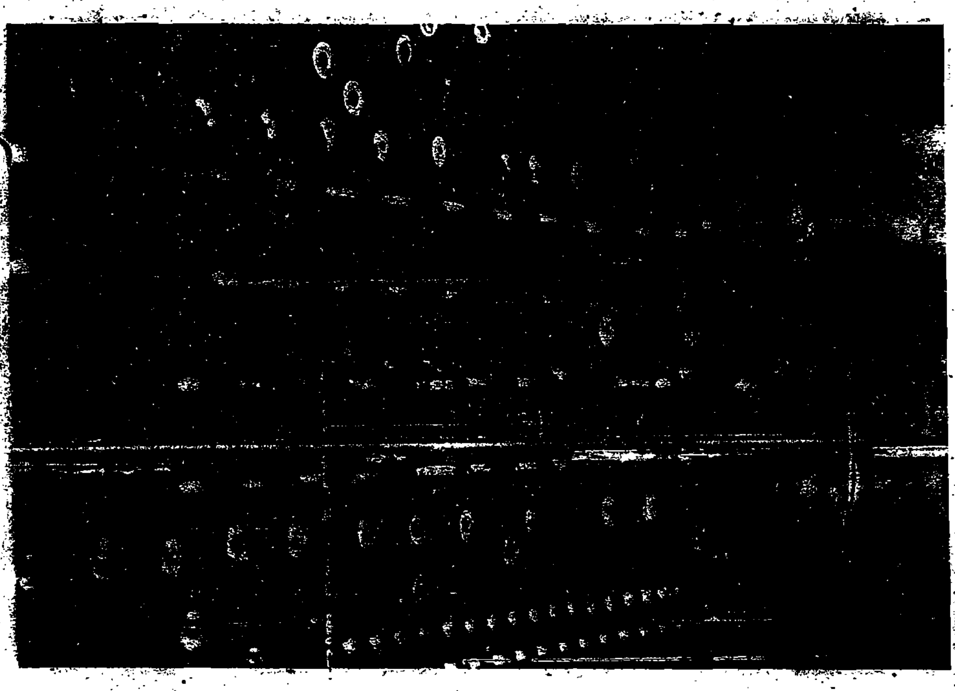
EMBEDDED SYSTEMS IN COMMUNICATIONS

Understand the generic embedded system, its types and design requirements, and how a simple SMS from a mobile can be used to control a remote appliance!



■ DR G.R. KULKARNI, PROF A.C. SUTHAR & ASHISH N. JANI

An artist named Samuel Morse was the first to use electricity to send messages from one place to another. Decades have passed since then and now we don't even require a wire to communicate. Wireless communication has announced its arrival on the big stage and the world is going mobile. We want to control everything without moving an inch, and this remote control of appliances is possible through embedded systems.

Embedded systems have made their way into almost every electronic device in use today. Right from our watches to cellphones, they form an integral part of the circuit. Each of these systems is unique, and the hardware is highly specialised to the application domain.

What is an embedded system?

An embedded system is a combination of computing hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. This is in direct contrast to the personal computer in your family room.

The PC too comprises computing hardware and software, and mechanical components like disk drive. However, it is not designed to perform a specific function. Rather, it is able to do many different things. Many people use the term 'general-purpose computer' to make this distinction clear.

Usually, an embedded system is a component within some larger system. For example, modern trucks and cars contain many embedded systems. One embedded system controls the antilock breaks, another monitors the vehicle's emissions, and the third displays information on the desk board.

At the possible risk of confusing you, it is important to point out that a general-purpose computer is itself made up of numerous embedded systems. For example, our computer consists of a keyboard, mouse, video card, modem, hard drive, floppy drive and sound card—each of which is an embedded system. Each of the devices contains a processor and software, and is designed to perform a specific function. For example, the modem is designed to send and receive digital data over an analogue telephone line.

If an embedded system is designed well, the existence of the processor and software could be completely unnoticed by a user of the device. Such is the case with microwave ovens, VCRs or alarm clocks. In some cases, it could even be possible to build a device that does not contain the processor and software. This could be done by replacing the combination with a custom integrated circuit that performs the same functions in hardware.

A generic embedded system

Unlike software designed for general-purpose computers, embedded software cannot usually be run on other embedded systems without significant modifications. This is mainly because of the incredible variety in the underlined hardware.

Each embedded system is tailored specifically to the application, in order to keep the system's cost low. As a result, unnecessary circuitry is eliminated and hardware resources are shared wherever possible.

By definition, all embedded systems contain a processor and software but what other features do they have in common? Certainly, in order to have software, there must be a place to store the executable code and temporary storage for run-time data manipulation. These take the form of ROM and RAM, respectively; any embedded system will have the sum of both.

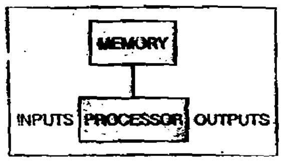


Fig. 1: Generic embedded system

If only a small amount of memory is required, it might be contained within the same chip as the processor. Otherwise, one or both types of memory will reside in external memory chips.

All the embedded systems also contain some types of inputs and outputs. For example, in a microwave oven, the inputs are the buttons on the front panel and a temperature probe, and the outputs are the human-readable displays and the microwave radiations. It is almost always the case that the output of the embedded system is a function of its input and several other factors like elapsed time, current temperature, etc. The inputs to the system usually take the form of sensors and probes, communication signals, or control knobs and buttons. The outputs are typically displays, communication signals or changes to the physical world.

Types of embedded systems

The three types of embedded systems are standalone, mobile and real-time.

Standalone embedded systems are difficult to move from one place to another. Examples are VCRs, VCD players and network appliances like Web servers.

Mobile embedded systems are free to move from one location to another, e.g., mobile handsets.

Real-time embedded systems have timing constraints. In these systems, calculations and decisions have deadlines for completion. Deadlines may be 'soft' or 'hard.' The more severe the consequences, the harder the deadline. Examples are nuclear reactors and car air-bags. At the other end of the continuum are embedded systems with soft deadlines, e.g., airline ticket reservation systems.

Design requirements of embedded systems

Processing power. The amount of processing power necessary to get the job done. A common way to compare processing power is the MIPS (millions of instructions per second) rating. If two processors have ratings of 25 MIPS and 40 MIPS, the latter is said to be the more powerful of the two.

Register width. The register width typically ranges from 8 to 64 bits. Today's general-purpose computers use 32- and 64-bit processors exclusively, but embedded systems are still commonly built with older and less costly 8- and 16-bit processors.

Memory. The amount of memory (ROM and RAM) required to hold the executable software and the data it manipulates. The amount of memory required can also affect the processor selection. In general, the register width of a processor establishes the upper limit of the amount of memory that it can access. For example, an 8-bit address register can select one of only 256 unique memory locations.

Development cost. The cost of the hardware and software design processes. This is a fixed, one-time cost, so it might be that money is no object (usually for high-volume products) or that this is the only accurate measure of system cost (in the case of a small number of units produced).

Number of units. The trade-off between production cost and development cost is affected most by the number of units expected to be produced and sold. For example, it is usually undesirable to develop your own custom hardware components for a low-volume product.

Expected lifetime. How long must the system continue to function (on an average)? A month, a year or a decade? This affects all sorts of design decisions from the selection of hardware components to how much the system may cost to develop and produce.

Reliability. How reliable must the final product be? If it is a children's toy it doesn't always have to work right, but if it's part of a space shuttle or a car it has to do what it is supposed to each and every time.

Remote control using SMS

Consider this: You leave home on an emergency call and in a hurry, forget to switch off the mains of your home.

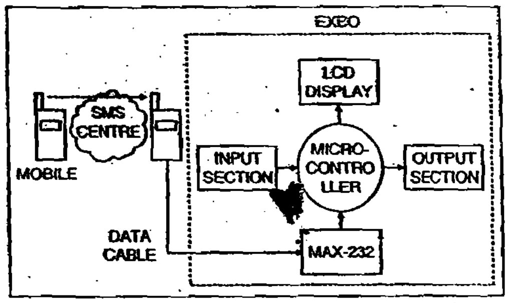


Fig. 2: Block diagram for remote control using SMS

Once you realise about it, you have already gone very far away and it is now virtually impossible for you to make a dash back home.

What if you could control your home appliances from your mobile phone? Yes, it's possible using SMS. Remote control using SMS overcomes the limitations of conventional remote control, such as physical presence, range and remote location access.

The system required for the purpose is nothing but a microcontroller-based SMS box to which a mobile (Siemens C35i) is permanently connected (see Fig. 2). Its one port is used as the output and the other for the LCD. The second mobile will be with the user who is at a remote location.

When the user sends an SMS from a remote place, it first goes to the SMS centre and then to the destination mobile. The message is in the form of the protocol description unit (PDU), which contains all the information regarding the SMS centre, sender's mobile number and message.

At the receiving end, this information is collected by a mobile, and serially passed to the microcontroller by MAX232 IC via data cable. This IC is used for serial communication as well as voltage conversion. The microcontroller decodes all the information. If the information is found to be valid, the necessary action is taken. Simultaneously, the SMS is displayed on the LCD.

LEDs used at the output indicate which system connected at the output is 'on.' Connectors are used at the outputs, to which the system to be controlled is connected through relays. On the input side, connectors are connected to sensors, while LEDs indicate which input is activated.

Hardware required

MAX232. MAX232 is designed for serial communication. It is basically a voltage converter that converts 5 volts into 12 volts and vice-versa. The MAX232, which is in the data cable, does the first conversion, while the MAX232, which is on the exbo, does the second. It has two pairs of transmitters and two pairs of receivers, of which one pair of receivers is R2IN (pin 8) and R2OUT (pin 9). The pair of transmitters is T2OUT (pin 7) and T2IN (pin 10). This IC has four 1µF capacitors connected.

Microcontroller. The micro-controller that we have used here is AT89C52. It has 256 bytes of data memory and 8 kB of flash programmable and erasable read-only memory. We are using a 11.0592MHz crystal because the Siemens C35i operates on baud rate of 19,200 and only this crystal gives zero error for this baud rate.

Port-1 is used as the input from sensors, port-2 is used as the output and port-0 is used for the LCD. Timer-1 is used for baud rate generation and timer-2 for delay. The output of MAX232 is connected to the transmit pin (txd) of the microcontroller. T1 pin is connected to RS pin, WR to RW, and RD to EN of the LCD.

Octal inverter buffer (74LS540). IC 74LS540 is an octal buffer/line driver with pin-outs on the opposite side of the package. This device can be used as a memory address drive, clock driver or bus-oriented transmitter/receiver. It is especially useful as an output port for the microcontroller, allowing ease of layout and greater PC board density.

In our exbo, we have connected LEDs to the microcontroller's output but when the output is connected to connectors through relays, buffers are necessary for increasing the current capacity.

Mobile. Any mobile with serial communication pins—one for transmission and the other for reception—can be used in the project. We have used Siemens C35i in our project as it is affordable and PC-compatible, has built-in modem and baud rate of 19,200, and its data cable is cheaper.

Pin-outs for Siemens C35i

GND	5-TX	9-GNDMIC
SB	6-RX	10- HF-MIC
POWER	7-CLOCK	11-AUDIO
NC	8-DATA	12- GND AUDIO

Acknowledgements for Normal Data Communication

Response	Numeric	Meaning
Ok	0	Command executed, no errors
Ring	2	Ring detected
No carrier	3	Link not established or disconnected
Error	4	Invalid command or command line too long
No dial tone	6	No dial tone, dialing impossible, wrong mode
Busy	7	Remote station busy

AT command set

Similar to the modems, GSM cellphones accept the AT command set only. The European Telecommunications Standard Institute (ETSI) specifies this command set.

According to the guidelines, commands should begin with the character string 'AT' and end with '<CR>.' The input of a command is acknowledged by the display 'OK' or 'ERROR.' A command currently in process is interrupted by each additional character entered. This means that the next command should not be entered until we receive the acknowledgement. Otherwise, the current command will be interrupted. The commands supported are listed below:

Mobile initialisation commands
1. ATE0: Deactivate command echo
2. ATE1: Active command echo
3. ATQ0: Display acknowledgements
4. ATQ1: Suppress acknowledgements
5. ATV0: Output acknowledgement as numbers
6. ATV1: Output acknowledgement as text
7. AT+CGMI: Issue manufacturer's ID code
8. AT+CGMM: Issue model ID code

SMS-related commands
1. AT+CMGF: SMS format
2. AT+CMGL: List SMS
3. AT+CMGR: Read in an SMS
4. AT+CMGL: Delete an SMS in the SMS memory

Software

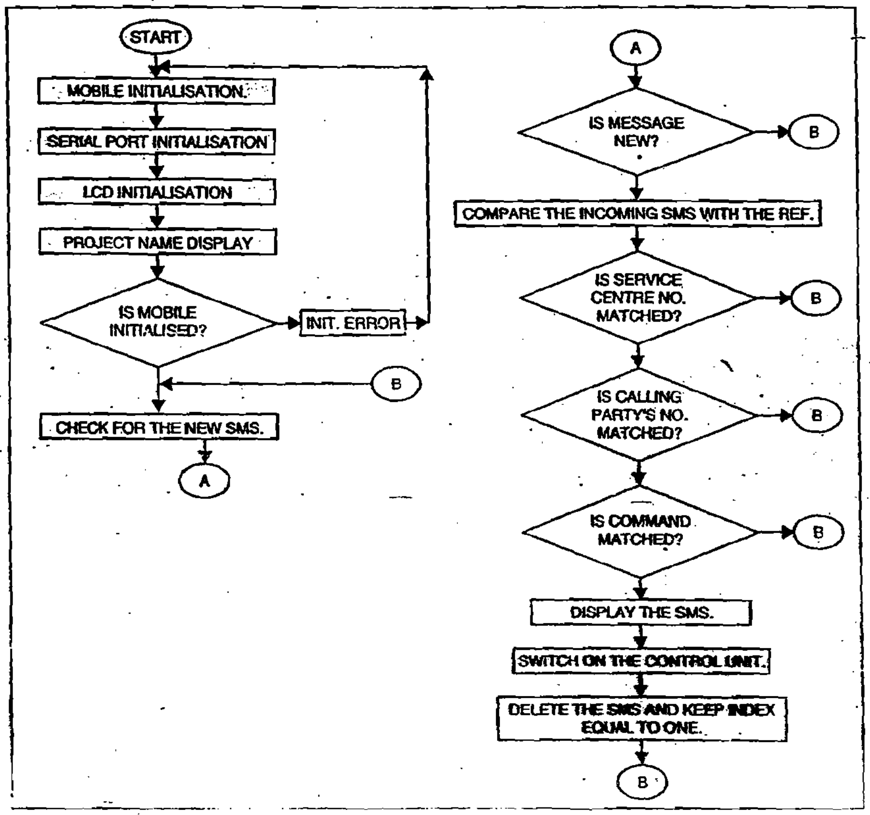
All the software is written in Assembly language using the instruction set of 8051. The sequential flow of commands is as follows:

1. Initialise the mobile by sending the required AT commands through the Hyperterminal software.

2. Continuously loop in the program to check whether any new SMS is received.

3. Read in the SMS, store it in the memory and give command to delete the SMS from the SMS memory in the mobile to keep index 1 for the next new SMS.

4. Check for the mobile number


START
MOBILE INITIALISATION.
SERIAL PORT INITIALISATION
LCD INITIALISATION
PROJECT NAME DISPLAY
IS MOBILE INITIALISED?
INIT. ERROR
B
CHECK FOR THE NEW SMS.
A
A
IS MESSAGE NEW?
B
COMPARE THE INCOMING SMS WITH THE REF.
IS SERVICE CENTRE NO. MATCHED?
B
IS CALLING PARTY'S NO. MATCHED?
B
IS COMMAND MATCHED?
B
DISPLAY THE SMS.
SWITCH ON THE CONTROL UNIT.
DELETE THE SMS AND KEEP INDEX EQUAL TO ONE.
B

Flow-chart of the program

and the incoming SMS. If both are correct, display the SMS and time of arrival on the LCD.

5. According to the SMS, the respective LEDs glow. If any error occurs in the above procedure, display the error on the LCD and loop back.

In this software, we have written small subroutines, which make programming easy. Using the look-up table, we have stored all the responses for AT commands. All the interrupts, except those for serial communication, are masked.

Possible modifications

This circuit can be made more flexible by making the following modifications:

1. Our kit accepts the incoming SMS only and only if the remote mobile is a specific mobile. This ensures privacy but certain alterations can be made in the software so that the user of any mobile handset can send the SMS and that can be accepted and decoded by the kit.

2. The Internet could be used to send the message through a PC, obviating the need for the remote mobile.

3. Some hardware can be added to make it more efficient; for example, RTC72423 can be connected. This IC will keep track of the timing of events, such as at what time an SMS has come. This time can be seen on the LCD that we are using. Also we can view the current time on the LCD. GAL16V8 can be used in place of the decoder to make the kit more flexible.

4. What we have done is a forward operation, i.e., from the remote mobile to the mobile attached to the kit. We are assuming that the kit will always work. We don't get confirmation of the success of the operation done. Certain modifications can be made in the software to allow for a reverse SMS on the remote mobile that confirms that the task has been successfully performed.

5. Reverse operation is possible. We can connect some sensors to port pins of the microcontroller. The sensors can be of any type, including fire sensors, LDRs, LVDTs and so on. When the sensors are working, these will make the respective port pins high. This will call the subroutines for that respective sensor previously fed in the memory. Depending on the response of the subroutine, necessary actions like LED 'on'/'off' and relay 'on' can be performed. Simultaneously, as mentioned earlier, a reverse SMS can be sent to inform the remote user about the action taken.

Applications

Home security. This application is based on the fact that on the input side we can connect a number of sensors. Based on the response of sensors, the output connection can be made. For example, by means of input like infrared detectors, outputs such as burglar alarm can be activated. Since reverse flow is possible simultaneously, SMS on the remote mobile can be sent about the event.

Remote factory automation. By using the relays on the output side, the remote mobile can be used as a remote control to operate any device. Thus, physical presence is not required and the automation of the entire system can be achieved. This application can be used even in labs and offices.

Remote logging system. Since PC interface is possible, any database transfer is possible.

Communication interface for controllers. Serial communication is the backbone of the entire system. This kit can be further modified to interface with other ICs.

In a nutshell

Using SMS removes the drawbacks of remote control devices. No physical presence is required at the location where you have to control the device. Also, directivity doesn't play any role in this project. SMS can be sent to any location from anywhere in the world, which makes this application versatile with no limitation of range. ●

Dr G.R. Kulkarni is principal at Rajaram Shinde College of Engineering in Ratnagiri, Ashish N. Jani, an M.Sc in IT, is a faculty member at Atmiya Institute of Technology and Science, Rajkot (Gujarat) and A.C. Suthar is a professor in Electronics & Communication Department, C.U. Shah College of Engineering and Technology, Wadhwan City, Distt Surendranagar (Gujarat).