EEE 4706

Project Report

Bluetooth Interfacing with 8051 Microcontroller

Lab Group A1
Group 2

Ahmed Jawad Rashid (190021107)
Shihab Shahriar (190021123)
Samiha Ishrat (190021129)
Md. Mahfuzul Islam (190021113)
Mohamuud Musse Yusuf (190021141)

Bluetooth Interfacing with 8051 Microcontroller

Introduction

The objective of this project was to combine Bluetooth technology with the 8051 microprocessor. The objective was to facilitate different operations that were activated by certain characters sent via a Serial Bluetooth Android application. These characters enabled several functions, including initiating countdowns, controlling relays and buzzers, transmitting Morse code, and implementing encryption. The report provides an extensive documentation that outlines the practical aspects, features, and importance of incorporating Bluetooth technology into the 8051 microcontroller in order to facilitate a wide range of real-time programmes.

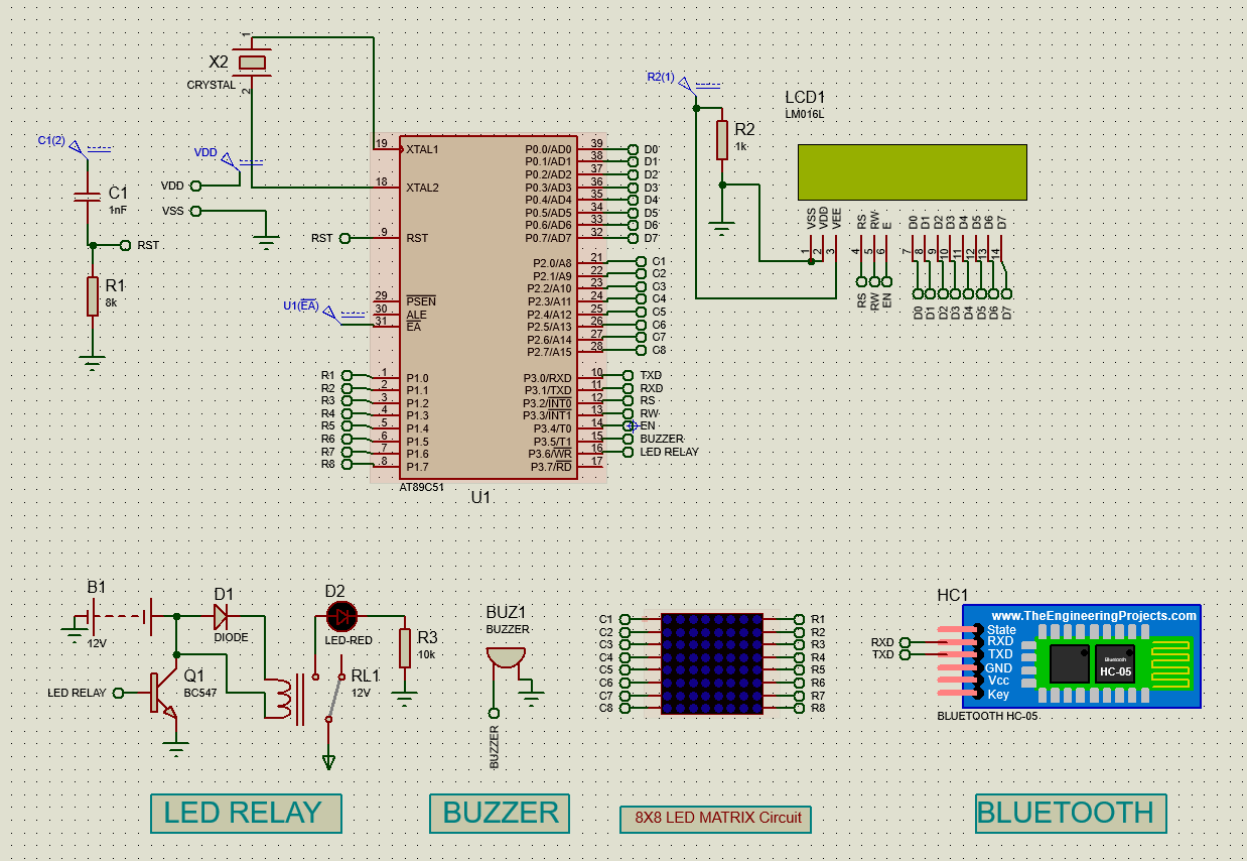
Objectives

1. **Receive Data:** 8051 Microcontroller to receive data from the HC 05 Bluetooth module.
2. **Check for Specific Characters:** Once data is received, the uC needs to check for specific special characters (C, L, y, n, 1, M, X, D, 0).
3. **Execute Corresponding Actions:** Implementation functions or logic to perform the desired actions based on the received characters. For example:
    - For 'C': Countdown and buzzer ringing with LCD display updates.
    - For uppercase 'L' and lowercase 'L': Controlling the LED relay accordingly.
    - For 'y' and 'n': Activate or deactivate the buzzer.
    - For '1': Toggle the LED relay based on subsequent numbers received.
    - For 'M': Activate the Morse code mode.
    - For 'X': Switch to the 8x8 LED Matrix mode.
    - For 'D': Implement the encryption mode for the messages.
4. **Exit Modes:** When '0' is received, reset the system to its initial state.

Required Components

Name	Price (BDT)
8051 Development Board	7000
Bluetooth HC-05	350
Total	**7350**

Schematic Circuit



Our chosen Bluetooth Module: HC 05

HM-06 is a Bluetooth module designed for establishing short range wireless data communication between two microcontrollers or systems. The module works on Bluetooth 2.0 communication protocol and it can only act as a slave device. This is cheapest method for wireless data transmission and more flexible compared to other methods and it even can transmit files at speed up to 2.1Mb/s.

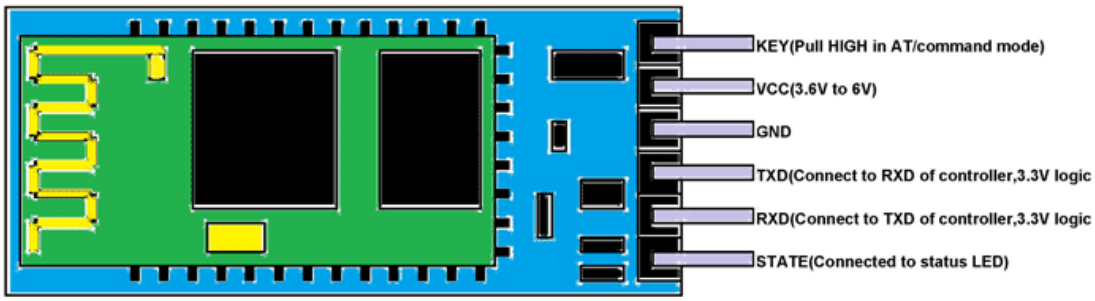


Figure 1. HC 05 Bluetooth Module

HC-06 uses frequency hopping spread spectrum technique (FHSS) to avoid interference with other devices and to have full duplex transmission. The device works on the frequency range from 2.402 GHz to 2.480GHz.

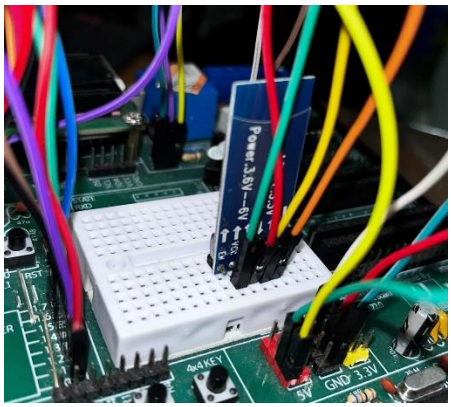


Figure 2. HC05 Hardware Implementation

Code

```
; for Bluetooth module
MOV TMOD,#20H ; timer 1 mode 2 is selected
MOV TH1,#0FDH ; baud rate
MOV SCON,#50H ; serial mode 1 10 bit total isn, 8db, 1STOPb
CLR TI ; making TI reg zero
SETB TR1 ; starting timer 1
```

Pinout of the Project

P0.0	LCD
P0.1	
P0.2	
P0.3	
P0.4	
P0.5	
P0.6	
P0.7	
P1.0	R1
P1.1	R2
P1.2	R3
P1.3	R4
P1.4	R5
P1.5	R6
P1.6	R7
P1.7	R8
P2.0	C1
P2.1	C2
P2.2	C3
P2.3	C4
P2.4	C5
P2.5	C6
P2.6	C7
P2.7	C8
P3.0	TXD
P3.1	RXD
P3.2	RS
P3.3	RW
P3.4	EN
P3.5	BUZZER
P3.6	LED
P3.7	N/A

Control Panel

Sl No.	Character	Mode Name	Description
1	C	Counter	The system initiates a countdown from 9 to 1, subsequently activating a buzzer on the 10th second for a specified duration. Simultaneously, the countdown was displayed on an LCD screen
2	Uppercase L	LED ON	Turns LED Relay ON
3	Lowercase L	LED OFF	Turns LED Relay OFF
4	y	Buzzer ON	Turns Buzzer ON
5	n	Buzzer OFF	Turns Buzzer OFF
6	M	Morse Code	Sends Messages in Morse Code of either A or B.
7	X	LED Matrix	Activates 8x8 LED Matrix of the 8051 Development Board and has the ability to display either A, B or C.
8	D	Encryption	Triggers Encryption Mode where messages were replaced with predetermined encrypted message formats
9	1 (One)	Relay Control	Toggle LED relay based on subsequent numbers received
10	0 (Zero)	Mode OFF	Upon sending, turns of any of the above activated modes and reverts back to the original state when the 8051 was turned ON.

LCD Initialisation

The following is a snippet of the LCD initialization part of the entire code. We used DPTR look tables to access the values for COMWRT to be set for LCD. And subsequently, we added a small message to begin the project demonstration.

Code

```
ORG 00H

;------------DISPLAY INITIALIZATION----------
       MOV DPTR,#MYLCD    ;DPTR stores the LCD initialization sequence
CIU1:   CLR A
       MOVC A,@A+DPTR
       LCALL COMNWRT
       LCALL DELAY
       JZ SIU1           ;Runs the rest of the code
       INC DPTR
       SJMP CIU1


SIU1:   MOV DPTR,#MSG1
DIU1:   CLR A
       MOVC  A,@A+DPTR
       LCALL DATAWRT
       LCALL DOT
       JZ SIU2           ;Runs rest of the code
       INC DPTR
       SJMP DIU1
SIU2:  MOV A, #01    ;Clear LCD
       ACALL COMNWRT ;Call command subroutine
       ACALL DELAY   ;Give LCD some time
       LCALL DATAWRT
       MOV DPTR,#MSG2
DIU2:  CLR A
       MOVC  A,@A+DPTR
       LCALL DATAWRT
       LCALL DOT
       JZ CONT1          ;Runs rest of the code
```

```
       INC DPTR
       SJMP DIU2

CONT1:
GOBACK:
CONT_RE:
CONT_M:
CONT_PRE:
CLR RI ; register involved in receiving data from bluetooth and ensuring it
REP: JNB RI, REP
; preparing LCD
MOV DPTR,#MYLCD2    ;DPTR stores the LCD initialization sequence
CIIU1:  CLR A
       MOVC A,@A+DPTR
       LCALL COMNWRT
       LCALL DELAY
       JZ SIIU1           ;Runs the rest of the code
       INC DPTR
       SJMP CIIU1

ORG 300H
MSG1: DB " GROUP 2 ",0
MSG2: DB " PROJECT 2 ",0
MYLCD : DB 38H,0EH,01,06,80H,0
MYLCD2 : DB 38H,0EH,01,06,0CH,0
END
```

Demonstration

Counter

This is where we basically print 9 to 1, where a number changes each second. And at the end of counting, on the 10th second, we turn the Buzzer on. And the buzzer automatically turns off after a while.

Code

```
CJNE A,#'C', GOBACK1
LJMP GO3
GOBACK1: LJMP GOBACK
GO3:
MOV R1, #9
MOV R2, #39H

COUNT_LOOP:
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY

MOV A,#38H
ACALL COMNWRT
ACALL DELAY
MOV A,#0EH
ACALL COMNWRT
ACALL DELAY
MOV A,#01H
ACALL COMNWRT
ACALL DELAY
MOV A,#06H
ACALL COMNWRT
ACALL DELAY
MOV A,#0CH
ACALL COMNWRT
ACALL DELAY
MOV A,R2
DEC R2
ACALL DATAWRT
ACALL DELAY
DJNZ R1, COUNT_LOOP

CLR P3.5 ; Buzzer on
ACALL DELAY1
SETB P3.5 ; after a certain time,
turn it off
```

N.B.Demonstration can be shown with video or physical demonstration

Morse Code Mode

Definition: Morse code is a method used in telecommunication to encode text characters as standardized sequences of two different signal durations, called DOTS and DASHES.

Say, for example, we can send A using the following pattern “DOT DASH”. And B using “DASH DOT DOT DOT”. And the patterns for the various letters of Alphabet are given here,

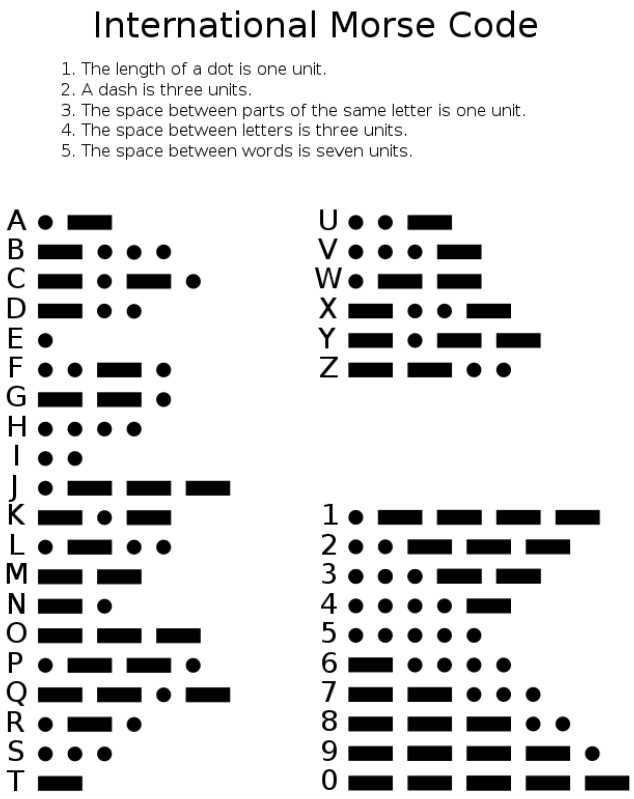


Figure. International Morse Code List

Here, we basically created 2 separate delay sub-routine for the DOT and DASH. And then, according to our need, we generated light and sound using LED Relay and Buzzer. Our implementation scope was only for A and B. We didn’t continue to do for the rest, because our goal was to show the implementation for any letter. And we did 2 of them, to show that we can send A and B in any order we want but still we would be able to achieve that, up and until we press ‘0’ to go into the **EXIT mode**.

Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;
CHECKNEXT_RE:
CJNE A,#'M', CHECKN_M
CLR RI

GO_M:
CLR RI
REP4: JNB RI,REP4
SJMP GO5
CHECKN_M: LJMP CHECKNEXT_M
GO5:

MOV A,SBUF
CJNE A,#'0',JA_M
LJMP CONT_M
JA_M:

CJNE A,#'A',M_B
CLR P3.5
CLR P3.6
ACALL DOT
SETB P3.5
SETB P3.6
ACALL DELAY_RE
CLR P3.5
CLR P3.6
ACALL DASH
ACALL DASH
SETB P3.5
SETB P3.6
ACALL DELAY
CLR RI
LJMP GO_M

M_B:
CJNE A,#'B',GO_M
CLR P3.5
CLR P3.6
ACALL DASH
SETB P3.5
SETB P3.6
ACALL DELAY_RE
CLR P3.5
CLR P3.6
ACALL DOT
SETB P3.5
SETB P3.6
ACALL DELAY_RE
CLR P3.5
CLR P3.6
ACALL DOT
SETB P3.5
SETB P3.6
ACALL DELAY
CLR RI
LJMP GO_M
```

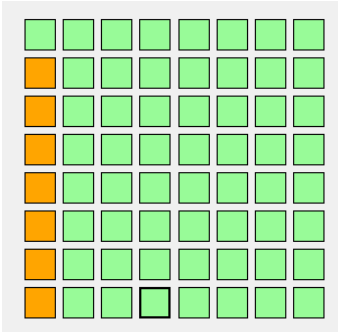
LED Matrix

An LED dot matrix display consists of a matrix of LED's arranged in a rectangular configuration. An 8×8 matrix consists of 64 dots or pixels. There is a LED for each pixel and these LEDs are connected to total of 16 pins. It can be used to display almost anything by switching ON /OFF a desired configuration of LED's. You can identify the pin out and circuit diagram of it using the following figure. The dot matrix display has 64 LEDs and evenly grouped into 8 columns and 8 rows. Any individual LED or a group of LEDs in the matrix can be activated by switching the required number of rows and columns. For example, in the above figure if Row1 is made high and Column1 is made low, the top left LED (address R1C1) will glow.

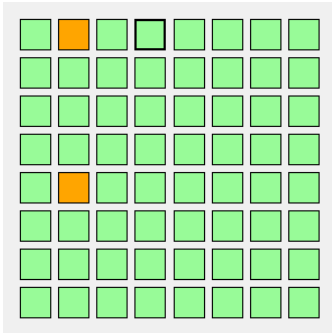
Working Procedure

R1-R8 → Row Initialisation (Active High)

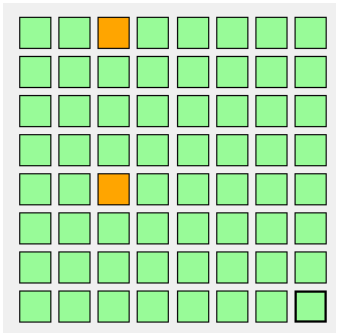
C1-C8 → Column Initialisation (Active Low)



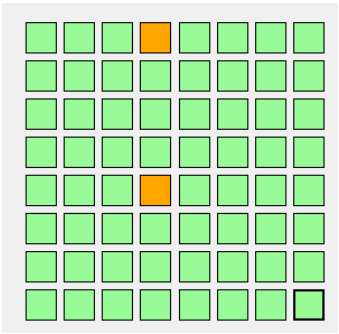
R1	R2	R3	R4	R5	R6	R7	R8
0	1	1	1	1	1	1	0
C1	C2	C3	C4	C5	C6	C7	C8
0	1	1	1	1	1	1	1



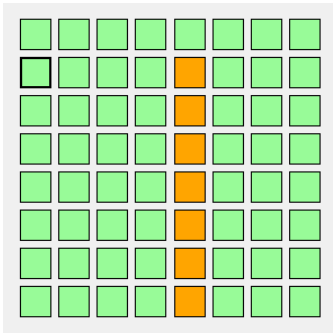
R1	R2	R3	R4	R5	R6	R7	R8
0	0	0	1	0	0	0	1
C1	C2	C3	C4	C5	C6	C7	C8
1	0	1	1	1	1	1	1



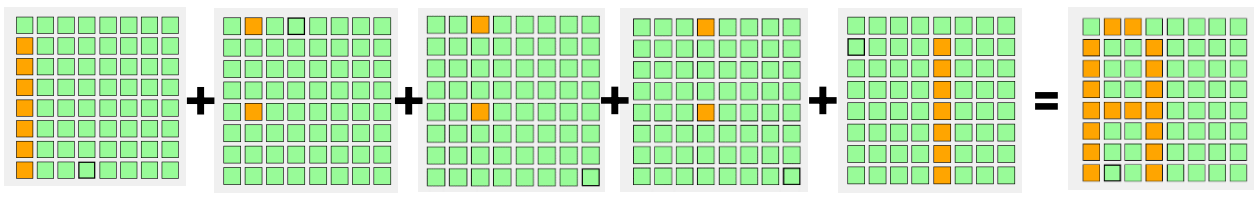
R1	R2	R3	R4	R5	R6	R7	R8
0	0	0	1	0	0	0	1
C1	C2	C3	C4	C5	C6	C7	C8
1	1	0	1	1	1	1	1



R1	R2	R3	R4	R5	R6	R7	R8
0	0	0	1	0	0	0	1
C1	C2	C3	C4	C5	C6	C7	C8
1	1	1	0	1	1	1	1



R1	R2	R3	R4	R5	R6	R7	R8
0	0	0	1	0	0	0	1
C1	C2	C3	C4	C5	C6	C7	C8
1	1	1	0	1	1	1	1



Code

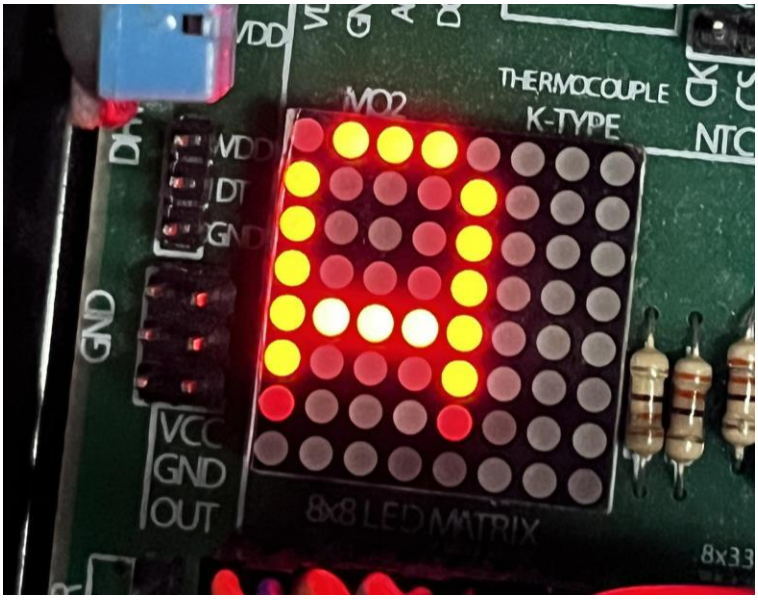
```
CJNE A,#'X',CHECKN_PRE
CLR RI
GO_PRE:
CLR RI
MOV P2,#00000000B
MOV P1,#00000000B
REP14: JNB RI,REP14
SJMP GO15
CHECKN_PRE: LJMP CHECKNEXT_PRE
GO15:
MOV A,SBUF
CJNE A,#'0',JA_PRE
MOV P2,#11111111B
MOV P1,#11111111B
LJMP CONT_PRE
JA_PRE:

CJNE A,#'A',LED_B
       MOV R1,#1
       LED_LOOP2: MOV R2,#2
       LED_LOOP1: MOV R3,#255
       LED_LOOP:
       CLR RI
       MOV P1,#01111110B
       MOV P2,#11111110B
       ACALL DELAY_LED
       MOV P1,#10001000B
       MOV P2,#11111101B
       ACALL DELAY_LED
       MOV P1,#10001000B
       MOV P2,#11111011B
       ACALL DELAY_LED
```

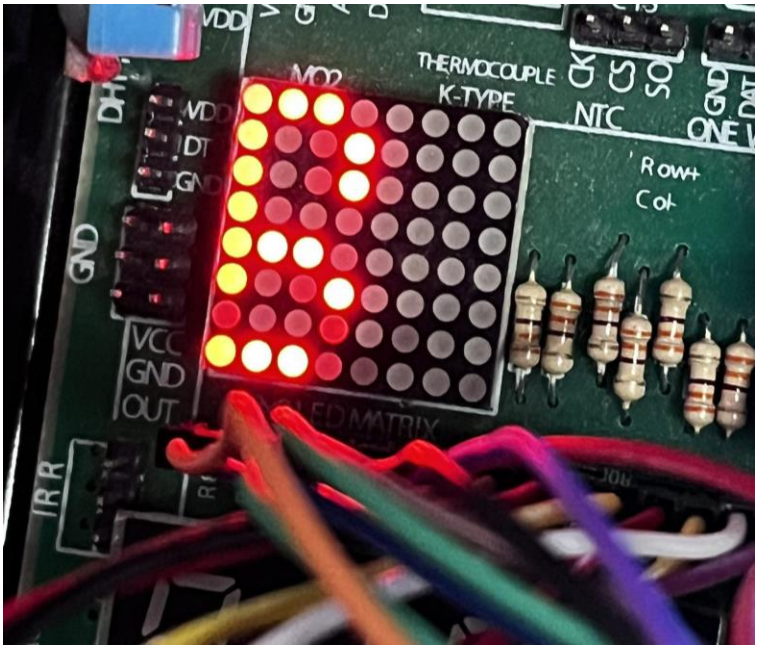
```
        MOV P1,#10001000B
        MOV P2,#11110111B
        ACALL DELAY_LED
        MOV P1,#01111110B
        MOV P2,#11101111B
        ACALL DELAY_LED
        DJNZ R3,LED_LOOP
        DJNZ R2,LED_LOOP1
        DJNZ R1,LED_LOOP2
        LJMP GO_PRE
LJMP GO_PRE
LED_B:
CJNE A,#'B',GO_PRE
        MOV R1,#1
        LED_LOOPB2: MOV R2,#3
        LED_LOOPB1: MOV R3,#255
        LED_LOOPB:
        MOV P1,#11111111B
        MOV P2,#11111110B
        ACALL DELAY_LED
        MOV P1,#10001001B
        MOV P2,#11111101B
        ACALL DELAY_LED
        MOV P1,#10001001B
        MOV P2,#11111011B
        ACALL DELAY_LED
        MOV P1,#01100110B
        MOV P2,#11110111B
        ACALL DELAY_LED
        DJNZ R3,LED_LOOPB
        DJNZ R2,LED_LOOPB1
        DJNZ R1,LED_LOOPB2
```

Demonstration

A. Display of ‘A’



B. Display of ‘B’



Encryption

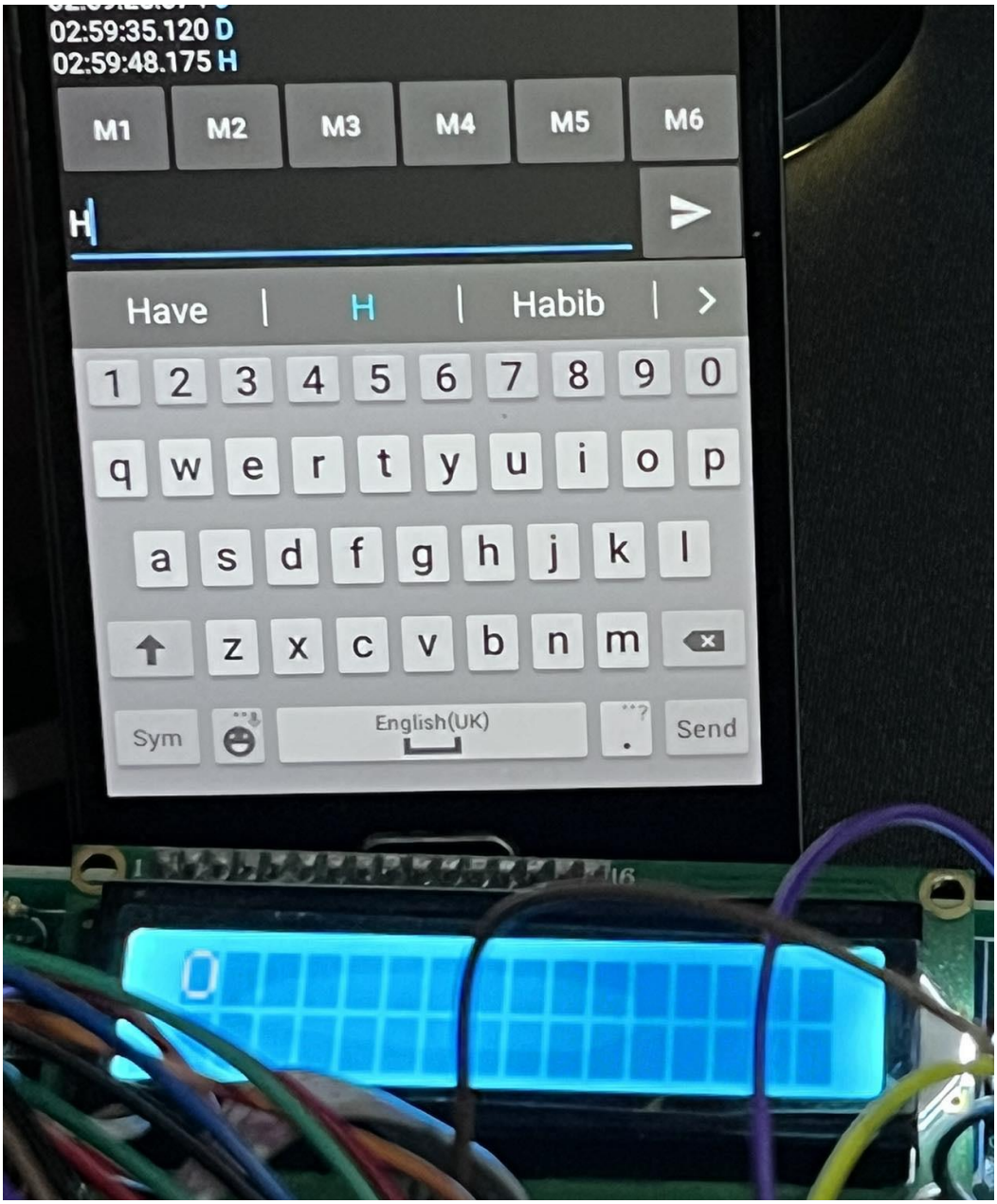
The encryption mode, activated by pressing 'D', transformed incoming messages by shifting their ASCII values by a decimal factor of 7. This shift entailed a cryptographic transformation, effectively encoding the transmitted information. For instance, a character 'A' (ASCII value 65) would be shifted by 7 positions, becoming 'H' (ASCII value 72). This method of encryption added a layer of security to the exchanged data, ensuring confidentiality and demonstrating a rudimentary yet functional cryptographic technique within the Bluetooth-8051 interface.

Code

```
CJNE A,#'D',CHECKN2
CLR RI ; register involved in receiving data from bluetooth and ensuring it
REP1: JNB RI, REP1
SJMP GO2
CHECKN2: LJMP CHECKNEXT2
GO2:
GOGO:
REP2: JNB RI, REP2
; preparing LCD
MOV A,#38H ; creatiing 2 lines and 5*7 matrix
ACALL COMNWRT
ACALL DELAY
MOV A,#0EH ; display on, cursor blinking
ACALL COMNWRT
ACALL DELAY
MOV A,#01H ; clear display skin
ACALL COMNWRT
ACALL DELAY
MOV A,#06H ; cursor shift right
ACALL COMNWRT
ACALL DELAY
MOV A,#0CH ; display on, cursor off
ACALL COMNWRT
ACALL DELAY
```

```
; writing the data from bluetooth
MOV A,SBUF  ; data from bluetooth stored in SBUF
ADD A,#07H
CJNE A,#'7',JAWAD
LJMP CONT1
JAWAD:
ACALL DATAWRT  ; takes data from app and prints it in LCD
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY
CLR RI ; register involved in receiving data from bluetooth and ensuring it
LJMP GOGO; to the start mode
```

Demonstration



LED Relay Control

The relay, an electrically operated switch, facilitated electrical isolation, ensuring safe and controlled operations. Through specific commands sent via Bluetooth, the microcontroller triggered the relay to either complete or interrupt the circuit, thus turning the LED on or off.

In this specific feature, after activating the LED relay control mode (triggered by the character '1'), the subsequent digit sent via Bluetooth determined the duration for which the relay controlled the LED. For instance, upon receiving '1' followed by '5' via Bluetooth, the microcontroller interpreted this sequence as an instruction to toggle the LED relay on and off five times consecutively, with each on-off cycle lasting for a predefined duration.

Code

```
CJNE A,#'1', CHECKN_RE
CLR RI
GO_RE:
REP3: JNB RI,REP3
SJMP GO4
CHECKN_RE: LJMP CHECKNEXT_RE
GO4:

MOV A,SBUF
CJNE A,#'0',JA_RE
LJMP CONT_RE
JA_RE:
ANL A, 0FH
MOV R2,A
RE_LOOP:
CLR P3.6
ACALL DELAY_RE
ACALL DELAY_RE
SETB P3.6
ACALL DELAY_RE
ACALL DELAY_RE
```

```
DJNZ R2, RE_LOOP
CLR RI
```

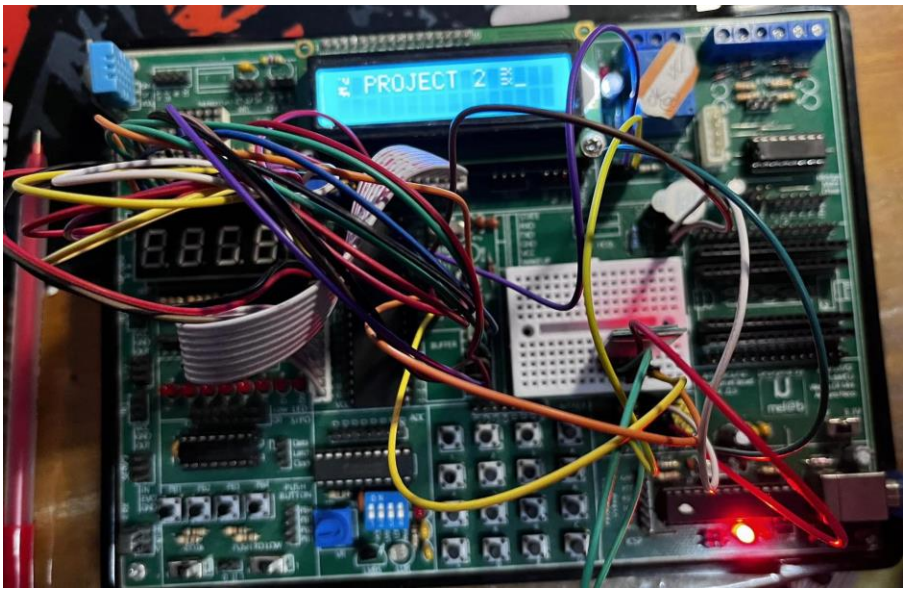
Demonstration of this part has to be shown via video or live demonstration.

EXIT Mode

By transmitting the character '0' over Bluetooth, the exit mode is activated, functioning as a global command to restore the system to its original or initial condition. When the 8051 microcontroller receives the value '0', it interprets this as a directive to stop any modes or operations that were previously initiated.

This function efficiently ended all existing activities, restoring the system to its original configuration, prepared to accept new instructions and function based on the default settings. The exit mode serves as a regulatory mechanism to cease current operations, facilitating a seamless transition to a neutral state and allowing the user to reset or terminate any active modes initiated by previous instructions.

Complete Hardware Setup



Problems Faced

1. We were unable to produce an LED matrix lookup table and thus, we were limited to the letters ‘A’ and ‘B’ respectively.
2. We were unable to produce an Morse Code lookup table and thus, we were limited to the letters ‘A’ and ‘B’ respectively.
3. A further and more thorough encryption could’ve been done and it is a work in progress for future development
4. ADC Implementation was halted due to the damaged LM35 sensor on the development board. However, the LDR sensor was able to be converted to a digital value in a separate file, although not integrated within the scope of this project.
5. The display and input from the Bluetooth could’ve been arranged in a more formal manner.

Final Discussion

In conclusion, the successful integration of Bluetooth technology with the 8051 Microcontroller yielded a multifaceted system capable of diverse real-time operations triggered by specific characters sent via a Serial Bluetooth Android app.

The project showcased the adaptability and versatility of the 8051 microcontroller in interpreting Bluetooth-transmitted commands to enact various functionalities. From controlling countdowns, LEDs, buzzers, and relays to implementing encryption, Morse code transmission, and display functions, each feature demonstrated the microcontroller's ability to interact with external components and execute complex actions in response to wireless commands. In the end, we want to conclude with the spirit of debugging unforgivable typing errors at the cost of wasting 3 days.

Complete Code

```
ORG 00H
CLR P2.5 ; LED connected
SETB P2.4 ; as buzzer connected
; for Bluetooth module
MOV TMOD,#20H ; timer 1 mode 2 is selected
MOV TH1,#0FDH ; baud rate
MOV SCON,#50H ; serial mode 1 10 bit total isn, 8db, 1STOPb
CLR TI ; making TI reg zero
SETB TR1 ; starting timer 1
;------------DISPLAY INITIALIZATION----------
       MOV DPTR,#MYLCD    ;DPTR stores the LCD initialization sequence
CIU1:   CLR A
       MOVC A,@A+DPTR
       LCALL COMNWRT
       LCALL DELAY
       JZ SIU1           ;Runs the rest of the code
       INC DPTR
       SJMP CIU1

SIU1:   MOV DPTR,#MSG1
DIU1:   CLR A
       MOVC  A,@A+DPTR
       LCALL DATAWRT
       LCALL DOT
       JZ SIU2           ;Runs rest of the code
       INC DPTR
       SJMP DIU1
SIU2:  MOV A, #01    ;Clear LCD
       ACALL COMNWRT ;Call command subroutine
       ACALL DELAY   ;Give LCD some time
       LCALL DATAWRT
       MOV DPTR,#MSG2
DIU2:  CLR A
```

```
       MOVC  A,@A+DPTR
       LCALL DATAWRT
       LCALL DOT
       JZ CONT1          ;Runs rest of the code
       INC DPTR
       SJMP DIU2

CONT1:
GOBACK:
CONT_RE:
CONT_M:
CONT_PRE:
CLR RI ; register involved in receiving data from bluetooth and ensuring it
REP: JNB RI, REP

; preparing LCD
MOV DPTR,#MYLCD2    ;DPTR stores the LCD initialization sequence
CIIU1:  CLR A
       MOVC A,@A+DPTR
       LCALL COMNWRT
       LCALL DELAY
       JZ SIIU1           ;Runs the rest of the code
       INC DPTR
       SJMP CIIU1
; writing the data from bluetooth
SIIU1:
MOV A,SBUF  ; data from bluetooth stored in SBUF
ACALL DATAWRT  ; takes data from app and prints it in LCD
ACALL DELAY

CJNE A,#'X',CHECKN_PRE
CLR RI
GO_PRE:
CLR RI
```

```
MOV P2,#00000000B
MOV P1,#00000000B
REP14: JNB RI,REP14
SJMP GO15
CHECKN_PRE: LJMP CHECKNEXT_PRE
GO15:
MOV A,SBUF
CJNE A,#'0',JA_PRE
MOV P2,#11111111B
MOV P1,#11111111B
LJMP CONT_PRE
JA_PRE:

CJNE A,#'A',LED_B
       MOV R1,#1
       LED_LOOP2: MOV R2,#2
       LED_LOOP1: MOV R3,#255
       LED_LOOP:
       CLR RI
       MOV P1,#01111110B
       MOV P2,#11111110B
       ACALL DELAY_LED
       MOV P1,#10001000B
       MOV P2,#11111101B
       ACALL DELAY_LED
       MOV P1,#10001000B
       MOV P2,#11111011B
       ACALL DELAY_LED
       MOV P1,#10001000B
       MOV P2,#11110111B
       ACALL DELAY_LED
       MOV P1,#01111110B
       MOV P2,#11101111B
       ACALL DELAY_LED
```

```
        DJNZ R3,LED_LOOP
        DJNZ R2,LED_LOOP1
        DJNZ R1,LED_LOOP2

        LJMP GO_PRE
LJMP GO_PRE

LED_B:
CJNE A,#'B',GO_PRE
        MOV R1,#1
        LED_LOOPB2: MOV R2,#3
        LED_LOOPB1: MOV R3,#255
        LED_LOOPB:
        MOV P1,#11111111B
        MOV P2,#11111110B
        ACALL DELAY_LED
        MOV P1,#10001001B
        MOV P2,#11111101B
        ACALL DELAY_LED
        MOV P1,#10001001B
        MOV P2,#11111011B
        ACALL DELAY_LED
        MOV P1,#01100110B
        MOV P2,#11110111B
        ACALL DELAY_LED
        DJNZ R3,LED_LOOPB
        DJNZ R2,LED_LOOPB1
        DJNZ R1,LED_LOOPB2

        LJMP GO_PRE


CHECKNEXT_PRE:
CJNE A,#'y',CHEKK
```

```
CLR P3.5



CHEKK:
CJNE A,#'n',CHEKK1
SETB P3.5

CHEKK1:
CJNE A,#'l', CHECKNEXT
SETB P3.6

CHECKNEXT:
CJNE A,#'L', CHECKNEXT1
CLR P3.6
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CHECKNEXT1:
CJNE A,#'1', CHECKN_RE
CLR RI
GO_RE:
REP3: JNB RI,REP3
SJMP GO4
CHECKN_RE: LJMP CHECKNEXT_RE
GO4:

MOV A,SBUF
CJNE A,#'0',JA_RE
LJMP CONT_RE
JA_RE:
ANL A, 0FH
MOV R2,A
RE_LOOP:
CLR P3.6
ACALL DELAY_RE
ACALL DELAY_RE
```

```
SETB P3.6
ACALL DELAY_RE
ACALL DELAY_RE
DJNZ R2, RE_LOOP
CLR RI
LJMP GO_RE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CHECKNEXT_RE:
CJNE A,#'M', CHECKN_M
CLR RI

GO_M:
CLR RI
REP4: JNB RI,REP4
SJMP GO5
CHECKN_M: LJMP CHECKNEXT_M
GO5:

MOV A,SBUF
CJNE A,#'0',JA_M
LJMP CONT_M
JA_M:

CJNE A,#'A',M_B
CLR P3.5
CLR P3.6
ACALL DOT
SETB P3.5
SETB P3.6
ACALL DELAY_RE
CLR P3.5
CLR P3.6
ACALL DASH
ACALL DASH
```

```
SETB P3.5
SETB P3.6
ACALL DELAY
CLR RI
LJMP GO_M

M_B:
CJNE A,#'B',GO_M
CLR P3.5
CLR P3.6
ACALL DASH
SETB P3.5
SETB P3.6
ACALL DELAY_RE
CLR P3.5
CLR P3.6
ACALL DOT
SETB P3.5
SETB P3.6
ACALL DELAY_RE
CLR P3.5
CLR P3.6
ACALL DOT
SETB P3.5
SETB P3.6
ACALL DELAY
CLR RI
LJMP GO_M

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CHECKNEXT_M:
CJNE A,#'D',CHECKN2
CLR RI ; register involved in receiving data from bluetooth and ensuring it
```

```
REP1: JNB RI, REP1
SJMP GO2
CHECKN2: LJMP CHECKNEXT2
GO2:
GOGO:

REP2: JNB RI, REP2
; preparing LCD
MOV A,#38H ; creatiing 2 lines and 5*7 matrix
ACALL COMNWRT
ACALL DELAY
MOV A,#0EH ; display on, cursor blinking
ACALL COMNWRT
ACALL DELAY
MOV A,#01H ; clear display skin
ACALL COMNWRT
ACALL DELAY
MOV A,#06H ; cursor shift right
ACALL COMNWRT
ACALL DELAY
MOV A,#0CH ; display on, cursor off
ACALL COMNWRT
ACALL DELAY

; writing the data from bluetooth

MOV A,SBUF  ; data from bluetooth stored in SBUF
ADD A,#07H
CJNE A,#'7',JAWAD
LJMP CONT1
JAWAD:
ACALL DATAWRT  ; takes data from app and prints it in LCD
ACALL DELAY
ACALL DELAY
```

```
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY

CLR RI ; register involved in receiving data from bluetooth and ensuring it


LJMP GOGO


CHECKNEXT2:
CJNE A,#'C', GOBACK1
LJMP GO3
GOBACK1: LJMP GOBACK

GO3:
MOV R1, #9

MOV R2, #39H


COUNT_LOOP:
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY
ACALL DELAY

MOV A,#38H
ACALL COMNWRT
ACALL DELAY
```

```
MOV A,#0EH
ACALL COMNWRT
ACALL DELAY
MOV A,#01H
ACALL COMNWRT
ACALL DELAY
MOV A,#06H
ACALL COMNWRT
ACALL DELAY
MOV A,#0CH
ACALL COMNWRT
ACALL DELAY
MOV A,R2
DEC R2
ACALL DATAWRT
ACALL DELAY
DJNZ R1, COUNT_LOOP

CLR P3.5 ; Buzzer on
ACALL DELAY1
SETB P3.5 ; after a certain time, turn it off

ACALL GOBACK



AGAIN: SJMP AGAIN



; for reading and writing in LCD subroutine and delay subroutine

COMNWRT: ; for command writing
MOV P0,A
CLR P3.2  ; RS=0
```

```
CLR P3.3 ; RW=0
SETB P3.4 ; EN=1
ACALL DELAY
CLR P3.4 ; EN=0 (high to low operation)
RET

DATAWRT: ; for data writing
MOV P0,A
SETB P3.2 ; RS=1
CLR P3.3 ; RW=0
SETB P3.4 ; EN=1
ACALL DELAY
CLR P3.4 ; EN=0(high to low)
RET

DATAENC:
ANL A, #07H
MOV P0,A
SETB P3.2 ; RS=1
CLR P3.3 ; RW=0
SETB P3.4 ; EN=1
ACALL DELAY
CLR P3.4 ; EN=0(high to low)
RET


; Delay subroutines
DELAY:
       MOV R3,#50
HERE2: MOV R4,#255
HERE: DJNZ R4, HERE
       DJNZ R3,HERE2
RET
```

```
DOT:
       MOV R2,#10
HERR_D1: MOV R5,#50
HERR_D2: MOV R6,#255
HERR_D3: DJNZ R6,HERR_D3
       DJNZ R5,HERR_D2
       DJNZ R2,HERR_D1
RET

DASH:
       MOV R2,#20
HERR_DD1: MOV R5,#50
HERR_DD2: MOV R6,#255
HERR_DD3: DJNZ R6,HERR_DD3
       DJNZ R5,HERR_DD2
       DJNZ R2,HERR_DD1
RET

;for buzzer alarm
DELAY1:
       MOV R2,#50
HERR1: MOV R5,#50
HERR2: MOV R6,#255
HERR3: DJNZ R6,HERR3
       DJNZ R5,HERR2
       DJNZ R2,HERR1
RET


DELAY_RE:
       MOV R3,#255
HERRR2: MOV R4,#255
HERRRE: DJNZ R4, HERRRE
       DJNZ R3,HERRR2
```

```
RET

DELAY_PRE:
       MOV R2,#100
HERR1_PRE: MOV R5,#255
HERR2_PRE: MOV R6,#255
HERR3_PRE: DJNZ R6,HERR3_PRE
       DJNZ R5,HERR2_PRE
       DJNZ R2,HERR1_PRE
RET

DELAY_LED:
              MOV    R6,#255D     ; 1ms delay subroutine
HERE_LED:     DJNZ   R6,HERE_LED
       RET

ORG 300H
MSG1: DB " GROUP 2 ",0
MSG2: DB " PROJECT 2 ",0
MYLCD : DB 38H,0EH,01,06,80H,0
MYLCD2 : DB 38H,0EH,01,06,0CH,0
END
```