```
        {
            ADCClk();
        }

        OE=1;
        ADCClk();
        FastDelay(5);

        ADCdata=P0;
        ADCClk();
        OE=0;
        ADCClk();

        ACCLToMotorSpeed();//Update the duty cycle of motr based on accelerometer
output


    }
    interrupt_counter++;//Increment the counter to track number of times entered
into the ISR
}

/*****************************************************************************
***
Description   : This is the external hardware interrupt ISR which is triggered
every
                time a push button switch is pressed. It allows the user to shift
                control between the flex sensor and the accelerometer
Input         : N/A
Output        : N/A
******************************************************************************
**/

void external_interrupt()interrupt 0//ISR for external hardware interrupt 0
{
    sensor=!(sensor);//Toggle the value of sensor
    printf_tiny("\n\r Sensor source changed");
}

//Main Program
void main()
{

    ADCInit();//Initialize the ADC
    sensor=FLEX;//Set default sensor value to flex sensor
    duty_cycle=0x99;//Set default duty cycle to 40%
    PWMON();//Turn on the motor
    interrupt_counter=0;//Initialize the counter to zero
    TCON|=0x01;//Enabled falling edge triggered interrupts
    IEN0=0x83;//Enable timer 0 interrupt and external hardware 0 interrupt

    timer0();//Start timer 0
    while(1);



}
```

```
/*

************************************************************************************
 * Description    : This file contains functions to generate delays that are needed
for
                  ADC
 * Author         : Subhradeep
 * Date           : 22 November 2016
 * File name      : Delays.c

************************************************************************************
 */


/***********************************************************************************
***
Description    : This function is used to generate a long delay

Input          : A number indicating the amount of delay required
Output         : N/A
************************************************************************************
**/

 void Delay(int a)
 {

     while(a!=0)
     {
     int j;
     int i;
     for(i=0;i<=a;i++)
     {
         for(j=0;j<100;j++)
         {

         }
     }
     a--;

     }

 }

/***********************************************************************************
***
Description    : This function is used to generate a very short delay
```

```
Input         :  A number indicating the amount of delay required
Output        : N/A
***********************************************************************************
**/

void FastDelay(uint8_t var)
{
    while(var!=0)
        var--;
}
```

```
/*

***********************************************************************************
 * Description  : Header file for Delays.c
 * Author       : Subhradeep
 * Date         : 22 November 2016
 * File name    : Delays.h

***********************************************************************************
 */

#ifndef _DELAYS_H
#define _DELAYS_H
#include <at89c51ed2.h>
#include <mcs51reg.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

/** Function prototypes */
void Delay(int a);
void FastDelay(uint8_t var)

#endif
```

8.4 Appendix - Data Sheets and Application Notes:

These are submitted as separate pdf files in the datasheets folder.