

**Islamic University of Technology**

**Project Report**

**Course Name:** Microcontroller-Based System Design Lab

**Course Code:** EEE 4706

**Project Topic:** Bluetooth interfacing with Microcontroller

**Group No:** 03

**Group Members:**

Wasik Billah Ibn Rashid (190021335)

Sumaiya Tasnim Prapti (190021325)

A.S.M. Sarwar Jahan Shad (190021337)

Tasmiah Rahman (190021313)

Ziyaulhaq Haruna (190021345)

# Objective

This project aims to develop a Bluetooth interface with an 8051 microcontroller assembly language. This project incorporates an interface connecting the Microcontroller and Bluetooth. All additional functionalities are programmed in assembly language, and the chosen hardware for implementation is the 8051 microcontrollers.

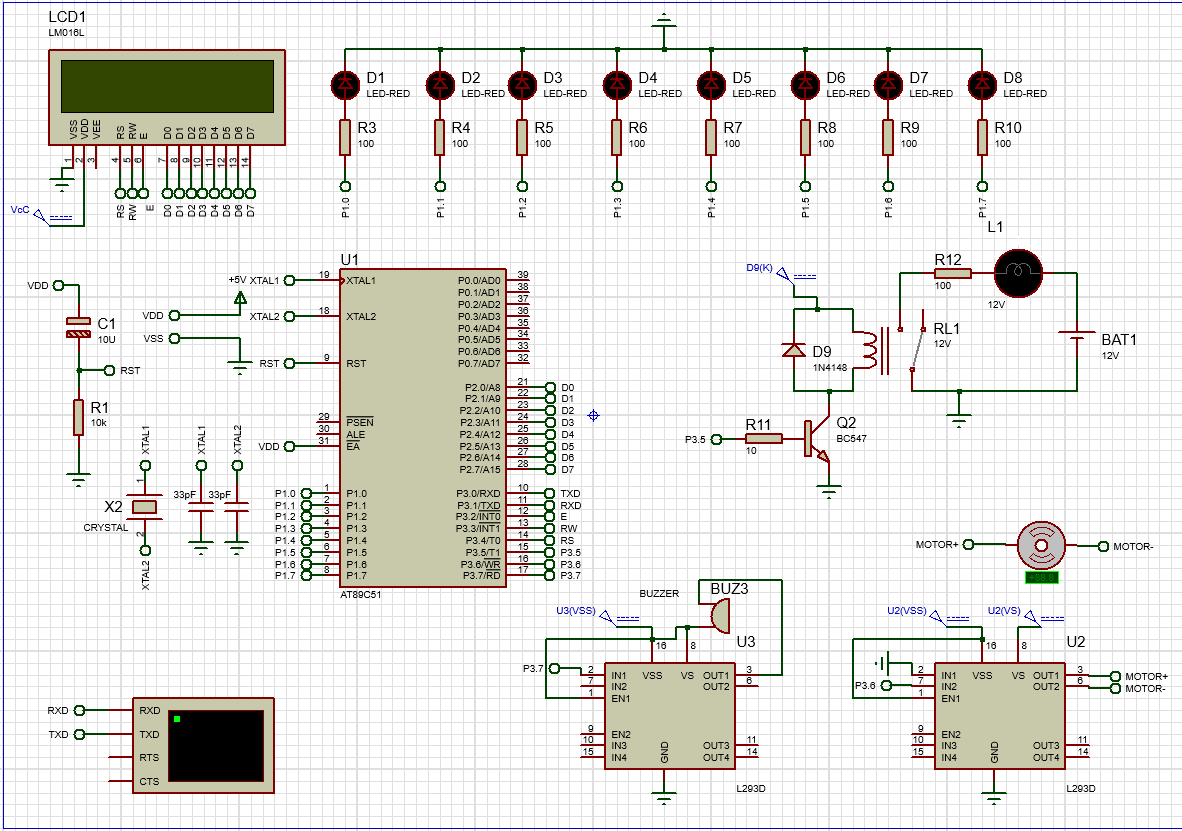
The Bluetooth modules allow for wireless data transmission and reception between two devices. Communication through smartphones to other devices is commonly used in home automation and new technologies. It is primarily applicable for controlling other devices.

The Bluetooth connection between the Microcontroller and Bluetooth module gives the power to control something remotely. Also, it helps to send data from one device to another device.

# Required Components

|  |  |
| --- | --- |
| Components | Cost (Tk) |
| Microcontroller Development Board | 7000 |
| Jumper wires |
| DC Motor |
| Relay |
| Buzzer |
| Bluetooth Module HC-05 | 450 |
| Total | 7450 |

Circuit Diagram



# Mandatory Features

1. Control all 8 LEDs via Bluetooth and send out Morse code from Bluetooth device using any one of the LEDs. Display the coded message on the LCD.

2. Use two relays to connect and control two lights using Bluetooth.

3. Send an encrypted message from a Bluetooth device and display the decrypted message on the LCD.

# Additional features

1. Display Morse code on the LCD when alphabets are sent from a smartphone.

2. Turn on the Buzzer accordingly when Morse code is sent from the smartphone to the Microcontroller.

3. Jump to the next line automatically when the 1st line of LCD is filled up.

4. Two-way communication: Retrieve data from Bluetooth and display it on the connected mobile phone. Additionally, we can transmit data from the phone to the Microcontroller and display it on the LCD monitor.

5. Motor control via Bluetooth: A motor can be controlled through a Bluetooth device. Using this Motor, we can move any other device.

6. Control the Buzzer: Wirelessly control the Buzzer.

# Working Principle

For Bluetooth interfacing, a virtual terminal was used in Proteus. At first, we initialized the serial port communication. Here, Timer mode-2 was taken to use 8-bit auto-reload. Then, the TH1 register was given a 0FDH value for 9600 baud rates as the crystal frequency is 11.0592MHz. The SBUF register is used with the Bluetooth HC05 module to facilitate serial communication. The data from the Bluetooth goes into 8051 through the RX line. In the Microcontroller, the data was kept in the SBUF register. Then, the flag register TI is raised, and the data is moved from SBUF to A register. That is how the input was taken from the HC-05 Bluetooth module. After displaying the input in the LCD, the TI flag register was cleared, and the next input could be taken from the HC-05 module. In the hardware, we used the code and burned it into the Microcontroller. We sent our data from a smart smartphone through the HC-05 Bluetooth module. The RX and TX pins of HC-05 were connected with the Microcontroller's TX and RX pins, respectively. Microcontroller for transmission and receiving. The following symbols have to be sent from the smartphone to perform the mentioned tasks:

Turn On LEDS: 1 to 8

Turn Off LEDS: ! to \*

Turn On Relay: 9

Turn Off Relay: (

Turn On Motor: +

Turn Off Motor: =

Turn on the Buzzer: "

Turn off Buzzer: :

Show MORSE Code: A to Z

Go back to taking new input: ,

Start encryption: <; A will be shown as B, B as C, and Z as A.

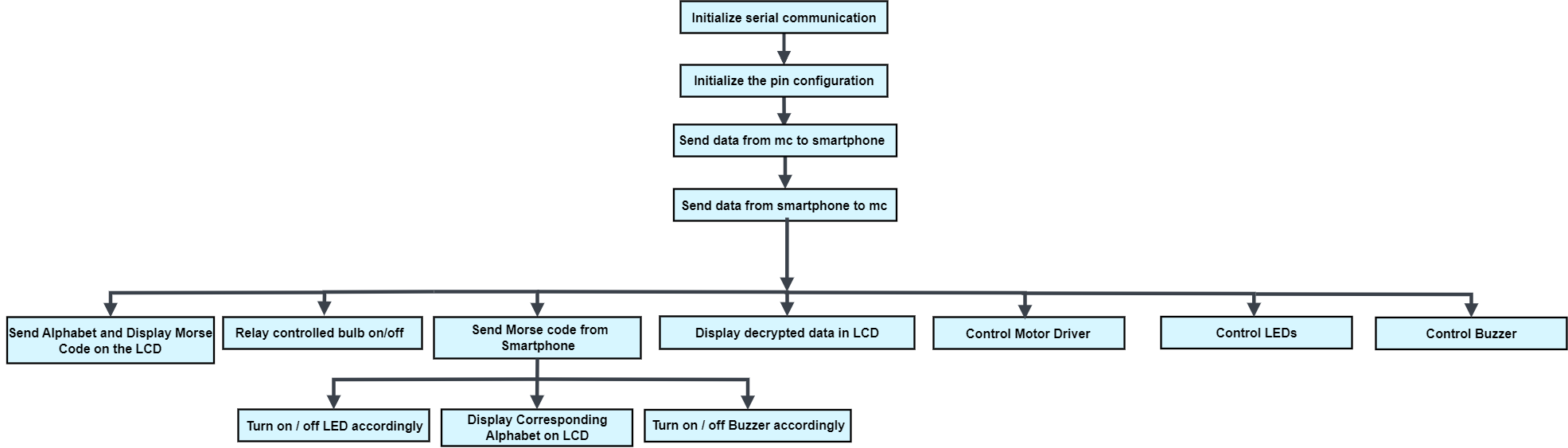
Stop encryption : >

Start Morse decrypt:?

Print Morse decrypt letter : )

Stop Morse decrypt: /

The given flowchart shows the workflow of this project.



# Code

ORG 00H

; INITIALIZE SERIAL COMMUNICATION

MOV TMOD,#20H ;Timer 1, mode 2 ; 0010 0000 ; 8 bit auto reload

MOV TH1,#0FDH ;9600 baud rate

MOV SCON,#50H; 8 bit data,1 stop bit, 1 start bit, REN enabled ; 0101 0000

CLR T1 ; Clear Timer 1 register

SETB TR1 ;start timer 1

;PIN CONFIGURATION

LED\_1 BIT P1.0 ; the buffer port is port 1.

LED\_2 BIT P1.1

LED\_3 BIT P1.2

LED\_4 BIT P1.3

LED\_5 BIT P1.4

LED\_6 BIT P1.5

LED\_7 BIT P1.6

LED\_8 BIT P1.7

LED\_RELAY BIT P3.5

MOTOR BIT P3.6

BUZZER BIT P3.7

CLR LED\_1

CLR LED\_2

CLR LED\_3

CLR LED\_4

CLR LED\_5

CLR LED\_6

CLR LED\_7

CLR LED\_8

SETB LED\_RELAY

CLR MOTOR

RS EQU P3.4

RW EQU P3.3

ENBL EQU P3.2

;SEND DATA FROM MICROCONTROLLER TO SMARTPHONE

MOV DPTR, #MYDATA

GO:

CLR A

MOVC A,@A+DPTR

JZ LCD ; LCD Subroutine

ACALL SEND

SJMP GO

SEND:

MOV SBUF, A

INC DPTR

HERE: JNB TI, HERE ;wait for the last bit to transfer

CLR TI ;clear TI for the next INPUT

RET

;MAKING THE LCD READY FOR DISPLAYING DATA FROM SMARTPHONE

; LCD Sub routine

LCD:

MOV SP,#70H ; 112D of ROM

MOV PSW,#00H

MOV A,#38H ;LCD 2 lines,5 by 7 matrix

LCALL COMMAND ;call command subroutine

LCALL DELAY ;give LCD some time

MOV A,#0EH ;display on, cursor on

LCALL COMMAND

LCALL DELAY

WELCOME:

MOV 32H, #16

LCALL CLEAR\_LCD

MOV A,#06H ;shift cursor right

LCALL COMMAND

LCALL DELAY

MOV A,#80H ;force cursor to begin at 1st line

LCALL COMMAND

LCALL DELAY

;DISPLAY WELCOME MESSAGE

LCALL DELAY

; Display the MYDATA string

MOV DPTR, #MYDATA ; Load the address of the string

LOOP\_1 : CLR A

MOVC A,@A+DPTR

JZ FINISH

LCALL DISPLAY

LCALL DELAY

INC DPTR

LJMP LOOP\_1

FINISH:

LJMP SMARTPHONE

MSG\_A: DB ".-",0

MSG\_B: DB "-...",0

MSg\_C: DB "-.-.",0

MSG\_D: DB "-..", 0

MSG\_E: DB ".", 0

MSG\_F: DB "..-.", 0

MSG\_G: DB "--.", 0

MSG\_H: DB "....", 0

MSG\_I: DB "..", 0

MSG\_J: DB ".---", 0

MSG\_K: DB "-.-", 0

MSG\_L: DB ".-..", 0

MSG\_M: DB "--", 0

MSG\_N: DB "-.", 0

MSG\_O: DB "---", 0

MSG\_P: DB ".--.", 0

MSG\_Q: DB "--.-", 0

MSG\_R: DB ".-.", 0

MSG\_S: DB "...", 0

MSG\_T: DB "-", 0

MSG\_U: DB "..-", 0

MSG\_V: DB "...-", 0

MSG\_W: DB ".--", 0

MSG\_X: DB "-..-", 0

MSG\_Y: DB "-.--", 0

MSG\_Z: DB "--..", 0

MYDATA: DB "Group 03, Bluetooth Interfacing",0 ; this data is shown from virtual terminal

;COMMAND FROM THE SMARTPHONE

SMARTPHONE:

MOV A, "H"

LCALL TRANS

TRANS:

MOV SBUF, A

MOV 32H, #16

LCALL CLEAR\_LCD

INPUT\_NEW:

LCALL GENERAL\_INPUT

MOV A, SBUF ;save incoming byte in A

CJNE A,#'1', LED\_1\_OFF

SETB LED\_1;turn on the LED

LJMP INPUT\_NEW ;print the INPUT\_NEW\_NEWacter pressed

LED\_1\_OFF:

MOV A,SBUF

CJNE A,#'2', LED\_2\_OFF

SETB LED\_2 ;turn ofF the LED

LJMP INPUT\_NEW

LED\_2\_OFF:

MOV A,SBUF

CJNE A,#'3', LED\_3\_OFF

SETB LED\_3

LJMP INPUT\_NEW

LED\_3\_OFF :

MOV A,SBUF

CJNE A,#'4', LED\_4\_OFF

SETB LED\_4

LJMP INPUT\_NEW

LED\_4\_OFF:

MOV A,SBUF

CJNE A,#'5',LED\_5\_OFF

SETB LED\_5

LJMP INPUT\_NEW

LED\_5\_OFF:

MOV A,SBUF

CJNE A,#'6',LED\_6\_OFF

SETB LED\_6

LJMP INPUT\_NEW

LED\_6\_OFF:

MOV A,SBUF

CJNE A,#'7',LED\_7\_OFF

SETB LED\_7

LJMP INPUT\_NEW

LED\_7\_OFF:

MOV A,SBUF

CJNE A,#'8',LED\_8\_OFF

SETB LED\_8

LJMP INPUT\_NEW

LED\_8\_OFF:

MOV A,SBUF

CJNE A,#'!',TURN\_ON\_RELAY

CLR LED\_1

LJMP INPUT\_NEW

TURN\_ON\_RELAY:

MOV A,SBUF

CJNE A,#'9',TURN\_OFF\_RELAY

CLR LED\_RELAY

LJMP INPUT\_NEW

TURN\_OFF\_RELAY:

MOV A,SBUF

CJNE A,#'(', NOT\_TURN\_OFF\_RELAY

SETB LED\_RELAY

LJMP INPUT\_NEW

NOT\_TURN\_OFF\_RELAY: ; TURN ON MOTOR

MOV A,SBUF

CJNE A,#'+', NOT\_TURN\_ON\_MOTOR

SETB MOTOR

LJMP INPUT\_NEW

NOT\_TURN\_ON\_MOTOR: ; TURN OFF MOTOR

MOV A,SBUF

CJNE A,#'=', NOT\_TURN\_OFF\_MOTOR

CLR MOTOR

LJMP INPUT\_NEW

NOT\_TURN\_OFF\_MOTOR:

MOV A,SBUF

CJNE A, #'"' , NOT\_TURN\_ON\_BUZZER

CLR BUZZER

LJMP INPUT\_NEW

NOT\_TURN\_ON\_BUZZER:

MOV A, SBUF

CJNE A, #':' , NOT\_TURN\_OFF\_BUZZER

SETB BUZZER

LJMP INPUT\_NEW

NOT\_TURN\_OFF\_BUZZER:

LED\_1\_ON:

MOV A, SBUF

CJNE A,#'@',LED\_2\_ON

CLR LED\_2

LJMP INPUT\_NEW

LED\_2\_ON:

MOV A, SBUF

CJNE A,#'#',LED\_3\_ON

CLR LED\_3

LJMP INPUT\_NEW

LED\_3\_ON:

MOV A, SBUF

CJNE A,#'$',LED\_4\_ON

CLR LED\_4

LJMP INPUT\_NEW

LED\_4\_ON:

MOV A, SBUF

CJNE A,#'%',LED\_5\_ON

CLR LED\_5

LJMP INPUT\_NEW

LED\_5\_ON:

MOV A, SBUF

CJNE A,#'^',LED\_6\_ON

CLR LED\_6

LJMP INPUT\_NEW

LED\_6\_ON:

MOV A, SBUF

CJNE A,#'&',LED\_7\_ON

CLR LED\_7

LJMP INPUT\_NEW

LED\_7\_ON:

MOV A, SBUF

CJNE A,#'\*',CHECK\_A

CLR LED\_8

LJMP INPUT\_NEW

CHECK\_A:

MOV A, SBUF

CJNE A,#'A' , NOT\_A ; Jump if not equal to "A"

MOV DPTR, #MSG\_A

LJMP DISPLAY\_MORSE ; Display Morse code for "A"

NOT\_A:

MOV A, SBUF

CJNE A, #'B', NOT\_B

MOV DPTR, #MSG\_B

LJMP DISPLAY\_MORSE

NOT\_B:

MOV A, SBUF

CJNE A, #'C', NOT\_C

MOV DPTR, #MSG\_C

LJMP DISPLAY\_MORSE

NOT\_C:

MOV A, SBUF

CJNE A, #'D', NOT\_D

MOV DPTR, #MSG\_D

LJMP DISPLAY\_MORSE

NOT\_D:

MOV A, SBUF

CJNE A, #'E', NOT\_E

MOV DPTR, #MSG\_E

LJMP DISPLAY\_MORSE

NOT\_E:

MOV A, SBUF

CJNE A, #'F', NOT\_F

MOV DPTR, #MSG\_F

LJMP DISPLAY\_MORSE

NOT\_F:

MOV A, SBUF

CJNE A, #'G', NOT\_G

MOV DPTR, #MSG\_G

LJMP DISPLAY\_MORSE

NOT\_G:

MOV A, SBUF

CJNE A, #'H', NOT\_H

MOV DPTR, #MSG\_H

LJMP DISPLAY\_MORSE

NOT\_H:

MOV A, SBUF

CJNE A, #'I', NOT\_I

MOV DPTR, #MSG\_I

LJMP DISPLAY\_MORSE

NOT\_I:

MOV A, SBUF

CJNE A, #'J', NOT\_J

MOV DPTR, #MSG\_J

LJMP DISPLAY\_MORSE

NOT\_J:

MOV A, SBUF

CJNE A, #'K', NOT\_K

MOV DPTR, #MSG\_K

LJMP DISPLAY\_MORSE

NOT\_K:

MOV A, SBUF

CJNE A, #'L', NOT\_L

MOV DPTR, #MSG\_L

LJMP DISPLAY\_MORSE

NOT\_L:

MOV A, SBUF

CJNE A, #'M', NOT\_M

MOV DPTR, #MSG\_M

LJMP DISPLAY\_MORSE

NOT\_M:

MOV A, SBUF

CJNE A, #'N', NOT\_N

MOV DPTR, #MSG\_N

LJMP DISPLAY\_MORSE

NOT\_N:

MOV A, SBUF

CJNE A, #'O', NOT\_O

MOV DPTR, #MSG\_O

LJMP DISPLAY\_MORSE

NOT\_O:

MOV A, SBUF

CJNE A, #'P', NOT\_P

MOV DPTR, #MSG\_P

LJMP DISPLAY\_MORSE

NOT\_P:

MOV A, SBUF

CJNE A, #'Q', NOT\_Q

MOV DPTR, #MSG\_Q

LJMP DISPLAY\_MORSE

NOT\_Q:

MOV A, SBUF

CJNE A, #'R', NOT\_R

MOV DPTR, #MSG\_R

LJMP DISPLAY\_MORSE

NOT\_R:

MOV A, SBUF

CJNE A, #'S', NOT\_S

MOV DPTR, #MSG\_S

LJMP DISPLAY\_MORSE

NOT\_S:

MOV A, SBUF

CJNE A, #'T', NOT\_T

MOV DPTR, #MSG\_T

LJMP DISPLAY\_MORSE

NOT\_T:

MOV A, SBUF

CJNE A, #'U', NOT\_U

MOV DPTR, #MSG\_U

LJMP DISPLAY\_MORSE

NOT\_U:

MOV A, SBUF

CJNE A, #'V', NOT\_V

MOV DPTR, #MSG\_V

LJMP DISPLAY\_MORSE

NOT\_V:

MOV A, SBUF

CJNE A, #'W', NOT\_W

MOV DPTR, #MSG\_W

LJMP DISPLAY\_MORSE

NOT\_W:

MOV A, SBUF

CJNE A, #'X', NOT\_X

MOV DPTR, #MSG\_X

LJMP DISPLAY\_MORSE

NOT\_X:

MOV A, SBUF

CJNE A, #'Y', NOT\_Y

MOV DPTR, #MSG\_Y

LJMP DISPLAY\_MORSE

NOT\_Y:

MOV A, SBUF

CJNE A, #'Z', NOT\_Z

MOV DPTR, #MSG\_Z

LJMP DISPLAY\_MORSE

DISPLAY\_MORSE:

LOOP\_MORSE: CLR A

MOVC A,@A+DPTR

JNZ NOT\_ZERO\_MORSE

ZERO\_MORSE : LJMP INPUT\_NEW

NOT\_ZERO\_MORSE:

LCALL DISPLAY

LCALL DELAY

INC DPTR

LJMP LOOP\_MORSE

NOT\_Z:

MOV A, SBUF

CJNE A, #'<', NOT\_ENCRYPTION

LJMP ENCRYPTION

NOT\_ENCRYPTION:

MOV A, SBUF

CJNE A, #'?', NOT\_MORSE\_DECRYPT

LJMP MORSE\_DECRYPT

NOT\_MORSE\_DECRYPT:

MOV A, SBUF

CJNE A, #',' , NOT\_NEW\_INPUT

LJMP SMARTPHONE

NOT\_NEW\_INPUT:

LJMP SMARTPHONE

ENCRYPTION:

MOV 32H, #16

LCALL CLEAR\_LCD

ENCRYPTION\_INPUT:

LCALL GENERAL\_INPUT

MOV A, SBUF

CJNE A, #'Z', NOT\_Z\_ENCRYPTION

MOV A, #'A' ; EQUAL TO Z

LCALL DISPLAY

LCALL DELAY

LJMP ENCRYPTION\_INPUT

NOT\_Z\_ENCRYPTION:

MOV A, SBUF

CJNE A, #'>', CONTINUE\_ENCRYPTION

LJMP WELCOME ; STOP ENCRYPTION AND GO BACK TO MAIN INPUT SUBROUTINE

CONTINUE\_ENCRYPTION:

ADD A, #1

LCALL DISPLAY

LCALL DELAY

LJMP ENCRYPTION\_INPUT

MORSE\_DECRYPT:

MOV 32H, #16

;SETB LED\_1

LCALL CLEAR\_LCD

INPUT\_NEW\_MORSE:

MOV R0 , #0

MOV R2 , #0

MOV R6 , #0

MOV R7 , #0

MOV R5 , #1 ;COUNTER

MORSE\_DECRYPT\_INPUT:

LCALL GENERAL\_INPUT

MOV A, SBUF

CJNE A, #'/', NOT\_STOP\_MORSE\_DECRYPT

LJMP WELCOME

NOT\_STOP\_MORSE\_DECRYPT:

MOV A, SBUF

CJNE A, #')' , CONTINUE\_MORSE\_DECRYPT

LJMP DISPLAY\_MORSE\_DECRYPT

CONTINUE\_MORSE\_DECRYPT:

MOV A, SBUF

CJNE R5 , #1 , NOT\_1 ; if R5=1, store the input in R0

SJMP COUNTER\_1

NOT\_1:

CJNE R5 , #2 , NOT\_2 ; if R5=2, store the input in R6

SJMP COUNTER\_2

NOT\_2:

CJNE R5 , #3 , NOT\_3

SJMP COUNTER\_3

NOT\_3:

CJNE R5 , #4 , NOT\_4

SJMP COUNTER\_4

NOT\_4:

LJMP WELCOME

COUNTER\_1:

MOV R0 , A

INC R5

LJMP MORSE\_DECRYPT\_INPUT

COUNTER\_2:

MOV R6 , A

INC R5

LJMP MORSE\_DECRYPT\_INPUT

COUNTER\_3:

MOV R2, A

INC R5

LJMP MORSE\_DECRYPT\_INPUT

COUNTER\_4:

MOV R7 , A

LJMP MORSE\_DECRYPT\_INPUT

DISPLAY\_MORSE\_DECRYPT:

;CHECK FOR A

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_A

CJNE R6, #'-' , NOT\_MORSE\_DECRYPT\_A

CJNE R2, #0 , NOT\_MORSE\_DECRYPT\_A

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_A

LJMP MORSE\_DECRYPT\_A

;CHECK FOR B

NOT\_MORSE\_DECRYPT\_A:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_B

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_B

CJNE R2, #'.' , NOT\_MORSE\_DECRYPT\_B

CJNE R7, #'.' , NOT\_MORSE\_DECRYPT\_B

LJMP MORSE\_DECRYPT\_B

; CHECK FOR C

NOT\_MORSE\_DECRYPT\_B:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_C

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_C

CJNE R2, #'-' , NOT\_MORSE\_DECRYPT\_C

CJNE R7, #'.' , NOT\_MORSE\_DECRYPT\_C

LJMP MORSE\_DECRYPT\_C

; CHECK FOR D

NOT\_MORSE\_DECRYPT\_C:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_D

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_D

CJNE R2, #'.' , NOT\_MORSE\_DECRYPT\_D

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_D

LJMP MORSE\_DECRYPT\_D

; CHECK FOR E

NOT\_MORSE\_DECRYPT\_D:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_E

CJNE R6, #0 , NOT\_MORSE\_DECRYPT\_E

CJNE R2, #0 , NOT\_MORSE\_DECRYPT\_E

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_E

LJMP MORSE\_DECRYPT\_E

; CHECK FOR F

NOT\_MORSE\_DECRYPT\_E:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_F

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_F

CJNE R2, #'-' , NOT\_MORSE\_DECRYPT\_F

CJNE R7, #'.' , NOT\_MORSE\_DECRYPT\_F

LJMP MORSE\_DECRYPT\_F

; CHECK FOR G

NOT\_MORSE\_DECRYPT\_F:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_G

CJNE R6, #'-' , NOT\_MORSE\_DECRYPT\_G

CJNE R2, #'.' , NOT\_MORSE\_DECRYPT\_G

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_G

LJMP MORSE\_DECRYPT\_G

; CHECK FOR H

NOT\_MORSE\_DECRYPT\_G:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_H

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_H

CJNE R2, #'.' , NOT\_MORSE\_DECRYPT\_H

CJNE R7, #'.' , NOT\_MORSE\_DECRYPT\_H

LJMP MORSE\_DECRYPT\_H

; CHECK FOR I

NOT\_MORSE\_DECRYPT\_H:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_I

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_I

CJNE R2, #0 , NOT\_MORSE\_DECRYPT\_I

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_I

LJMP MORSE\_DECRYPT\_I

; CHECK FOR J

NOT\_MORSE\_DECRYPT\_I:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_J

CJNE R6, #'-' , NOT\_MORSE\_DECRYPT\_J

CJNE R2, #'-' , NOT\_MORSE\_DECRYPT\_J

CJNE R7, #'-' , NOT\_MORSE\_DECRYPT\_J

LJMP MORSE\_DECRYPT\_J

; CHECK FOR K

NOT\_MORSE\_DECRYPT\_J:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_K

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_K

CJNE R2, #'-' , NOT\_MORSE\_DECRYPT\_K

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_K

LJMP MORSE\_DECRYPT\_K

; CHECK FOR L

NOT\_MORSE\_DECRYPT\_K:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_L

CJNE R6, #'-' , NOT\_MORSE\_DECRYPT\_L

CJNE R2, #'.' , NOT\_MORSE\_DECRYPT\_L

CJNE R7, #'.' , NOT\_MORSE\_DECRYPT\_L

LJMP MORSE\_DECRYPT\_L

; CHECK FOR M

NOT\_MORSE\_DECRYPT\_L:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_M

CJNE R6, #'-' , NOT\_MORSE\_DECRYPT\_M

CJNE R2, #0 , NOT\_MORSE\_DECRYPT\_M

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_M

LJMP MORSE\_DECRYPT\_M

; CHECK FOR N

NOT\_MORSE\_DECRYPT\_M:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_N

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_N

CJNE R2, #0 , NOT\_MORSE\_DECRYPT\_N

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_N

LJMP MORSE\_DECRYPT\_N

; CHECK FOR O

NOT\_MORSE\_DECRYPT\_N:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_O

CJNE R6, #'-' , NOT\_MORSE\_DECRYPT\_O

CJNE R2, #'-' , NOT\_MORSE\_DECRYPT\_O

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_O

LJMP MORSE\_DECRYPT\_O

; CHECK FOR P

NOT\_MORSE\_DECRYPT\_O:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_P

CJNE R6, #'-' , NOT\_MORSE\_DECRYPT\_P

CJNE R2, #'-' , NOT\_MORSE\_DECRYPT\_P

CJNE R7, #'.' , NOT\_MORSE\_DECRYPT\_P

LJMP MORSE\_DECRYPT\_P

; CHECK FOR Q

NOT\_MORSE\_DECRYPT\_P:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_Q

CJNE R6, #'-' , NOT\_MORSE\_DECRYPT\_Q

CJNE R2, #'.' , NOT\_MORSE\_DECRYPT\_Q

CJNE R7, #'-' , NOT\_MORSE\_DECRYPT\_Q

LJMP MORSE\_DECRYPT\_Q

; CHECK FOR R

NOT\_MORSE\_DECRYPT\_Q:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_R

CJNE R6, #'-' , NOT\_MORSE\_DECRYPT\_R

CJNE R2, #'.' , NOT\_MORSE\_DECRYPT\_R

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_R

LJMP MORSE\_DECRYPT\_R

; CHECK FOR S

NOT\_MORSE\_DECRYPT\_R:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_S

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_S

CJNE R2, #'.' , NOT\_MORSE\_DECRYPT\_S

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_S

LJMP MORSE\_DECRYPT\_S

; CHECK FOR T

NOT\_MORSE\_DECRYPT\_S:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_T

CJNE R6, #0 , NOT\_MORSE\_DECRYPT\_T

CJNE R2, #0 , NOT\_MORSE\_DECRYPT\_T

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_T

LJMP MORSE\_DECRYPT\_T

; CHECK FOR U

NOT\_MORSE\_DECRYPT\_T:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_U

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_U

CJNE R2, #'-' , NOT\_MORSE\_DECRYPT\_U

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_U

LJMP MORSE\_DECRYPT\_U

; CHECK FOR V

NOT\_MORSE\_DECRYPT\_U:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_V

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_V

CJNE R2, #'.' , NOT\_MORSE\_DECRYPT\_V

CJNE R7, #'-' , NOT\_MORSE\_DECRYPT\_V

LJMP MORSE\_DECRYPT\_V

; CHECK FOR W

NOT\_MORSE\_DECRYPT\_V:

CJNE R0, #'.' , NOT\_MORSE\_DECRYPT\_W

CJNE R6, #'-' , NOT\_MORSE\_DECRYPT\_W

CJNE R2, #'-' , NOT\_MORSE\_DECRYPT\_W

CJNE R7, #0 , NOT\_MORSE\_DECRYPT\_W

LJMP MORSE\_DECRYPT\_W

; CHECK FOR X

NOT\_MORSE\_DECRYPT\_W:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_X

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_X

CJNE R2, #'.' , NOT\_MORSE\_DECRYPT\_X

CJNE R7, #'-' , NOT\_MORSE\_DECRYPT\_X

LJMP MORSE\_DECRYPT\_X

; CHECK FOR Y

NOT\_MORSE\_DECRYPT\_X:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_Y

CJNE R6, #'.' , NOT\_MORSE\_DECRYPT\_Y

CJNE R2, #'-' , NOT\_MORSE\_DECRYPT\_Y

CJNE R7, #'-' , NOT\_MORSE\_DECRYPT\_Y

LJMP MORSE\_DECRYPT\_Y

; CHECK FOR Z

NOT\_MORSE\_DECRYPT\_Y:

CJNE R0, #'-' , NOT\_MORSE\_DECRYPT\_Z

CJNE R6, #'-' , NOT\_MORSE\_DECRYPT\_Z

CJNE R2, #'.' , NOT\_MORSE\_DECRYPT\_Z

CJNE R7, #'.' , NOT\_MORSE\_DECRYPT\_Z

LJMP MORSE\_DECRYPT\_Z

NOT\_MORSE\_DECRYPT\_Z:

LJMP INPUT\_NEW\_MORSE

MORSE\_DECRYPT\_A:

MOV A, #'A'

LJMP MORSE\_LED

MORSE\_DECRYPT\_B:

MOV A, #'B'

LJMP MORSE\_LED

MORSE\_DECRYPT\_C:

MOV A, #'C'

LJMP MORSE\_LED

MORSE\_DECRYPT\_D:

MOV A, #'D'

LJMP MORSE\_LED

MORSE\_DECRYPT\_E:

MOV A, #'E'

LJMP MORSE\_LED

MORSE\_DECRYPT\_F:

MOV A, #'F'

LJMP MORSE\_LED

MORSE\_DECRYPT\_G:

MOV A, #'G'

LJMP MORSE\_LED

MORSE\_DECRYPT\_H:

MOV A, #'H'

LJMP MORSE\_LED

MORSE\_DECRYPT\_I:

MOV A, #'I'

LJMP MORSE\_LED

MORSE\_DECRYPT\_J:

MOV A, #'J'

LJMP MORSE\_LED

MORSE\_DECRYPT\_K:

MOV A, #'K'

LJMP MORSE\_LED

MORSE\_DECRYPT\_L:

MOV A, #'L'

LJMP MORSE\_LED

MORSE\_DECRYPT\_M:

MOV A, #'M'

LJMP MORSE\_LED

MORSE\_DECRYPT\_N:

MOV A, #'N'

LJMP MORSE\_LED

MORSE\_DECRYPT\_O:

MOV A, #'O'

LJMP MORSE\_LED

MORSE\_DECRYPT\_P:

MOV A, #'P'

LJMP MORSE\_LED

MORSE\_DECRYPT\_Q:

MOV A, #'Q'

LJMP MORSE\_LED

MORSE\_DECRYPT\_R:

MOV A, #'R'

LJMP MORSE\_LED

MORSE\_DECRYPT\_S:

MOV A, #'S'

LJMP MORSE\_LED

MORSE\_DECRYPT\_T:

MOV A, #'T'

LJMP MORSE\_LED

MORSE\_DECRYPT\_U:

MOV A, #'U'

LJMP MORSE\_LED

MORSE\_DECRYPT\_V:

MOV A, #'V'

LJMP MORSE\_LED

MORSE\_DECRYPT\_W:

MOV A, #'W'

LJMP MORSE\_LED

MORSE\_DECRYPT\_X:

MOV A, #'X'

LJMP MORSE\_LED

MORSE\_DECRYPT\_Y:

MOV A, #'Y'

LJMP MORSE\_LED

MORSE\_DECRYPT\_Z:

MOV A, #'Z'

LJMP MORSE\_LED

MORSE\_LED:

LCALL DISPLAY

LCALL DELAY

CHECK\_R0:

CJNE R0, #'-' , R0\_NOT\_DASH

LJMP R0\_DASH

R0\_NOT\_DASH:

CJNE R0, #'.' , R0\_NOT\_DOT

LJMP R0\_DOT

R0\_NOT\_DOT:

CJNE R0, #'0' , R0\_NOT\_ZERO

LJMP R0\_ZERO

R0\_NOT\_ZERO:

LJMP INPUT\_NEW\_MORSE

R0\_DASH:

LCALL DASH

LJMP CHECK\_R6

R0\_DOT:

LCALL DOT

LJMP CHECK\_R6

R0\_ZERO:

LJMP INPUT\_NEW\_MORSE

CHECK\_R6:

CJNE R6, #'-' , R6\_NOT\_DASH

LJMP R6\_DASH

R6\_NOT\_DASH:

CJNE R6, #'.' , R6\_NOT\_DOT

LJMP R6\_DOT

R6\_NOT\_DOT:

CJNE R6, #'0' , R6\_NOT\_ZERO

LJMP R6\_ZERO

R6\_NOT\_ZERO:

LJMP INPUT\_NEW\_MORSE

R6\_DASH:

LCALL DASH

LJMP CHECK\_R2

R6\_DOT:

LCALL DOT

LJMP CHECK\_R2

R6\_ZERO:

LJMP INPUT\_NEW\_MORSE

CHECK\_R2:

CJNE R2, #'-' , R2\_NOT\_DASH

LJMP R2\_DASH

R2\_NOT\_DASH:

CJNE R2, #'.' , R2\_NOT\_DOT

LJMP R2\_DOT

R2\_NOT\_DOT:

CJNE R2, #'0' , R2\_NOT\_ZERO

LJMP R2\_ZERO

R2\_NOT\_ZERO:

LJMP INPUT\_NEW\_MORSE

R2\_DASH:

LCALL DASH

LJMP CHECK\_R7

R2\_DOT:

LCALL DOT

LJMP CHECK\_R7

R2\_ZERO:

LJMP INPUT\_NEW\_MORSE

CHECK\_R7:

CJNE R7, #'-' , R7\_NOT\_DASH

LJMP R7\_DASH

R7\_NOT\_DASH:

CJNE R7, #'.' , R7\_NOT\_DOT

LJMP R7\_DOT

R7\_NOT\_DOT:

CJNE R7, #'0' , R7\_NOT\_ZERO

LJMP R7\_ZERO

R7\_NOT\_ZERO:

LJMP INPUT\_NEW\_MORSE

R7\_DASH:

LCALL DASH

LJMP INPUT\_NEW\_MORSE

R7\_DOT:

LCALL DOT

LJMP INPUT\_NEW\_MORSE

R7\_ZERO:

LJMP INPUT\_NEW\_MORSE

CLEAR\_LCD:

LCALL DELAY

MOV A,#01H ;clear lcd

LCALL COMMAND

LCALL DELAY

RET

GENERAL\_INPUT:

ACALL DELAY

CLR A

CLR RI ;get ready to receive data

WAIT\_INPUT: JNB RI, WAIT\_INPUT ;wait for the INPUT\_NEW to come in

RET

COMMAND:

LCALL READY

MOV P2,A

CLR RS ;control reg is selected, send command to LCD controller

CLR RW ;write to LCD

SETB ENBL

LCALL DELAY

CLR ENBL ;H to L pulse

RET

DISPLAY:

LCALL READY

MOV P2,A

SETB RS ;data register is selected, RS=1 send data to LCD

CLR RW ;write to LCD

SETB ENBL

LCALL DELAY

CLR ENBL ;H to L pulse

DJNZ 32H, SAME\_LINE

LJMP NEW\_LINE

NEW\_LINE:

LCALL DELAY

MOV A,#0C0H ;2ND LINE

LCALL COMMAND

LCALL DELAY

SAME\_LINE:

RET

;COMMAND and DISPLAY subroutine is same except the RS pin

READY:

SETB P2.7

CLR RS ;control reg is selected, send command to LCD controller

SETB RW ;reading from LCD

WAIT:

CLR ENBL ; disables communication with the LCD temporarily

ACALL DELAY ;L to H pulse

SETB ENBL

JB P2.7,WAIT ;checks the busy flag

RET

DASH:

SETB LED\_1

CLR BUZZER

LCALL DELAY

LCALL DELAY

LCALL DELAY

LCALL DELAY

LCALL DELAY

LCALL DELAY

CLR LED\_1

SETB BUZZER

LCALL DELAY

RET

DOT:

SETB LED\_1

CLR BUZZER

LCALL DELAY

LCALL DELAY

CLR LED\_1

SETB BUZZER

LCALL DELAY

RET

DELAY:

MOV R1,#1

AGAIN\_3:

MOV R3,#220

AGAIN\_2:

MOV R4,#220 ; delay

AGAIN:

DJNZ R4, AGAIN

DJNZ R3, AGAIN\_2

DJNZ R1, AGAIN\_3

RET

EXIT\_2: SJMP EXIT\_2

END

# Hardware Implementation



# Video Demonstration

<https://iutdhaka-my.sharepoint.com/:f:/g/personal/wasikbillah_iut-dhaka_edu/Eoq9mdkVzVtJt-4lHLmYiVIBrAITXZDlXIZdxVuS-u4guA?e=crVyT9>

N.B. Please copy and paste the link in any browser, preferably in a desktop browser.

# Problems Faced

1. The HC05 module in Proteus software was not working as expected. So, a virtual terminal was used.
2. Debugging required LEDs to be turned on and off at potential places of the code. This required many trial and error consuming a lot of time.

# Conclusion

The objective of the project on Bluetooth interfacing has been implemented here. With three mandatory features, six other new features were also included. In completing this project, the knowledge of Bluetooth communication using a serial port of interfacing Bluetooth module was gained and learned. The project is done in both hardware and software. Proteus 8.16 Professional was used to build the circuit for the software part. Hardware was implemented using a Microcontroller, LCD monitor, HC-05 Bluetooth module, Relay, Motor, and Buzzer.