

Molly Scarlet DIS Assignment

by Molly Scarlet

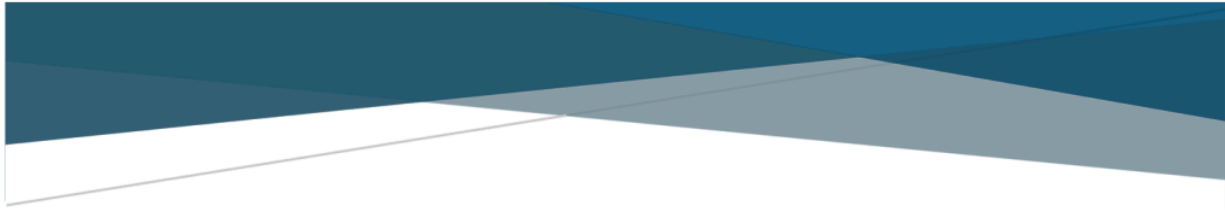
Submission date: 13-Nov-2024 02:22PM (UTC+0000)

Submission ID: 242542825

File name: 41243_Molly_Scarlet_Molly_Scarlet_DIS_Assignment_523660_1004071505.docx (442.56K)

Word count: 2457

Character count: 13571



ACDA5018 DISTRIBUTED INTELLIGENCE SYSTEMS

Assignment – A simple token based distributed
network

Abstract

Lab report detailing design and development of a simple token based distributed network
using C programming on PIC microprocessors using simulation software

Molly Scarlet

2216419@student.uwtsd.ac.uk

Contents

Introduction	4
Task 1 – Research	4
Quality Management Systems	4
Point to Point Transmission System	4
Point to Multipoint Transmission System	4
Multi-Master Transmission System	5
Task 2 – Proteus Hardware Design	6
Overview	6
Buttons	7
Lights/Horn	7
PIC Microcontrollers	8
RS232 Connection	8
Task 3 – PIC Programming	9
Master	9
Slave	10
Program Flow	11
Conclusion	12
Analysis of Work	12
References	13

Figure 1 Proteus Schematic.....	6
Figure 2 Proteus schematic of buttons	7
Figure 3 Proteus schematic of LEDs and buzzer	7
Figure 4 Proteus schematic of PIC microprocessors	8
Figure 5 Proteus schematic of RS232 connection	8
Figure 6 Embedded C code from master PIC	9
Figure 7 Embedded C code from master PIC	9
Figure 8 Embedded C code from master PIC	10
Figure 9 Embedded C code from master PIC	10
Figure 10 Embedded C code from master PIC	10
Figure 11 Embedded C code from master PIC	11
Figure 12 Embedded C code from master PIC	11
Figure 13 Embedded C code from master PIC	12
Figure 14 Embedded C code from master PIC	12

Introduction

This lab report will cover the design and development of a distributed intelligence system, demonstrated through the context of controlling the lights found on a car. All stages including research, hardware design, software development, and post-development analysis will be covered within the report.

Task 1 – Research

Quality Management Systems

Quality management systems are very important for a variety of reasons to the point where most would consider them necessary. By reporting and recording all the different aspects of the work, a business can analyse and determine the relevant improvements and updates that need to be made from a business, technical, financial and safety perspective. Businesses with poor quality management systems will often have frequent and/or repeat incidents, poor efficiency or other issues with delivery of the business goals which can be shown in a number of ways [1]. Especially in regulated businesses, a good quality management system is essential for ensuring that the business performs properly and more importantly safely and to the required regulatory standards in order to keep operating [2].

Point to Point Transmission System

At its core, a point to point transmission system is simply one node or endpoint communicating directly with another node or endpoint. The two systems can communicate bi-directionally, sharing data both ways whether it is symmetric or asymmetric. This form of transmission system is usually fairly simple and is extremely good for applications that require uninterruptible and high-speed transmission between them, as the transmission speed is only limited by the input/output speeds of the sender and receiver, as well as the transmission speed of the cable itself [3].

However, whilst they do have its advantages, point to point transmission systems do not have any redundancy should one of the nodes, endpoints or the cable itself fail. This means that the infrastructure from both a hardware and software side must be incredibly robust in order to maintain appropriate uptime. It is also not easily expandable, as any expansion to the transmission system requires significantly more infrastructure and complex management per device added [4].

Point to Multipoint Transmission System

A point to multipoint transmission system always comprises of multiple nodes or endpoints, with one set up as a transmitter or master node. The other devices on the network are then considered receivers or slaves. The data is sent out from solely the master transmitter node which is the central node within the system and the receiver slaves can then send the data back [5].

Point to multipoint transmission systems are popular in a variety of applications, particularly telecoms as the quantity of users of cellular and Wi-Fi devices is extremely large and the use of a single transmitter node (usually in the form of something like a cell tower or modem) which can communicate with potentially thousands of devices or more means that relatively little infrastructure is required compared to using something like point to point transmission systems [6].

Despite this, point to multipoint transmission systems are not perfect. Due to the reliance on a single central node which has to have the capability to transmit to potentially a very large number of

devices the upfront set up costs can be relatively quite high. Additionally, when one of the receiver/slave nodes failing it will not affect any of the other nodes, but if the central node fails the entire system will collapse. There are also further potential security and interference vulnerabilities due to the interconnected nature of it [7].

Multi-Master Transmission System

A multi-master transmission system builds further upon the previous transmission systems but instead of having only one master and one or multiple slaves, the system uses multiple masters to control and transmit data. This allows any of the master nodes to control the transmission of data across the network [8].

Multi-master transmission systems are popular for some applications as they are able to counteract the single point of failure within point to multipoint transmission systems, as if one of the master nodes fails in a multi-master, a different master node can take over. This is incredibly important for mission critical applications such as in health and safety applications where failure may result in severe consequences such as death [9].

The primary downside of multi-master transmission systems comes down to the complexity of the system. In order for multiple masters to communicate on the same network, each device must poll whether the transmission line is in use before any transmissions occur. This process can be complicated to set up and errors may occur. Additionally, where multiple masters are used the clock frequencies must all be the exact same in order to communicate on the same line. This process can also be complex and require additional functionality [10].

Furthermore, due to the fact that there are multiple master devices this inherently increases the cost as more equipment is required and more time required to set up all the infrastructure [11].

Task 2 – Proteus Hardware Design

Overview

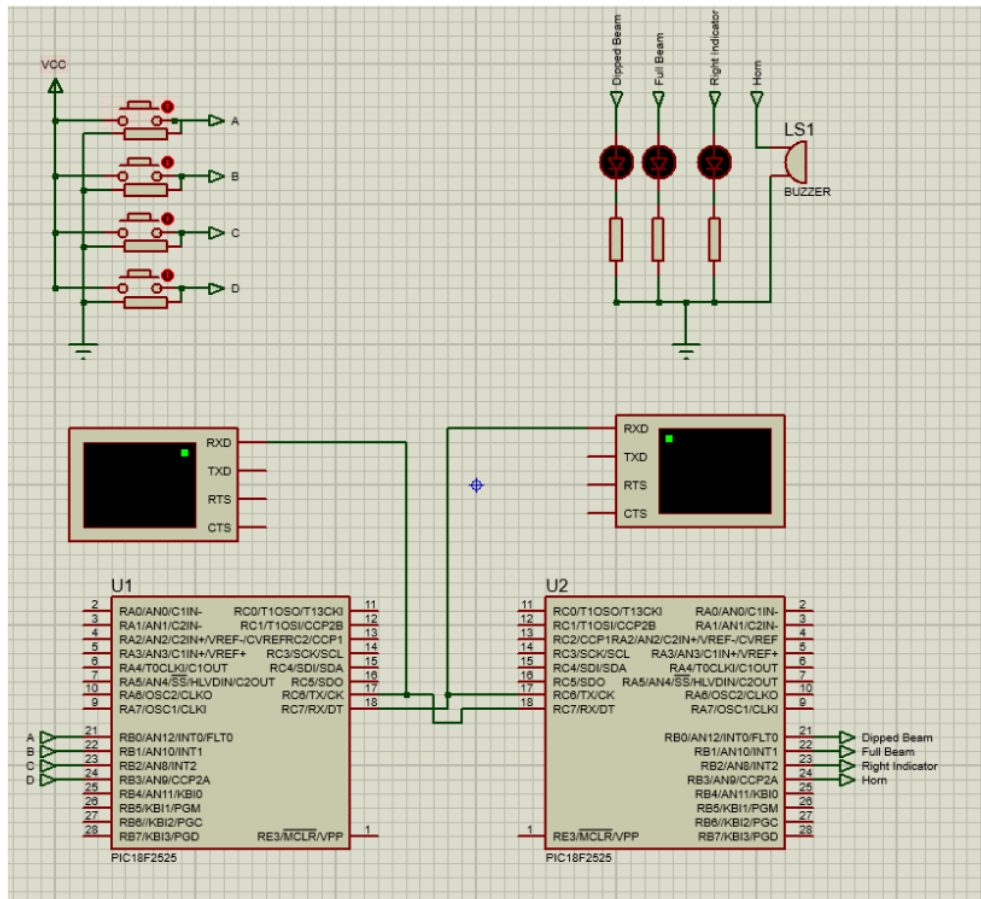


Figure 1 Proteus Schematic

Shown in Figure 1 is an image of the entire hardware schematic, designed and built within Proteus simulation software. This software will allow testing and analysis of the fundamentals of the software developed later.

Buttons

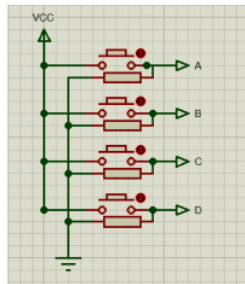


Figure 2 Proteus schematic of buttons

In order to actually enable the lights on the car model, a number of buttons are required as inputs for the user to press, just like in a real car. Within the schematic there are 4 buttons to correspond to the number of lights that can be activated. Each button also has a 10k Ohm resistor attached in order to pull the line low so that it will trigger appropriately.

Lights/Horn

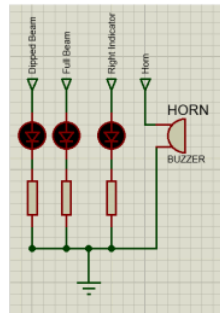


Figure 3 Proteus schematic of LEDs and buzzer

The lights themselves then are connected to both a ground and to input markers which take the output from the slave PIC and activate the lights when specified. Each light input marker is labelled based on the function. For this assignment, the front right corner of a car was used. This includes 3 lights – dipped beams, full beams and the right indicator – as well as the horn which in this assignment will be represented by a buzzer.

PIC Microcontrollers

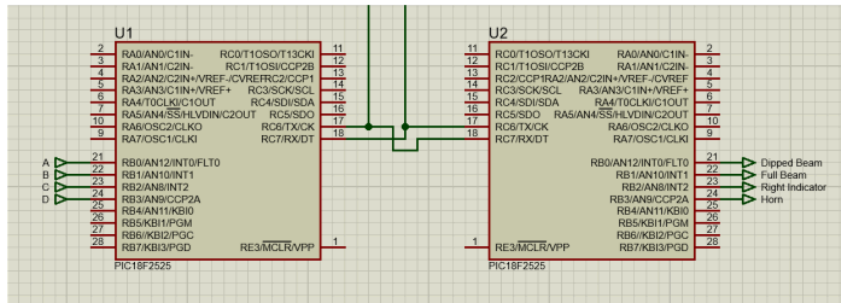


Figure 4 Proteus schematic of PIC microprocessors

In order to control the components, two PIC microprocessors were used. Whilst any PIC microprocessors could have been used, two PIC18F2525's were used. This PIC has support for a variety of communication methods such as I2C and RS232 which will be used for the communication between the microprocessors [12].

The PIC on the left in Figure 4 above is the master PIC which the buttons are connected to on port B. The RS232 connection then go through port C to the second PIC which is the slave. This has the lights connected to its own port B.

RS232 Connection

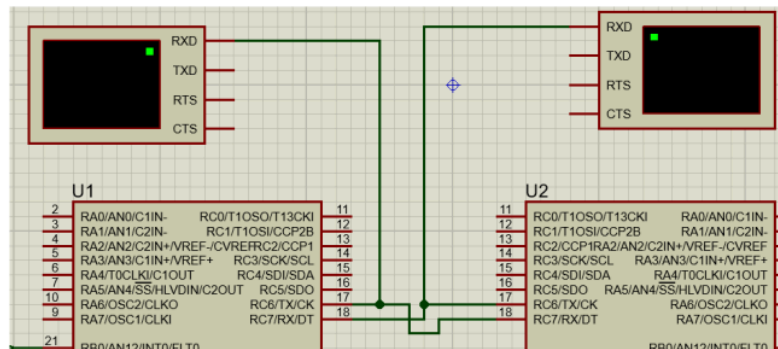


Figure 5 Proteus schematic of RS232 connection

In this model, in order to fulfil the “distributed” part of distributed intelligence systems, an RS232 connection was used in order to transfer data between the two devices. I2C was also a potential option for this functionality, using a flash memory chip as a central memory storage to maintain the token state. This would have enabled more redundancy and also make the system more expandable as further devices could be connected to the flash memory module to expand the number of potential systems connected, or alternatively to enable larger amounts of data to be stored such as sensor data.

For this assignment however, an RS232 connection was used instead of an I2C connection due to the relatively low complexity and lack of expansion needed. It is a common engineering principle to not make systems more complicated than necessary so using a simpler but perhaps slightly less reliable RS232 connection is suitable for this application [13].

A pair of virtual terminals were then attached to the RS232 transmission lines as a debug measure to monitor the data transferred between.

Task 3 – PIC Programming

Master

```
1 #include <main.h> //
2
3 char read_buttons(void);
4
5 char local_output; //
6 char txbyte; //define t
7
8 void main() //loop for
9 {
10     char buts=0;
11
12     while(TRUE) //
13     {
```

Figure 6 Embedded C code from master PIC

This beginning part of the code begins by defining a number of important variables, including line 1 which imports the main.h config file ensuring that the PIC has all of the relevant fuse configurations etc. correctly set. The other variables defined are used to store data at various parts of the program elsewhere. The main program loop is then initialised before an infinite while loop started in order to keep the program constantly checking for button inputs.

```
15     buts=read_buttons(); //store the button state in the va
16     switch (buts)
17     {
18         case 1: if(local_output & 0x01)
19         {
20             local_output-=1;
21             txbyte='a';
22         }
23         else
24         {
25             local_output+=1;
26             txbyte='A';
27         }
28         break;
29     }
```

Figure 7 Embedded C code from master PIC

The next section of the master code then uses the custom “read_buttons” function in order to read the state of the buttons and store it in a variable so that the program knows what buttons have been pressed and therefore can work out what token to send.

The switch case statement then begins, having a different case dependent on what different buttons were pressed. This enables the program to look at the button state and run the appropriate code based on which button was pressed.

Within each case, the program first checks using a local variable whether the light has already been activated, and from that will set the token variable to the correct token based on the existing light state. This ensures that if the light is toggled on it will toggle it off and vice versa. This code for the switch case then repeats for each of the relevant switch cases within the program.

```
67  
68  
69  
printf("%C",txbyte);
```

Figure 8 Embedded C code from master PIC

The final bit of the main code then simply prints the token stored in the token variable out via RS232. This effectively sends it to the other device to be read, and this can be verified when viewing the virtual terminal on Proteus.

Slave

```
1  #include <main.h>  
2  
3  char local_output=0;  
4  char control_byte=0;  
5  
6  void main()  
7  {  
8      CHAR buts=0;  
9  
10     while(TRUE)  
11     {  
12
```

Figure 9 Embedded C code from master PIC

Similar to the master code, the slave begins with configuration and setup as well although the variable set up is slightly different as the data needs to be stored differently. Once again, the program is set to go into an infinite loop, however this time it is to enable the program to be constantly waiting for a potential input from the other PIC microprocessor.

```
13  
14  
15  
16  
if(kbhit()) //wait for  
{  
    buts=getc();  
    printf("%C",buts);
```

Figure 10 Embedded C code from master PIC

The slave program then begins the main code section by using the “kbhit” function. This is normally used to register when a keyboard press is detected, however in this scenario because the input to

the console is via an RS232 line from the other PIC, it instead detects when the token is entered into the RS232 stream.

Once a token entry is registered, the program then stores the token with a variable before echoing the token into the virtual terminal to confirm receipt of the correct token. This could be used for a variety of functions such as passing it along to a third microprocessor however in this specific scenario it is used simply as a debug tool to ensure the correct data is being transmitted.

```
17         switch(buts)           //begin switch case for light activation
18     {
19
20         case 'A':               local_output+=0x01;
21                                 control_byte=control_byte+=0x01;
22                                 break;
23         case 'a':               local_output=local_output & 0xfe;
24                                 control_byte=control_byte & 0xfe;
25                                 break;
26         case 'B':               local_output+=0x02;
27                                 control_byte=control_byte+=0x02;
28                                 break;
29         case 'b':               local_output=local_output & 0xfd;
30                                 control_byte=control_byte & 0xfd;
31                                 break;
32         case 'C':               local_output+=0x04;
33                                 control_byte=control_byte+=0x04;
34                                 break;
35         case 'c':               local_output=local_output & 0xfb;
36                                 control_byte=control_byte & 0xfb;
37                                 break;
38         case 'D':               local_output+=0x08;
39                                 control_byte=control_byte+=0x08;
40                                 break;
41         case 'd':               local_output=local_output & 0x07;
42                                 control_byte=control_byte+=0x08;
43                                 break;
44     }
```

Figure 11 Embedded C code from master PIC

Once again, the slave program uses a switch case method for determining the correct thing to do based on the token selected. The switch case checks the token and runs the appropriate code based on the token value. As lower-case letters are used to represent deactivating the lights and upper case to activate the light, the program can differentiate between every single potential scenario and respond appropriately. The value to be displayed is then stored in a variable in such a way that it keeps the existing settings of other lights whilst also performing the change to the latest button press.

```
43     }
44     output_b(control_byte);
45 }
```

Figure 12 Embedded C code from master PIC

Last but not least, the program then outputs the value from the light state variable to port B which will activate the corresponding lights (and horn).

Program Flow

The flowcharts below show the overall flow of the programs from an operational perspective. Figure 13 shows the program flow for the master whereas Figure 14 shows the program flow for the slave. These illustrate how the two programs operate and interact with each other.

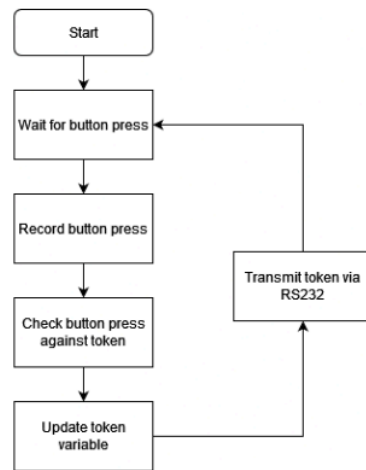


Figure 13 Embedded C code from master PIC

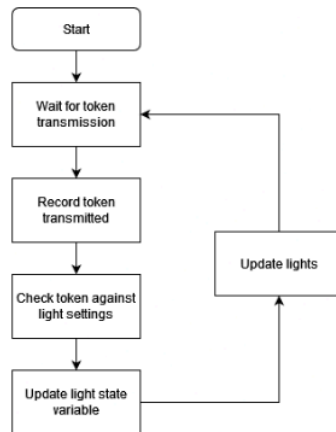


Figure 14 Embedded C code from master PIC

Conclusion

Analysis of Work

In conclusion, the program and schematic created throughout this report has successfully demonstrated a token based distributed network using PIC microprocessors and RS232 communication. Upon running of the code, the system can transmit different tokens from one microprocessor to another to enable the slave device to perform functions at the instruction of the master.

References

There are no sources in the current document.

- [1] “Quality management,” *Theiet.org*, 2024.
<https://education.theiet.org/secondary/teaching-resources/quality-management> (accessed Nov. 13, 2024).
- [2] “NHS England» Quality Management Systems,” *www.england.nhs.uk*.
<https://www.england.nhs.uk/nhsimpact/assessment-and-improvement/quality-management-systems/>
- [3] “Point to Point & Multipoint Radio Systems,” *CSE Crosscom*, Nov. 14, 2023.
<https://csecrosscom.net/solutions/communications/linking-and-backhaul/point-to-point-multipoint-radio-systems/> (accessed Nov. 13, 2024).
- [4] “Point to Point Communication Comprehensive Guide,” *Sanny Telecom*, Apr. 27, 2024. <https://www.sannytelecom.com/point-to-point-communication-comprehensive-guide/>
- [5] “What is Point-to-Multipoint Communication?,” *Sanny Telecom*, May 18, 2024.
<https://www.sannytelecom.com/what-is-point-to-multipoint-communication/>
- [6] L. MacGillivray, “The Benefits of Point to Multipoint Connectivity in Communications,” *www.chicomm.com*. <https://www.chicomm.com/blog/the-benefits-of-point-to-multipoint-connectivity-in-communications>
- [7] D. Edwards, “David Edwards,” *Net Insight*, Oct. 04, 2024.
<https://netinsight.net/securing-media-streams-with-rist/> (accessed Nov. 13, 2024).
- [8] “What is CAN Bus and how does it work?,” *Actisense*, Jul. 27, 2023.
<https://actisense.com/news/what-is-can-bus-and-how-does-it-work/>
- [9] Kittu20, “Multimaster in CAN,” *All About Circuits*, Mar. 24, 2024.
<https://forum.allaboutcircuits.com/threads/multimaster-in-can.199873/> (accessed Nov. 13, 2024).
- [10] “26.4 Multi-Master Bus Systems, Arbitration, and Synchronization,” *Microchip.com*, 2022. <https://onlinedocs.microchip.com/oxy/GUID-0EC909F9-8FB7-46B2-BF4B-05290662B5C3-en-US-12.1.1/GUID-76E847C6-8B91-40D9-86F6-A917307D5BEE.html> (accessed Nov. 13, 2024).
- [11] Wikipedia Contributors, “Multi-master bus,” *Wikipedia*, Dec. 03, 2021.
https://en.wikipedia.org/wiki/Multi-master_bus
- [12] “PIC18F2525/2620/4525/4620 Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology.” Available:
<https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/39626e.pdf>
- [13] “Serial communication protocols. Main types and characteristics.,” *Serial Port Monitor*. <https://www.serial-port-monitor.org/articles/serial-communication/types-of-serial-protocols/>

Molly Scarlet DIS Assignment

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

70/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

GRADING FORM: SMU GENERIC_GJ

MOLLY SCARLET

69.60

RESEARCH-INFORMED LITERATURE



You have explained what point to point and point to multipoint and multimaster are correctly. 65%

13

KNOWLEDGE AND UNDERSTANDING



The design of your circuit is suitable for the task 72%

18

ANALYSIS



You have completed the programs for both devices and the circuit works well, except you have failed to flash the indicator 68%

17

PRACTICAL APPLICATION AND DEP.



Your explanation is good 72%

10.8

SKILLS FOR PROFESSIONAL PRAC.



The report is well presented with sections to explain the code and include a reasonable number of references. 72%

10.8