20COA202 Coursework

*id number*

Semester 2 2020-2021

*In this template delete all the text in italics and replace with your own as appropriate.*

# 1 Introduction

*Briefly describe your implemetation including what is coming in the following sections.*

*Provide a list of instructions as to how to operate your code given someone is holding the Arduino with the code running at first boot*.

# 2 Your code - base implementation

*Do not write anything here–put it in the subsections following*

## 2.1 Data structures

The house descriptors will have their values stored in char Arrays:

* floorRoom={'F','G','O','S'}
* firstFlrRoom={'1','2','B'}
* groundRoom={'K','H','L}
* outsideRoom={'G','R}
* typeSetting={'L','A','H'}
* gardenSetting={'L','A','H','W'}
* deviceName={'M'}
* actionSetting={'1','0','L}

Each possible option that can be selected from the menu has a unique key within its own Array.

Another key data structure that is used, but briefly is the char\*\* house variable which is a pointer that points to the other char Arrays:

House {floorRoom, firstFlrRoom, …}

The char arrays will hold the key letter for each descriptor, this method was chosen for memory basis as opposed to choosing words stored in strings. The amount of memory the address’ used is known and it can also be manipulated easily.

Functions:

Void writeEE():

This is a function is intended to write the values from the data structures into the eeprom (if the data has not been written already) so that whenever the program is run, the data is read from the EEPROM directly and does not need to be reloaded. The function takes no parameters and returns no value hence the declaration as void. The function first instructs the eeprom to update address 0 with the value 1 , this is to signify that the eeprom has been written to and if so(if it has been written to ) then there is no need to call the write function.

Using hard coded char arrays that contain the house descriptors, i.e. floorRoom[6]={'F','G','O','S','-'} and actionSetting[5]={'1','0','L','x'}, the function stores each of these char arrays in the char\*\* variable ‘house’ (defined above) and loops through this data structure in order to write each single char along with a consecutively increasing address into the eeprom.

Void readEE(char\*\*):

A function designed to read data from the eeprom and store it inside multiple char arrays. The argument required is a char\*\* which will again be the house variable that stores empty house descriptor char arrays.

The pre defined char arrays that exist in the writeEE function, and have their values written to the EEPROM, make use of either ‘-‘ or ‘x’, meaning end of char array but not the final char array to read in the structure, and final char array to read in the data structure respectively. This allows the nested loop to know, when reading from the EEPROM, a dash ‘-‘ means the loop shoud move to the next char array i.e house[i][j] to house[i+1][j], and an ‘x’ means means the end of read and so the loop can break. The char values read from the EEPROM that represent house descriptors will subsequently be written into the empty char Arrays.

void printLcdSettings (char\* setting, int option, int num\_setting)

The purpose of this function is to print out the full house description to the lcd based on the key. Using switch statements each case has hard coded Strings to print to the lcd depending on the key. i.e for case 6 is the key is equal to ‘H’, the lcd is to print Heat. The arguments taken are: **char\*** for the char array that will contain the house descriptors, **int** which is the index of a house key of parameter **char\*** setting, and another **int** which specifies a switch case number.

int navSettings(char\* setting, int num\_setting)

This function is used to navigate the menu. It takes arguments char\* and int , these are passed into the function printLcdSettings where their purpose is explained above. Using left and right buttons on the Arduino the function is able to navigate the menu by incrementing or decrementing an int variable select respectively, which also acts as an index and so when the function printLcdSettings is called, the index is passed as the parameter int option, the char array is passed and char\* setting and the enum value of current menu options, i.e. floors, is passed as int num\_setting. This is run in a while loop which creates the menu navigation system. Once the select button is pressed the loop ends and the index of the chosen options house description key is returned.

int sumIterations(char\* firstIter,char\* nestedIter,char\* secondNestedIter)

This function takes 3 char \* arguments. The reason for this function is to create a value by summing the indexes within a triple nested loop. This value is used in the creation of a unique address by dividing the result value by the length of the char array of the first loop (first iter). In this home system that will be the type setting: water, heat… This provides a unique range for each type setting and so the creation of this value is a crucial step in creating a unique address.

int calculateAddressValue(char\* setting,int sumTotal,int typeChoice,int nameChoice,int actionChoice)

This function creates the unique address. The argument char\* is used the divide the return value of sumIterations by its length thus giving a value to create a unique range as aforementioned in the previous function explanation. Other arguments namely int sumTotal, typeChoice, nameChoice, actionChoice are used in the generation of the unique address. typeChoice is multiplied by the created range in order to create the range for that given choice.

Then this formula is used:

modValue=sumTotal/strlen(setting)

modProduct=(typeChoice\*modValue)

(modProduct+(typeChoice+1)+(nameChoice+1)+ actionChoice) **mod**  (modValue\*(actionChoice+1)+1)

This creates a unique value for the memory address.

## 2.2 FSMs

*Describe the finite state machine at the centre of your implementation. Show what states there are and the transitions. Draw the states and transitions as a picture and include it here.*

*You can use Visual Paradigm to draw a State Machine. Alternative ways are described on LEARN.*

*If there are other (sub) FSMs in your code then indicate those here.*

*Include any rationale for deciding what states to include.*

Diagram

Description automatically generated

## 2.3 Testing and testing

*Describe your approach to debugging. Include any code you have included to assit with debugging.*

*Describe any code that exists purely to test other aspects of your program.*

My approach to debugging is using #ifdef DEBUG statements that print to the serial monitor to check the correct values are being written and read by the program.

1. As my data structure for the house descriptors will be char arrays with the first letter of a descriptor being used as the char identifier i.e ‘F’ =Floor ,and so i have created a for loop to print out the char array alongside the memory location to check if it was written correctly so that when it is stored in the eeprom the correct data goes in the correct place.
2. Once the data has been read from the EEPROM via the function readEE(), there is a #ifdef DEBUG statement that will print out the char arrays to check if data was entered correctly.

# 3 Extension Features

*For each extension feature you have implemented describe the additional code you have included. Give examples of types, variables and code that is important. Start each in a new subsection.*

*do not write anything here–put it in the subsections following*

## 3.1 LAMP

*your text here.*

*Write “NOT IMPLEMENTED” if not implemented.*

## 3.2 OUTSIDE

*your text here.*

*Write “NOT IMPLEMENTED” if not implemented.*

## 3.3 QUERY

*your text here.*

*Write “NOT IMPLEMENTED” if not implemented.*

## 3.4 MEMORY

*your text here.*

*Write “NOT IMPLEMENTED” if not implemented.*

## 3.5 SOFT

*your text here.*

*Write “NOT IMPLEMENTED” if not implemented.*

## 3.6 EEPROM

*your text here.*

*Write “NOT IMPLEMENTED” if not implemented.*

# 4 Conclusions

*Reflect on what is fully working and what only partially working. Include a description of the parts of the code that are particulary worth mentioning.*

# 5 Submission

*Prepare the report as a PDF.*

## 5.1 From Word source

*If you have prepared this using the Word template then use the styles Heading 1 and Heading 2 for each section and subsection. It should create a new page for each Heading 1 and Heading 2. Please check this is the case.*

## 5.2 From Markdown source

*If you are preparing this in markdown, then I applaud you. To convert to a PDF use the pandoc and LaTeX software (available from* [*https://pandoc.org/*](https://pandoc.org/) *and* [*https://tug.org/texlive/*](https://tug.org/texlive/)*)*

pandoc -No output.pdf --template=coa202.latex input.md --shift-heading-level-by=-1

*coa202.latex is available from LEARN. This works for me with pandoc version 2.11.4.*

## 5.3 Gradescope Tagging

*There will be instructions on tagging with Gradescope*