A
Project Report On

Serial Communication using ATmega 32 μc & GSM Module

Mini-Project (2151102)

BACHELOR OF ENGINEERING
in
ELECTRONICS AND COMMUNICATION ENGINEERING

By

Chaitanya Tejaswi 140080111013

Under The Guidance of
Prof Anish Vahora
Professor, ET Department.

ELECTRONICS & TELECOMMUNICATION ENGINEERING
DEPARTMENT
BVM ENGINEERING COLLEGE
GUJARAT TECHNOLOGICAL UNIVERSITY
VALLABH VIDYANAGAR-388120
Academic Year- 2016-17

CERTIFICATE

This is to certify that the project report entitled **"Serial Communication using ATmega 32 µc & GSM Module",** submitted by **Chaitanya Tejaswi** (140080111013) in the subject of the ***Mini Project (2151102)*** for the Bachelor of Engineering in Electronics and Communication of BVM Engineering College, Vallabh Vidyanagar, Gujarat Technological University, is the record of work carried out by them under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Under The Guidance Of
Prof Anish Vahora
Professor, ET Department.

ELECTRONICS & TELECOMMUNICATION ENGINEERING
DEPARTMENT
BVM ENGINEERING COLLEGE
GUJARAT TECHNOLOGICAL UNIVERSITY
VALLABH VIDYANAGAR-388120
Academic Year- 2016-17

INDEX

Introduction

What is a GSM Modem?

1. GSM stands for Global System for Mobile Communications.It is a standard set developed by the EuropeanTelecommunications Standards Institute (ETSI) to describe protocols for second generation (2G) digital cellular networks used by mobilephones.
2. A Modem is a device which modulates and demodulates signals as required to meet the communication requirements.It modulates an analog carrier signal to encode digital information, and also demodulates such a carrier signal to decode the transmitted information.
3. A GSM Modem is a device that modulates and demodulates the GSM signals and in this particular case 2G signals. The modem we are using is SIMCOM SIM300. It is a Tri-band GSM/GPRS Modem as it can detect and operate at three frequencies (EGSM 900 MHz, DCS 1800 MHz and PCS1900 Mhz). Default operating frequencies are EGSM 900MHz and DCS 1800MHz.

GSM Module – SIM300

Sim300 is a widely used in many projects and hence many variants of development boards for this have been developed. These development boards are equipped with various features to make it easy to communicate with the SIM300 module. Some boards provide only TTL interface while some boards include an RS232 interface and some others include an USB interface. If your PC has a serial port(DB9) you can buy a GSM Modem that has both TTL and RS232 interfacings in economy.

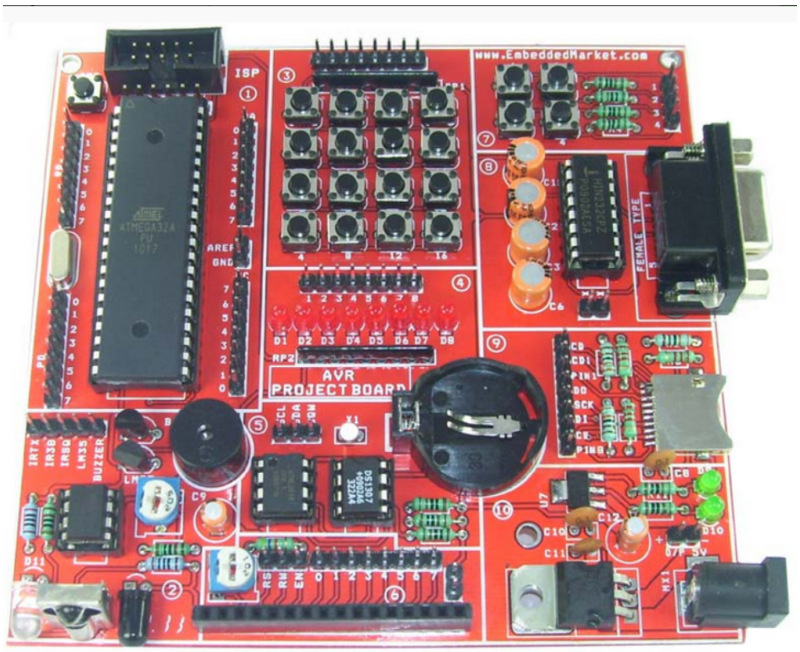
Sim300 GSM module used here consists of a TTL interface and an RS232 interface. The TTL interface allows us to directly interface with a microcontroller while the RS232 interface includes a MAX232 IC to enable communication with the PC. It also consists of a buzzer, antenna and SIM slot. Sim300 in this application is used as a DCE (Data Circuit-terminating Equipment) and PC as a DTE (Data Terminal Equipment).

AT Commands

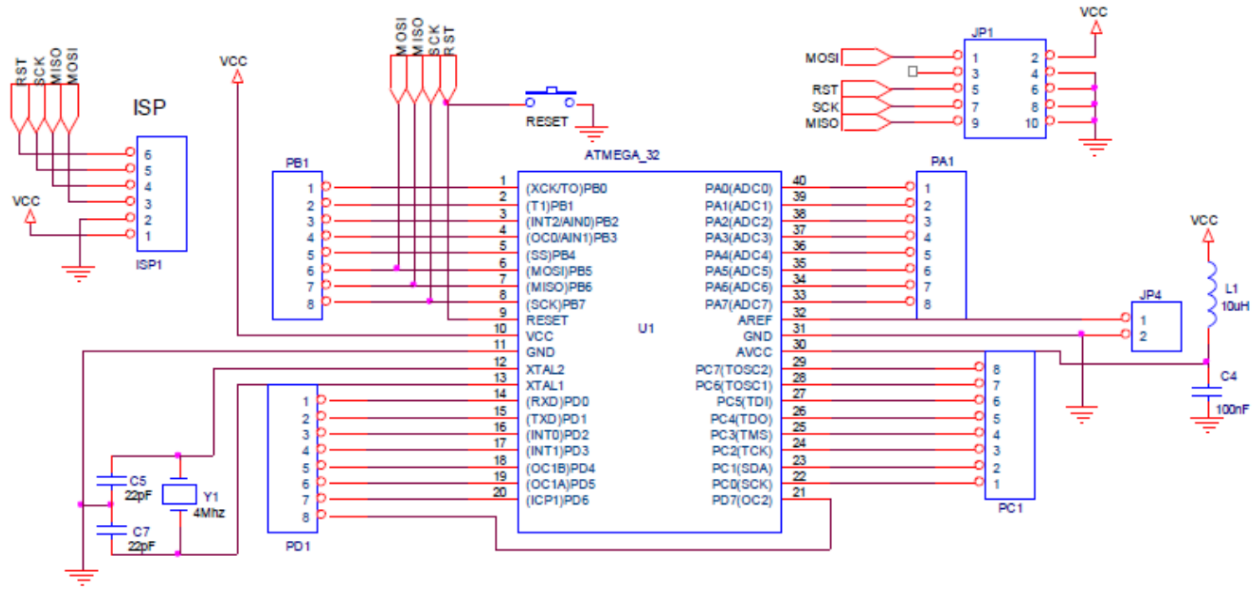
Sim300 GSM Module can be used to send and receive SMS connecting it to a PC when a SIM is inserted. The GSM Modem can be sent commands to send or receive SMS from the PC through a com port (serial port or an usb). These commands are called as AT commands. Through AT commands we can perform several actions like sending and receiving SMS, MMS, etc. Sim300 has an RS232 interface and this can be used to communicate with the PC. Sim300 usually operates at a baudrate of 9600, with 1 stopbits, No parity, No Hardware control and 8 databits. We shall see at some of the AT Commands necessary for sending and receiving SMS.

Command	Description
AT	It is the Prefix of every command sent to the modem. It is also used to test the condition of the modem. The GSM Modem responds with an \r\nOK\r\n or an \r\nERROR\r\n in case of error. where \r is the carriage return character and \n is a new line feed character).
AT+CSMINS?	Command to check if the Modem has a sim inserted in it. It checks if the sim Is inserted.
AT+CREG?	Command to check if the sim is registered with the network. It checks if the sim is registered and returns the status.
ATE1	Command to turn on the ECHO. The GSM Modem continnously echo's back the every byte of data sent to modem until a carriage return charater is sensed. It processes the command after a carriage return character is detected. It is usually better to turn off echo to reduce traffic. In this case ECHO is turned on to see how commands are sent and how they are processed.
AT+CMGF=1	Command to set the communication to TextMode. By default the communication is in the PDU mode.
AT+CMGR=1	Command to read an SMS at the index one.Generally the index depends upon the how many number of SMS that a sim can store. SIM Memory is the only memory available when GSM Modem is used and hence the number of SMS's stored depends on the SIM. It is usually 20. Any message received is arranged in the order of arrival at specific indices.
AT+CMGD=1	Command to delete the SMS at the index 1.
AT+CMGS	Command to send SMS from the GSM Modem.

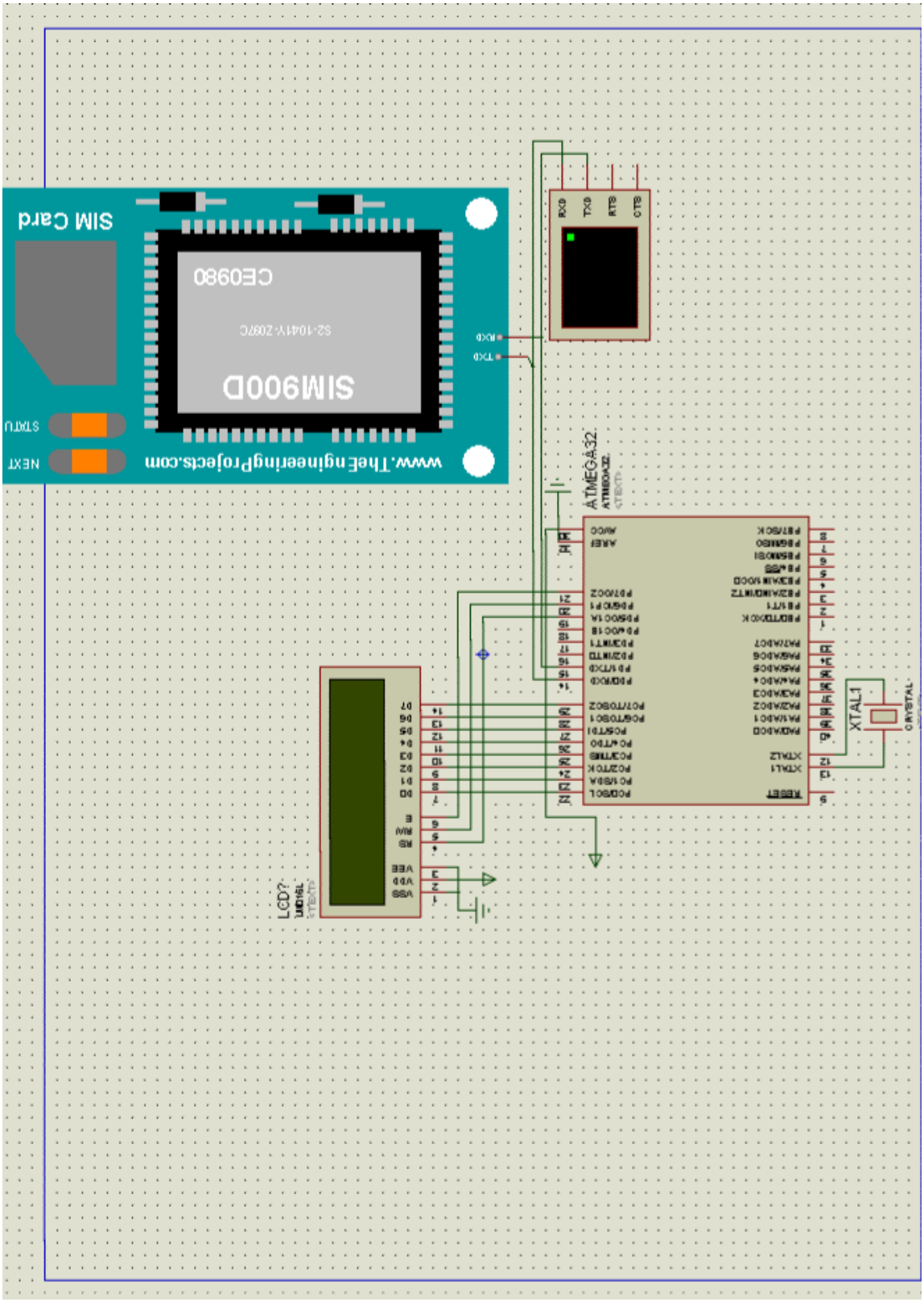
AVR Trainer Circuit



ATmega32 – All Port Pins open, ISP Port & Crystal



Schematic


SIM Card
SIM900D
www.TheEngineeringProjects.com
ATMEGA32
XTAL1
LCD?
RXD
TXD
RTS
CTS

Using the UART of Atmega32 to communicate with SIM300

Atmega32 has three types of serial communication peripherals and they are

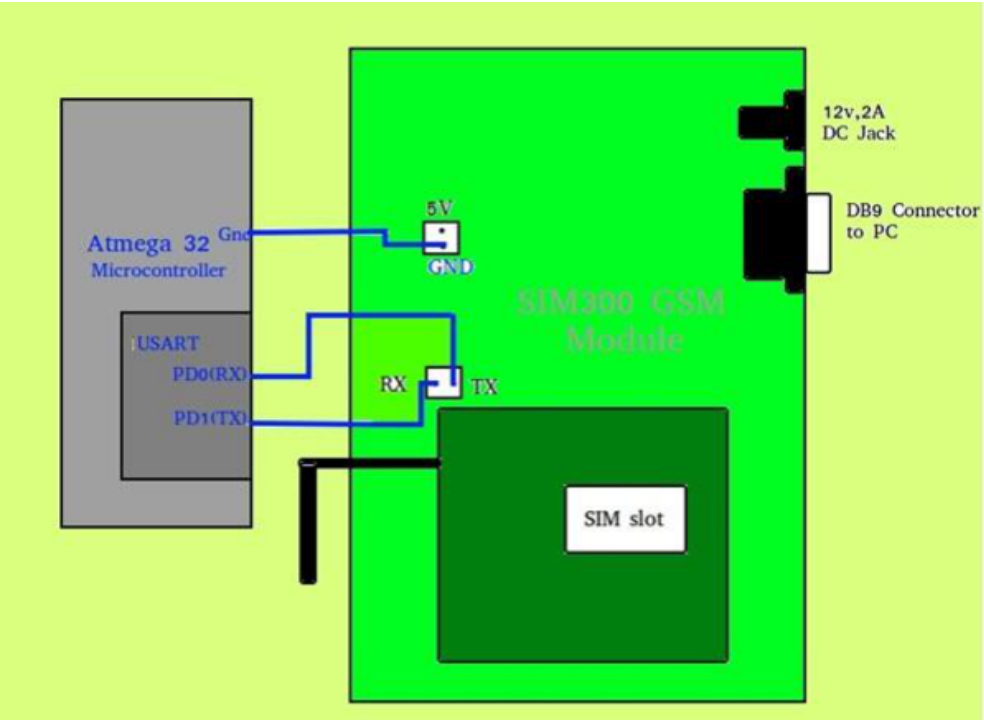
1. USART (Universal Synchronous/Asynchronous Receive and Transmit)
2. TWI (Two-Wire serial Interface) and
3. SPI (Serial Peripheral Interface)

Atmega32 has an USART(Universal Synchronous and Asynchronous Receive and Transmit) peripheral which enables us to perform serial communication either synchronously or asynchronously. Here asynchronous mode is preferred at a normal speed. The Pins Rx and Tx of the GSM Modem are connected to the Tx and Rx pins of Microcontroller. USART of the Microcontroller is configured to work at the baud rate of 9600, with 8 Data bits, 1 stop bit and Parity None.

Using Watchdog Timer to prevent infinite loops

A watchdogtimer is used to prevent the microcontroller from falling in to infinite loops. A watchdogtimer when set, counts down the specified and when the timer expires it resets the microcontroller. In Atmega32A a watchdog timer with a maximum time of 2 seconds is available.

While reading the data from GSM Modem via UART some times it is possible that we may not receive the expected bytes from the Modem. In such a case the the microcontroller keeps waiting for those expected bytes and keeps on waiting thus falling in to an infinite loop. To prevent such a condition a watchdogtimer is set with a time of 2 seconds(approximately). If expected bytes are received from the GSM Modem then immediately the timer is reset and the watchdogtimer is disabled. if the expected bytes are not received from the Modem then the timer expires and a reset occurs.



Algorithm

The GSM Modem is constantly checked for any new messages to respond and hence it follows the following algorithm:

Initialising the GSM Modem

1. Send AT command and wait for the response.if the response is correct then goto step 2,else the watchdogtimer causes a reset.

2. Send command to check if the sim is inserted in to the modem.if the response is correct then goto step 3,else the watchdog timer expires and causes a reset.

3. Send command to check if the sim is registered to the network.if the response is correct then goto next step,else the watchdog timer expires and causes a reset.

4. Send command to turnon the echo and wait for the response.

5. Send command to communicate in the textmode and wait for the response.

6. Now start reading the messages.

Reading the SMS

1. Send command to read the SMS at index 1.

2. Read the response

   (a) If the response is OK then go to step 3

   (b) Else if the response is "+CMGR" then start reading the number and the message.

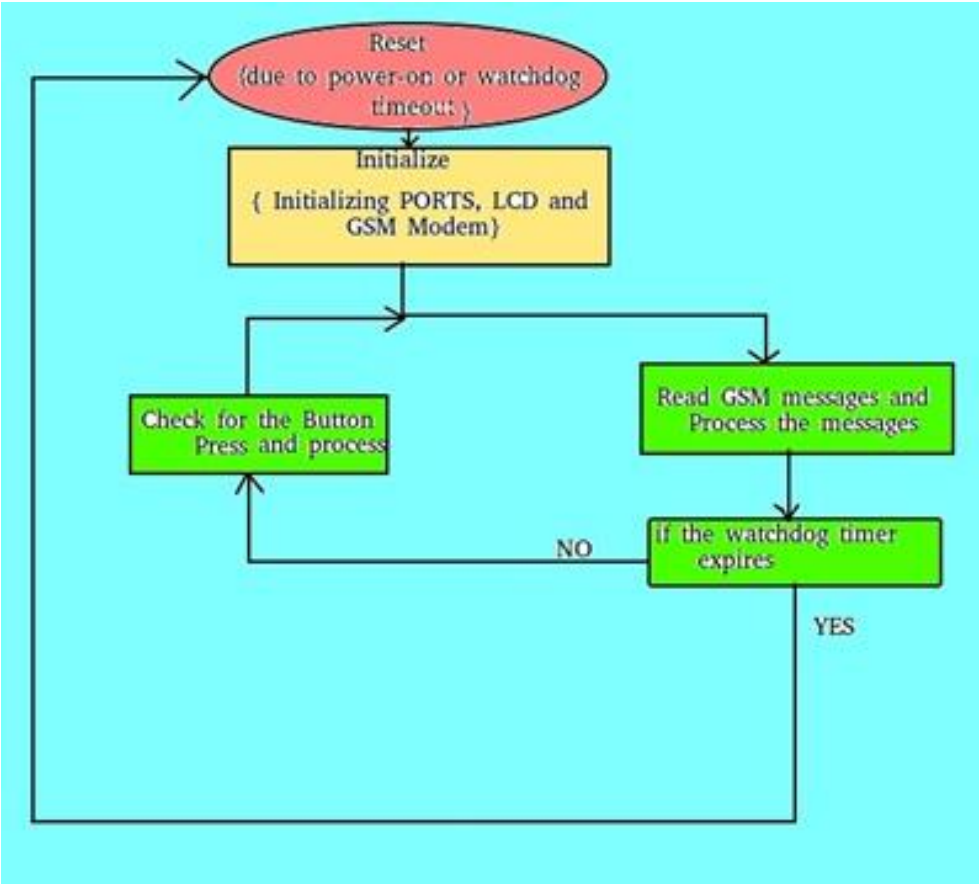
      1. If number is the registered number then process the message.

      2. Else discard the message.

      3. Delete the message at index

3. Wait for about 1000ms for the modem to rest.

Overall Process



Source Code

1. Main File
2. LCD File
3. GSM File
4. Button File

Main File (main.c)

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>

#include "GSM.h"
#include "button.h"
#include "lcd.h"



void gsm_read_all(void);

void gsm_read_all()
{
	i=1;
	while(i<2)
	{
		gsm_read_byindex(i);
		i++;
	}
}

int main(void)
{

	button_init();
	lcd_init();
	clearscreen();
	location(1,1);
	sendstring("Initializing... ");



	gsm_init();

	check_buttons();

	while(1)
	{
		check_buttons();
		gsm_read_all();
	}

}
```

LCD File (lcd.h)

```
#ifndef lcd
#define lcd

#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>

#define DataPORT            PORTC
#define DataPORTdir   DDRC
#define ControlPORT   PORTD
#define ControlPORTdir      DDRD
#define EN 7
#define RW 6
#define RS 5

char firstColumnPositionsForLCD[4] = {0, 64, 20, 84};

void check_busy(void);
void LCD_wait(void);
void sendcommand(unsigned char command);
void sendchar(unsigned char character);
void sendstring(char *StringOfCharacters);
void location(uint8_t x, uint8_t y);
void lcd_init(void);
void clearscreen(void);



void check_busy()
{
      DataPORTdir = 0x00;
      ControlPORT |= 1<<RW;
      ControlPORT &= ~(1<<RS);

      while (DataPORT >= 0x80)
      {
      LCD_wait();
      }

      DataPORTdir = 0xFF;
}

void LCD_wait()
{
      ControlPORT |= 1<<EN;
      ControlPORT &= ~(1<<EN);
}

void sendcommand(unsigned char command)
{
```

```
	check_busy();
	DataPORT = command;
	ControlPORT &= ~ ((1<<RW)|(1<<RS));
	LCD_wait();
	_delay_ms(5);
	DataPORT = 0;
}

void sendchar(unsigned char character)
{
	check_busy();
	DataPORT = character;
	ControlPORT &= ~ (1<<RW);
	ControlPORT |= 1<<RS;
	LCD_wait();
	_delay_us(200);
	DataPORT = 0x00;
}

void sendstring(char *StringOfCharacters)
{
	while(*StringOfCharacters > 0)
	{
		sendchar(*StringOfCharacters++);
	}
}

void sendint(int number)
{
char string[3];
itoa(number,string,10);
sendstring(string);
}

void location(uint8_t x, uint8_t y)
{
	sendcommand(0x80 + firstColumnPositionsForLCD[y-1] + (x-1));
}

void lcd_init()
{
	ControlPORTdir |= 1<<EN | 1<<RW | 1<<RS;
	_delay_ms(15);

	sendcommand(0x01);
	_delay_ms(2);
	sendcommand(0x38);
	_delay_us(50);
	sendcommand(0b00001100);
	_delay_us(50);
}
```

```
void clearscreen(void)
{
      sendcommand(0x01);
}


#endif
```

GSM File (GSM.h)

```
#ifndef GSM
#define GSM

#include <avr/io.h>
#include <util/delay.h>
#include <string.h>
#include <avr/wdt.h>
#include <avr/interrupt.h>


#include "lcd.h"

/******************** SOME VARIABLES ***********************/

char operator_phone[] = "+919427380794";
char status[50]="";
char msg[7];
char number[14];


int i,j;

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

/******************** UART FUNCTIONS ***********************/

void UART_Init( unsigned int baud );
void UART_Transmit_char( unsigned char data );
unsigned char UART_Receive( void );
void UART_Transmit_string( char *string );

void UART_Init( unsigned int baud )
{
   /* Set baud rate */
   UBRRH = (unsigned char)(baud>>8);
   UBRRL = (unsigned char)baud;

//For BR=9600, make UBRRH=0; UBRRL=0x19;

   /* Enable receiver and transmitter */
   UCSRB = (1<<RXEN)|(1<<TXEN);
   /* Set frame format: 8data, 1stop bit */
   UCSRC = (1<<URSEL)|(0<<USBS)|(3<<UCSZ0);
}

void UART_Transmit_char( unsigned char data )
{
   /* Wait for empty transmit buffer */
   while ( !( UCSRA & (1<<UDRE)) )
   ;
```

```
   /* Put data into buffer, sends the data */
   UDR = data;
}

unsigned char UART_Receive( void )
{
   /* Wait for data to be received */
   while ( !(UCSRA & (1<<RXC)) )
   ;
   /* Get and return received data from buffer */
   return UDR;
}
void UART_Transmit_string( char string[] )
{
   int i=0;
   while ( string[i] > 0)
   UART_Transmit_char(string[i++]);
}

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



void gsm_init(void);
void gsm_read_byindex(int i);
void gsm_send(char *number,char *string);
void gsm_delete(void);
void gsm_delete_byindex(int i);
void gsm_waitfor(char c);

void check_ports(void);

char temp;

void gsm_waitfor(char c)
{
      //enabling watchdogtimer with a time of 2.1secs
      wdt_enable(7);
      //waiting for the byte to be received
      while(UART_Receive()!= c);
      //resetting watchdogtimer and turning off the watchdogtimer
      wdt_reset();
      wdt_disable();
}



void gsm_init()
{

      UART_Init(71);
      location(1,2);
```

```
sendstring(" Testing Modem  ");
_delay_ms(500);
UART_Transmit_string("AT\r");
gsm_waitfor('O');
gsm_waitfor('K');
location(1,2);
sendstring("   Modem : OK   ");
_delay_ms(1000);

INS:


   location(1,2);
      sendstring(" Checking SIM   ");
      _delay_ms(500);
      UART_Transmit_string("AT+CSMINS?\r");
   gsm_waitfor( '\n');
   gsm_waitfor(',');
   if(UART_Receive() == '2')
      {
            location(1,2);
            sendstring("  SIM NOTFOUND  ");
            _delay_ms(1000);
            goto INS;
      }
      else if(UART_Receive() == '1');
      gsm_waitfor( 'K');
      gsm_waitfor( '\n');
      location(1,2);
      sendstring("   SIM FOUND    ");
      _delay_ms(1000);

REG:


   location(1,2);
      sendstring(" Network Status ");
      _delay_ms(500);
      UART_Transmit_string("AT+CREG?\r");
   gsm_waitfor( '\n');
   gsm_waitfor(',');
   if(UART_Receive() == '2')
      {
            location(1,2);
            sendstring("Network NotFound");
            _delay_ms(1000);
            goto REG;
      }
      else if(UART_Receive() == '1');
      gsm_waitfor( 'K');
      gsm_waitfor( '\n');
      location(1,2);
```

```
            sendstring(" Network Found  ");
            _delay_ms(1000);



      location(1,2);
      sendstring(" Turn on Echo  ");
      _delay_ms(500);
      UART_Transmit_string("ATE1\r");
      gsm_waitfor('O');
      gsm_waitfor('K');
      location(1,2);
      sendstring(" Echo turned on ");
      _delay_ms(1000);

      UART_Transmit_string("AT+CMGF=1\r");
      location(1,2);
      sendstring("Setting Textmode");
      gsm_waitfor('O');
      gsm_waitfor('K');
      location(1,2);
      sendstring("  Textmode set  ");
      _delay_ms(1000);

}

void gsm_read_byindex(int index)
{
      int k;
      char string[2];
      itoa(index,string,10);
      clearscreen();
      location(1,1);
      sendstring("Reading Messages");
      location(1,2);
      j++;
      if(j<16)
      {
            for(int a=0;a<j;a++)
            {
                  sendstring(".");
            }
      }
      else if(j>=16)
      {
            j=1;
            for(int a=0;a<j;a++)
            {
                  sendstring(".");
            }
```

```
	}
	UART_Transmit_string("AT+CMGR=");
	UART_Transmit_string(string);
	UART_Transmit_string("\r");

	gsm_waitfor('\r');
	gsm_waitfor('\n');
	     if(UART_Receive()=='+')
	     {
	          gsm_waitfor('M');
	          if(UART_Receive()=='G')
	          {
	               gsm_waitfor('A');
	               gsm_waitfor(',');
	               gsm_waitfor('"');


	               for(k=0;k<13;k++)
	               number[k] = UART_Receive();


	               gsm_waitfor(',');
	               gsm_waitfor(',');
	               gsm_waitfor('+');
	               gsm_waitfor('\n');

	               for(k=0;k<6;k++)
	               msg[k]=UART_Receive();

	               gsm_waitfor('K');
	               gsm_waitfor('\n');

	               _delay_ms(300);
	               clearscreen();
	               location(1,1);
	               sendstring("ph:");
	               sendstring(number);
	               location(1,2);
	               sendstring("Message:");
	               sendstring(msg);
	               _delay_ms(2000);

	               if(strcmp(number,operator_phone))
	               {
	                    gsm_send(number,"You are not authorized to
send this message");
	               }
	               else if(!(strcmp(number,operator_phone)))
	               {

	                    if(!strcmp(msg,"1 on  "))
```

```
{
      PORTA |= (1<<PA2);
}
if(!strcmp(msg,"2 on  "))
{
      PORTA |= (1<<PA3);
}
if(!strcmp(msg,"3 on  "))
{
      PORTA |= (1<<PA4);
}
if(!strcmp(msg,"4 on  "))
{
      PORTA |= (1<<PA5);
}
if(!strcmp(msg,"5 on  "))
{
      PORTA |= (1<<PA6);
}
if(!strcmp(msg,"6 on  "))
{
      PORTA |= (1<<PA7);
}

if(!strcmp(msg,"1 off "))
{
      PORTA &= ~(1<<PA2);
}
if(!strcmp(msg,"2 off "))
{
      PORTA &= ~(1<<PA3);
}
if(!strcmp(msg,"3 off "))
{
      PORTA &= ~(1<<PA4);
}
if(!strcmp(msg,"4 off "))
{
      PORTA &= ~(1<<PA5);
}
if(!strcmp(msg,"5 off "))
{
      PORTA &= ~ (1<<PA6);
}
if(!strcmp(msg,"6 off "))
{
      PORTA &= ~(1<<PA7);
}
if(!strcmp(msg,"report"))
{
      check_ports();
}
```

```
                    }

                    gsm_delete_byindex(index);
                }
            }

    _delay_ms(1000);
}

void gsm_send(char *number,char *string)
{
    UART_Transmit_string("AT+CMGS=\"");
    UART_Transmit_string(number);
    UART_Transmit_string("\"\r");

    gsm_waitfor('>');

    UART_Transmit_string(string);
    UART_Transmit_char(0x1A);

    while(UART_Receive()!= '+');
    while(UART_Receive()!= ' ');
    while(UART_Receive()!= '\n');

    _delay_ms(1000);

}



void gsm_delete()
{
    UART_Transmit_string("AT+CMGD=1\r");
    gsm_waitfor('K');
    gsm_waitfor('\n');
    _delay_ms(500);

}

void gsm_delete_byindex(int i)
{
    UART_Transmit_string("AT+CMGD=");
    char string[2];
    itoa(i,string,10);
    UART_Transmit_string(string);
    UART_Transmit_string("\r");
    gsm_waitfor('K');
    gsm_waitfor('\n');
    _delay_ms(500);

}
```

```
void check_ports()
{
      if(bit_is_set(PINA,2))
            {
                  strcat(status,"1_on ");
            }
            else
            {
                  strcat(status,"1_off ");
            }
            if(bit_is_set(PINA,3))
            {
                  strcat(status,"2_on ");
            }
            else
            {
                  strcat(status,"2_off ");
            }
            if(bit_is_set(PINA,4))
            {
                  strcat(status,"3_on ");
            }
            else
            {
                  strcat(status,"3_off ");
            }
            if(bit_is_set(PINA,5))
            {
                  strcat(status,"4_on ");
            }
            else
            {
                  strcat(status,"4_off ");
            }
            if(bit_is_set(PINA,6))
            {
                  strcat(status,"5_on ");
            }
            else
            {
                  strcat(status,"5_off ");
            }
            if(bit_is_set(PINA,7))
            {
                  strcat(status,"6_on ");
            }
            else
            {
                  strcat(status,"6_off ");
            }
```

```
            gsm_send(operator_phone,status);
            status[0] = '\0';
            clearscreen();
            location(1,1);
}

#endif
```

Button File (button.h)

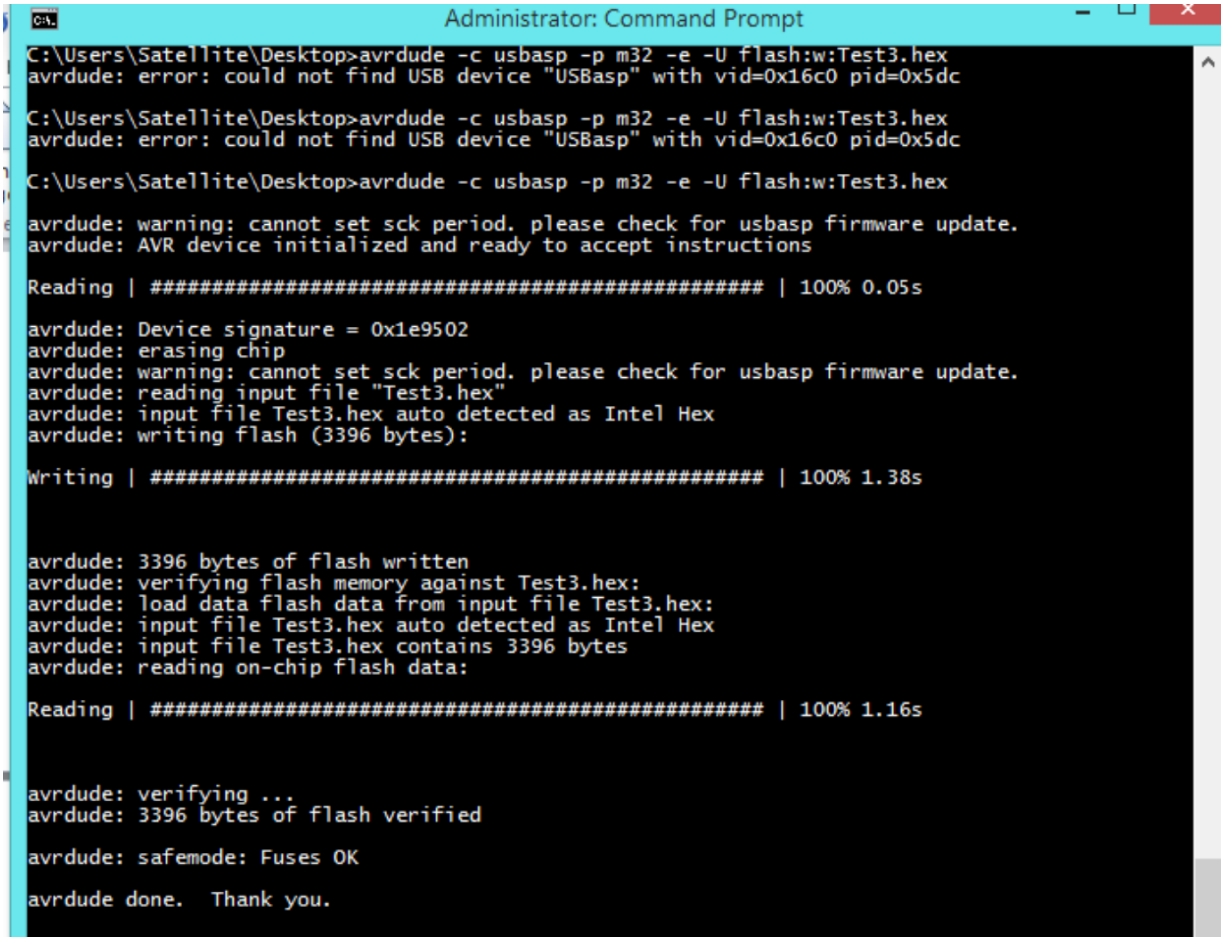
```
#ifndef button
#define button

#include "GSM.h"
#include "lcd.h"


void button_init(void);
void check_buttons(void);


void button_init()
{

      //setting PortA pin 0 high and
      PORTA |=  (1<<PA0);
      //setting it as an input
      DDRA  &= ~(1<<DDA0);

      //setting PortA pin 1 high and
      PORTA |=  (1<<PA1);
      //setting it as an input
      DDRA  &= ~(1<<DDA1);

      //Setting other pins on portA as output
      DDRA  |= (1<<PA2);
      DDRA  |= (1<<PA3);
      DDRA  |= (1<<PA4);
      DDRA  |= (1<<PA5);
      DDRA  |= (1<<PA6);
      DDRA  |= (1<<PA7);

}

void check_buttons()
{
      if(bit_is_clear(PINA,0))
      {
            clearscreen();
            location(1,1);
            sendstring("Deleting all... ");
            for(i=1;i<=20;i++)
            {
                  gsm_delete_byindex(i);
            }
      }
      if(bit_is_clear(PINA,1))
      {
            clearscreen();
            location(1,1);
```

```
            sendstring(" Sending status  ");
            location(1,2);
            sendstring("  to Operator...");

            check_ports();
            _delay_ms(1000);

      }


}



#endif
```

Practical Considerations:

1. Baud Rate is fixed to 9600, so refer comment in GSM file.
2. Button is not necessary for communication process if watchdog is set.
3. Programming the AVR Trainer kit:

   Programming is done using AVRDude command line utility.



Following is a screenshot of Trainer Kit info using AVRDude