

Presented By:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Name | G1 | G2 | G3 | D1 | D2 | D3 | D4 |
| 18106212 | Karim Ahmed Kohel |  |  |  |  |  |  |  |
| 18105818 | Mohamed Osama Badawy |  |  |  |  |  |  |  |
| 18104247 | Omar Tamer Rostom |  |  |  |  |  |  |  |

Smart Home

Department : Computer Engineering

Course Code : CC421

Course Name : Microprocessor Systems

Lecturer : Prof. Ahmed Fahmy

# Summary:

Standards of living are rising due to the rise of technology integration in everything around us in this day and age, which gave us the idea to develop a smart home system for a higher standard of living for the average consumer.

The system is both simple and easy to install in any home. The system is highly modular, to accommodate the preferences of each user. It relies on user input to achieve simple tasks around the house like turn on and off electric appliances, users can give commands to the system via a keypad to the main control unit of the system which then sends that command to the specific unit in charge of achieving the task specified by the user (e.g. the specific unit in charge of activating the coffee machine in another room).

# Introduction

The project relies on the concept of dividing tasks between µ-controllers, the master controller (Sender) is in charge of taking the input from the user using a 4x4 keypad array and sending it via a serial connection to all µ-controllers(Recivers) on the UART serial protocol supported by the 89S52 µ-controllers, while displaying the appropriate text to the user on the 16x2 LCD unit. Each controller then processes the incoming byte and decides if it should ignore the command or carry a task based on the specified set of commands the controller is responsible for handling (e.g. the controller in the living room is responsible for turning on or off the TV, room lights, ceiling fan and buzzer alarm system), which is demonstrated in the project as LEDs and buzzers. While the controller in the kitchen is responsble for controlling other home appliences (e.g. coffee machine and room lights). The µ-controllers however are not able to send out the necessary amount of power to turn on said appliences so we used transistors/relays to supply the needed power from another power source when given the signal from the µ-controllers.

# Methodology

The [Sender] µ-controller starts by initializing the serial communication UART protocol on port 3.0 and 3.1 (TX and RX), then initialize the 16x2 LCD display on port 2. then enters an infinit loop for checking input on the 4x4 keypad by comparing row to column and what button is pressed based on the combination of row and columns activated.

If a button is pressed the LCD will display the appropriate message to the user from the stored messages in the data segment, then the byte of data indicating which button was pressed is sent off to the other µ-controllers [Recievers].

The Recieving µ-controller are pre-loaded each with a set of codes that corespond to each task depending on which button was pressed.

The Recieving µ-controllers start by initializing the serial communication UART protocol and enter a loop where it waits for data to be recieved and breaks out of said loop to carry out the task then re-enters back the loop to wait for the next message to be recieved.

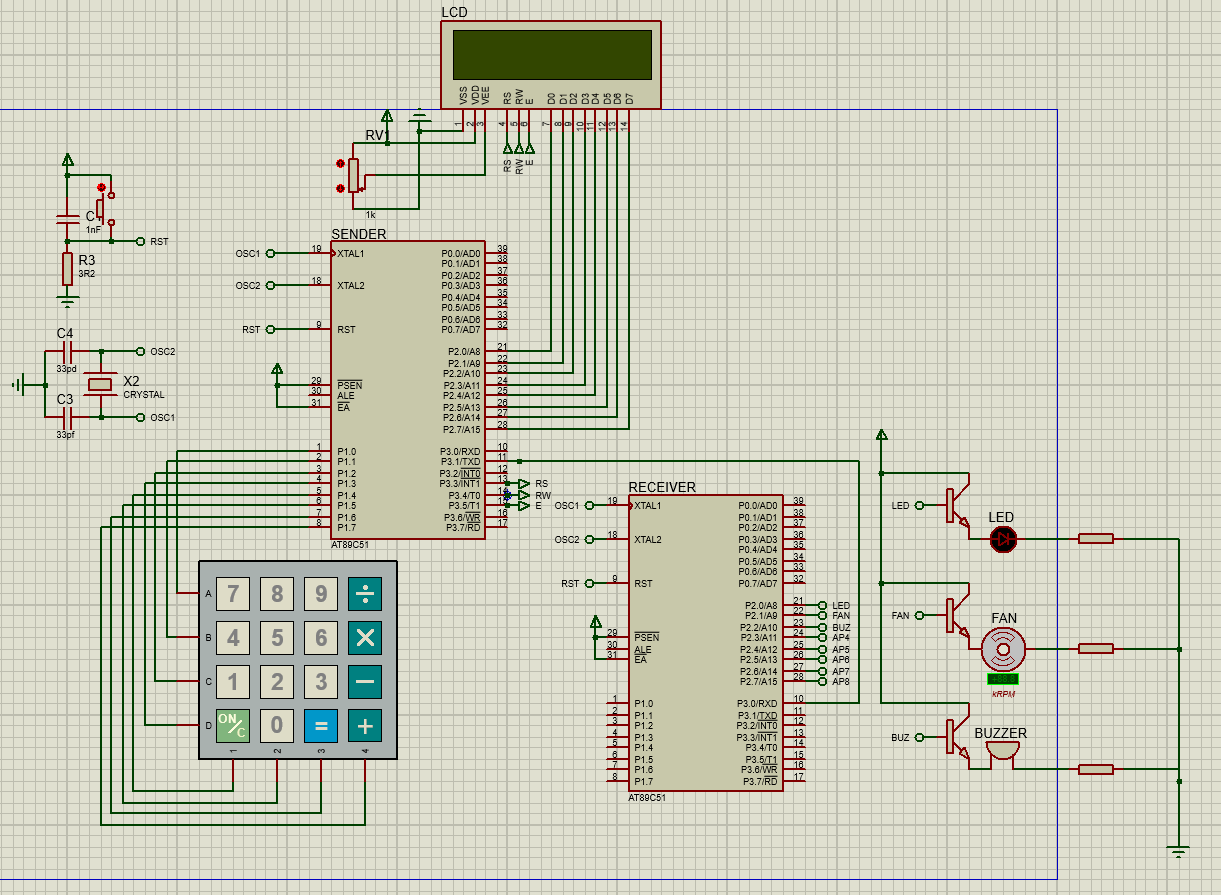
The Recieving µ-controller compare recieved messages to the stored set of codes that indicate what task is required of them and only do that task based on the state at which the µ-controller is in, for instance if the task was to toggle the LED light on or off, it will first check the state of the LED and depending on it’s previous state it will either turn it on or off.

The user is faced with the output of the main LCD and the tasks being carried at the same time to offer feedback for the input being taken in case the output task is running in another room.

The power output of the µ-controllers are insufficiant for powering up anything ,thus the signal taken from the µ-controller must activate a transistor/relay controlling the appliance to be activated.

This prototype includes 2 µ-controllers to act as Sender and Reciever, the user can add µ-controllers to meet their specific needs asthere is not limit to the number of µ-controller to be added on the serial line.

# Schematic:



# Components and Tools:

* 2, 89S52 µ-controllers
* 1, 16x2 LCD module
* 1, 4x4 keypad array
* 3, 2n2222a transistors
* 1, ULN2003AN motor driver
* 2, 11.0592 MHz crystals
* 4, 320 ohm current limiting resistors
* 1, potentiometer 1k ohm
* 2, low profile push buttons
* 1, 5v power supply
* 2, 100 µf capacitors
* 4, 33 µf capacitors
* 1, 5v dc fan
* 1, led
* 1, buzzer

# Code Illustration:

**Sender**

# Recievers

# 

# Code Text:

## Sender:

ORG 0

; Serial initialization

MOV TMOD, #20H ; or 00100000B => Mode 2 for Timer1 (8bit Auto Reload)

MOV TH1, #0FDH ;Setting BaudRate of 9600 (-3). SMOD is 0 by default

MOV SCON, #50H ;Serial Mode 1, REN Enabled or 01010000B

SETB TR1

RS BIT P3.3

RW BIT P3.4

E BIT P3.5

;LCD INITIALIZATION

MOV A, #38H ; INITIATE LCD ; 00 0011 1000 ; sets interface length , number of lines, and font

ACALL COMMANDWRT

ACALL DELAY

MOV A, #0CH ; DISPLAY & CURSOR ON ; 00 0000 1100

ACALL COMMANDWRT

ACALL DELAY

ACALL CLRLCD ; CLEAR LCD AND WELCOME

MOV DPTR, #WLC

ACALL STRING

; initializing and reading keypad continuously

BACK:

MOV P1,#11111111B ;// loads P1 with all 1's

CLR P1.0 ;// makes row 1 low

JB P1.4,NEXT1 ;// checks whether column 1 is low and jumps to NEXT1 if not low

MOV A,#'1' ;// loads a with 0D if column is low (that means key 1 is pressed)

ACALL SEND ;// calls SEND subroutine

ACALL CLRLCD

MOV DPTR, #LED

ACALL STRING

NEXT1:

JB P1.5,NEXT2 ;// checks whether column 2 is low and so on...

MOV A,#'2'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #FAN

ACALL STRING

NEXT2:

JB P1.6,NEXT3

MOV A,#'3'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #BUZZER

ACALL STRING

NEXT3:

JB P1.7,NEXT4

MOV A,#'A'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #SAPA

ACALL STRING

NEXT4:

SETB P1.0

CLR P1.1

JB P1.4,NEXT5

MOV A,#'4'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #AP4

ACALL STRING

NEXT5:

JB P1.5,NEXT6

MOV A,#'5'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #AP5

ACALL STRING

NEXT6:

JB P1.6,NEXT7

MOV A,#'6'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #AP6

ACALL STRING

NEXT7:

JB P1.7,NEXT8

MOV A,#'B'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #SAPB

ACALL STRING

NEXT8:

SETB P1.1

CLR P1.2

JB P1.4,NEXT9

MOV A,#'7'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #AP7

ACALL STRING

NEXT9:

JB P1.5,NEXT10

MOV A,#'8'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #AP8

ACALL STRING

NEXT10:

JB P1.6,NEXT11

MOV A,#'9'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #AP9

ACALL STRING

NEXT11:

JB P1.7,NEXT12

MOV A,#'C'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #SAPC

ACALL STRING

NEXT12:

SETB P1.2

CLR P1.3

JB P1.4,NEXT13

MOV A,#'\*'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #SAPS

ACALL STRING

NEXT13:

JB P1.5,NEXT14

MOV A,#'0'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #ZERO

ACALL STRING

NEXT14:

JB P1.6,NEXT15

MOV A,#'#'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #SAPT

ACALL STRING

NEXT15:

JB P1.7,DONEKEYPAD

MOV A,#'D'

ACALL SEND

ACALL CLRLCD

MOV DPTR, #SAPD

ACALL STRING

LJMP BACK

DONEKEYPAD:

LJMP BACK

SEND:

; initializing transmission

;MOVC A,@A+DPTR

;MOV P2,A

MOV SBUF, A

JNB TI, $

CLR TI

ACALL DELAY

ACALL DELAY

RET

STRING:

CLR A

MOVC A, @A+DPTR ; move char from data in A register we khalas

ACALL DATAWRT

INC DPTR

JZ FINISHED

SJMP STRING

FINISHED:

RET

DELAY:

MOV R0, #255 ;DELAY. HIGHER VALUE FOR FASTER CPUS

Y: MOV R1, #255

DJNZ R1, $

DJNZ R0, Y

RET

CLRLCD:

MOV A, #01H ; CLEAR LCD ; 00 0000 0001 ; clears ram in lcd

ACALL COMMANDWRT

ACALL DELAY

RET

;COMMAND SUB-ROUTINE FOR LCD CONTROL

COMMANDWRT:

MOV P2, A ;SEND DATA TO P1

CLR RS ;RS=0 FOR COMMAND

CLR RW ;R/W=0 FOR WRITE

SETB E ;E=1 FOR HIGH PULSE

ACALL DELAY ;SOME DELAY

CLR E ;E=0 FOR H-L PULSE

RET

DATAWRT:

MOV P2, A

SETB RS ;;RS=1 FOR DATA and rw=0 for write data command from data lines

CLR RW

SETB E

ACALL DELAY

CLR E

RET

ORG 300H

WLC: DB "WELCOME", 0

ZERO: DB "ALL OFF", 0

LED: DB "LED" ,0

FAN: DB "FAN" ,0

BUZZER: DB "BUZZER", 0

AP4: DB "AP4", 0

AP5: DB "AP5", 0

AP6: DB "AP6", 0

AP7: DB "AP7", 0

AP8: DB "AP8", 0

AP9: DB "AP9", 0

SAPA: DB "SAPA", 0

SAPB: DB "SAPB", 0

SAPC: DB "SAPC", 0

SAPD: DB "SAPD", 0

SAPS: DB "SAPS", 0

SAPT: DB "SAPH", 0

END

## Recievers

org 00H

LED EQU P2.0

FAN EQU P2.1

BUZ EQU P2.2

AP4 EQU P2.3

AP5 EQU P2.4

AP6 EQU P2.5

AP7 EQU P2.6

AP8 EQU P2.7

; Serial initialization

SERIAL\_INIT:

MOV TMOD, #20H ; or 00100000B => Mode 2 for Timer1 (8bit Auto Reload)

MOV TH1, #0FDH ;Setting BaudRate of 9600 (-3). SMOD is 0 by default

MOV SCON, #50H ;Serial Mode 1, REN Enabled or 01010000B

SETB TR1

; PORT 2 INIT

MOV P2, #0H

; RECIEVE INCOMING BYTE IN LOOP

RECIEVE:

JNB RI, $

MOV A, SBUF

CLR RI

MOV R5, A

MOV P1, R5

; TURN OFF EVERYTHING

COMPARE0:

MOV A, R5

SUBB A, #'0'

JNZ COMPARE1

CLR LED

CLR FAN

CLR BUZ

CLR AP4

CLR AP5

CLR AP6

CLR AP7

CLR AP8

LJMP RECIEVE

; COMPARE RECIEVED BYTE TO KNOWN COMMANDS AND EXECUTE IF EQUAL

; OR GO TO NEXT ONE IF NOT

COMPARE1:

MOV A, R5

SUBB A, #'1'

JNZ COMPARE2

JB LED, LEDOFF

SETB LED

LJMP RECIEVE

LEDOFF:

CLR LED

LJMP RECIEVE

COMPARE2:

MOV A, R5

SUBB A, #'2'

JNZ COMPARE3

JB FAN, FANOFF

SETB FAN

LJMP RECIEVE

FANOFF:

CLR FAN

LJMP RECIEVE

COMPARE3:

MOV A, R5

SUBB A, #'3'

JNZ COMPARE4

JB BUZ, BUZZOFF

SETB BUZ

LJMP RECIEVE

BUZZOFF:

CLR BUZ

LJMP RECIEVE

COMPARE4:

MOV A, R5

SUBB A, #'4'

JNZ COMPARE5

JB AP4, AP4OFF

SETB AP4

LJMP RECIEVE

AP4OFF:

CLR AP4

LJMP RECIEVE

COMPARE5:

MOV A, R5

SUBB A, #'5'

JNZ COMPARE6

JB AP5, AP5OFF

SETB AP5

LJMP RECIEVE

AP5OFF:

CLR AP5

LJMP RECIEVE

COMPARE6:

MOV A, R5

SUBB A, #'6'

JNZ COMPARE7

JB AP6, AP6OFF

SETB AP6

LJMP RECIEVE

AP6OFF:

CLR AP6

LJMP RECIEVE

COMPARE7:

MOV A, R5

SUBB A, #'7'

JNZ COMPARE8

JB AP7, AP7OFF

SETB AP7

LJMP RECIEVE

AP7OFF:

CLR AP7

LJMP RECIEVE

COMPARE8:

MOV A, R5

SUBB A, #'8'

JNZ HLT

JB AP8, AP8OFF

SETB AP8

LJMP RECIEVE

AP8OFF:

CLR AP8

LJMP RECIEVE

HLT:

LJMP RECIEVE

END

# References:

* 8051 instructions
  1. <https://www.keil.com/support/man/docs/is51/is51_instructions.htm>
* 89S52 datasheet
  1. <https://www.keil.com/dd/docs/datashts/atmel/at89s52_ds.pdf>
* Flashing 89S52 with avrdude linux tool
  1. https://www.instructables.com/How-to-Program-8051-Using-Arduino