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

*Describe the data strcutres you are using to implement the Coursework.*

*Include the types, the variables and any functions to update the information held in the structure.*

**Data Structure:**

The data structure I will be implementing for my house descriptors will be char arrays.

This will include:

int HOUSE\_LEN=10;

* char floors[5]={'F','G','O','-'};
* char first[6]={'1','2','B','P','-'};
* char ground[7]={'K','H','L','3','4','-'};
* char outside[4]={'G','R','-'};
* char type[6]={'L','A','H','W','-'};
* char names[8]={'M','C','D','B','P','W','-'};
* char qualifier[7]={'1','2','3','L','R','-'};
* char action[5]={'1','0','L','-'};
* char val[4]={'T','R','x'};
* char\* house[HOUSE\_LEN]={floors,first,ground,outside,type,names,qualifier,action,val};

The char arrays will hold the key word for each descriptor , I chose this method more for memory basis as I know exactly how much memory the array uses, and I will be making use of indexing and so this was chosen over using a linked list.

**Functions:**

Void writeEE():

This is a function to write the values from the data structures into the eeprom in order for it to store and remember the house descriptors. The eeprom stores a 1 as its first value when the data is initially written so that a check can be called thus if a 1 is in eeprom location 0 then no need to write the data and the program can move on to readEE() and read the data.

Void readEE(char\*\*):

This function reads data from the EEPROM and stores it inside different char arrays. It takes char\*\* house as a parameter which is an array that holds char arrays (or a pointer that points different memory locations).

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

This function is to allow navigation of the different house descriptions through the use of buttons and printing the house description to the lcd based on the house desc key read from the EEPROM.

Using the left and right buttons the different house settings can be navigated through.

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

This function returns the index of the house description that is selected according to the display shown from the navSettings function.

## 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.*

## 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*