Practical assignment of Digital Systems and Microprocessors Course 2024-2025

Practice 2

	Login	Name
Students	elia.clavaguera	Èlia Clavaguera Estella
	pau.arasa	Pau Arasa Cerdà

	Board	Report	Grade
Delivery			

Date	**9th of March 2024**

Cover of the report

TABLE OF CONTENTS:

1. Summary of the statement

The objective of this practical assignment is to design, build and program a light control system for a meeting room. When a user enters the room, the system adjusts the lights configuration to that user's settings. If no one is in the room, the lights must be turned off.

This involves communicating the MCU with various external peripherals: A matrix keyboard, 6 LEDs, a 16x2 LCDs, an RFID module and a computer (used as a CLI for the MCU).

Some of these peripherals communicate with the MCU through different protocols, like SPI or USART.

2. System design

Hardware used:

- PIC18F4321 MCU
- RFID module RC522
- Matrix keyboard
- LEDs
- 16x2 LCD

Communication protocols:

- SPI
- USART

MCU internal peripherals:

- Timer0

Software design:

We used the cooperative systems paradigm, which allows us to program the MCU to do various functions simultaneously by fractioning the implementation and executing them part by part. To do this, we needed to store the 'progress' of the functions.

We also used the TADs and engines concepts.

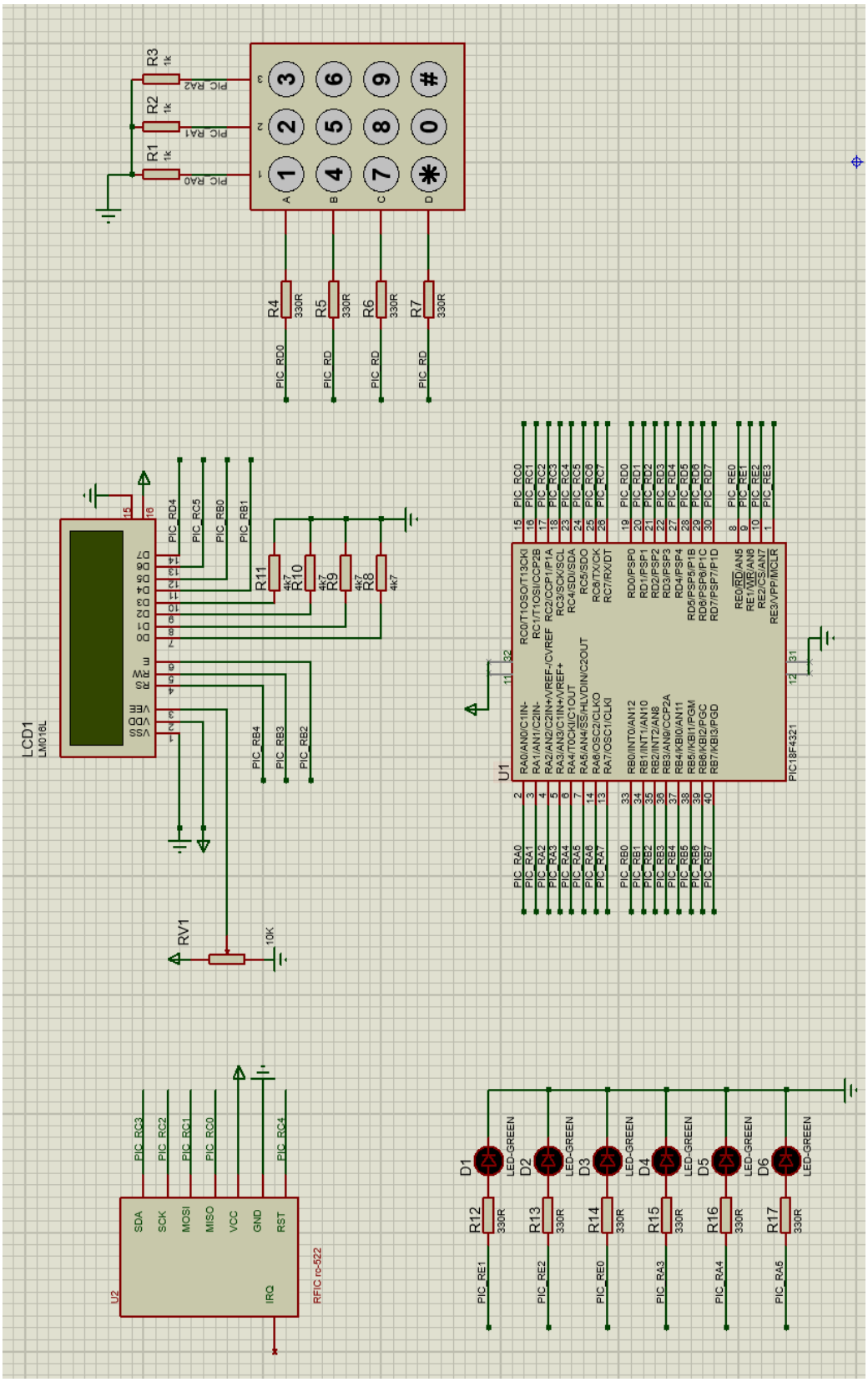
User inputs:

The user can interact with the system through the matrix keyboard and through the command-line interface (PUTTY via EUSART).

System outputs:

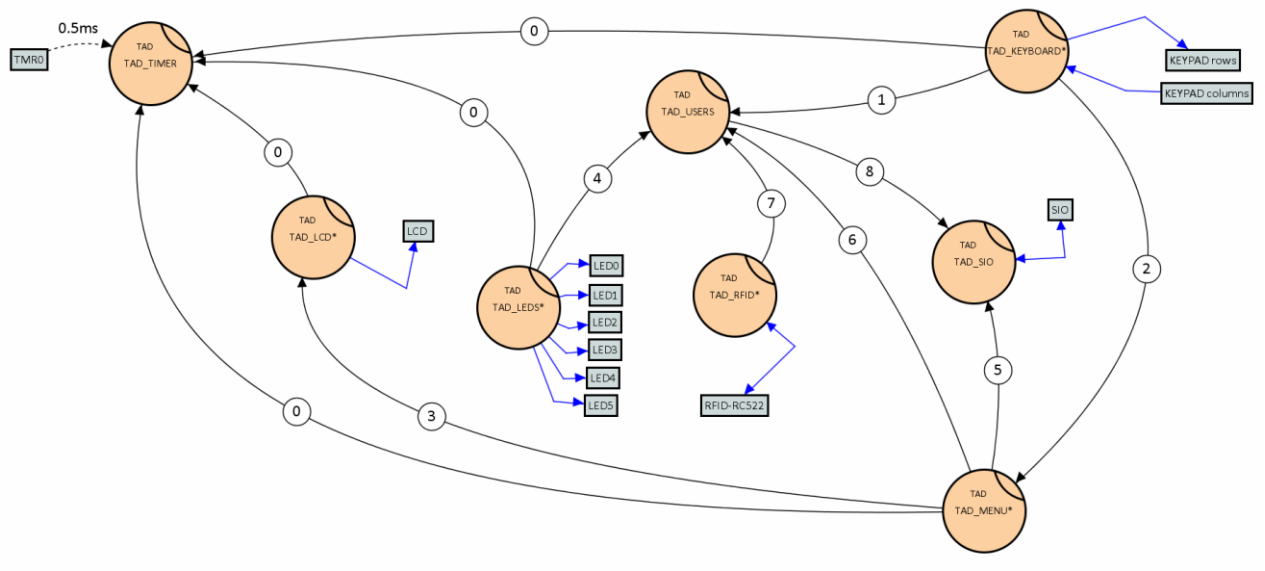
The system displays the outputs through various peripherals: it uses the LDC to display the current state of the room, the LEDs represent the actual lights of the room, and the CLI (EUSART transmission) to display the system's menu and its functions.

3. Electrical schematic



4. Diagram of TADs

//Outline of the different blocks implemented to solve the practice



Dictionary:

0:

unsigned char TI_NewTimer(unsigned char *TimerHandle);
// Post: Returns TI_TRUE when a new virtual timer has been successfully created, it returns TI_FALSE otherwise
// Writes in *TimerHandle the handled of the assigned timer

unsigned int TI_GetTics (unsigned char TimerHandle);
// Pre: TimerHandle comes from Ti_NewTimer.
// Post: Returns the number of tics that have happened since the last TI_ResetTics of TimerHandle.

void TI_ResetTics (unsigned char TimerHandle);
// Pre: TimerHandle comes from Ti_NewTimer.
// Post: Stores the time reference associated to TimerHandle.

1:

void resetUserSettings(void);

//Post: Resets the user light settings to their initial conditions

void setLightPreference(unsigned char value);

//Pre: There is a user in the room (roomHasUser() returns TRUE)

//Post: Will change the setting of one of the lights in the current user to the input

2:

void setMenuState(unsigned char state);

//Pre: The input is a valid state of the Menu motor

//Post: Sets the state of the Menu motor to the input value

3:

unsigned char getMinut_0(void);

//Post: Returns the current low digit of the system's minutes

unsigned char getMinut_1(void);

//Post: Returns the current high digit of the system's minutes

unsigned char getHora_0(void);

//Post: Returns the current low digit of the system's hours

unsigned char getHora_1(void);

//Post: Returns the current high digit of the system's hours

4:

unsigned char getLedConfigFromCurrentUser(unsigned char led, unsigned char call);

//Pre: currentUser is a valid user (not out of bounds). 'led' is a valid input [0..6].

//Post: returns the settings for the desired LED depending on the 'led' input. The format of the value returned depends on the variable 'call'.

5:

unsigned char SIO_RXAvail(void);

//Post: returns TRUE if there's a receive character pending to be read. Otherwise, false

unsigned char SIO_GetChar(void);

//Pre: SIO_RXAvail() has returned true

//Post: Performs a destructive read of the received character

unsigned char SIO_TXAvail(void);

//Post: returns TRUE if there's availability to send a character. Otherwise, false

void SIO_PutChar(unsigned char value);

//Pre: SIO_TXAvail has returned true

//Post: Queues a new character for transmission

unsigned char SIO_PutString(char* s);

//Pre: Must be called repeatedly until it has finished printing

//Post: Checks if sending is possible, and if so sends a character of the input string. In the next call it automatically sends the next character. Returns 1 if it has finished, else 0.

6:

unsigned char getCurrentUser(void);

//Post: Returns the id of the user in the room, or N_USERS if the room is empty

unsigned char printUserId(unsigned char user);

//Post: manages the printing of the ID of a user through EUSART

void setStateUsers(unsigned char state);

//Pre: the input is a valid state of the state machine of Users engine

//Post: sets the state of the Users engine to the input

7:

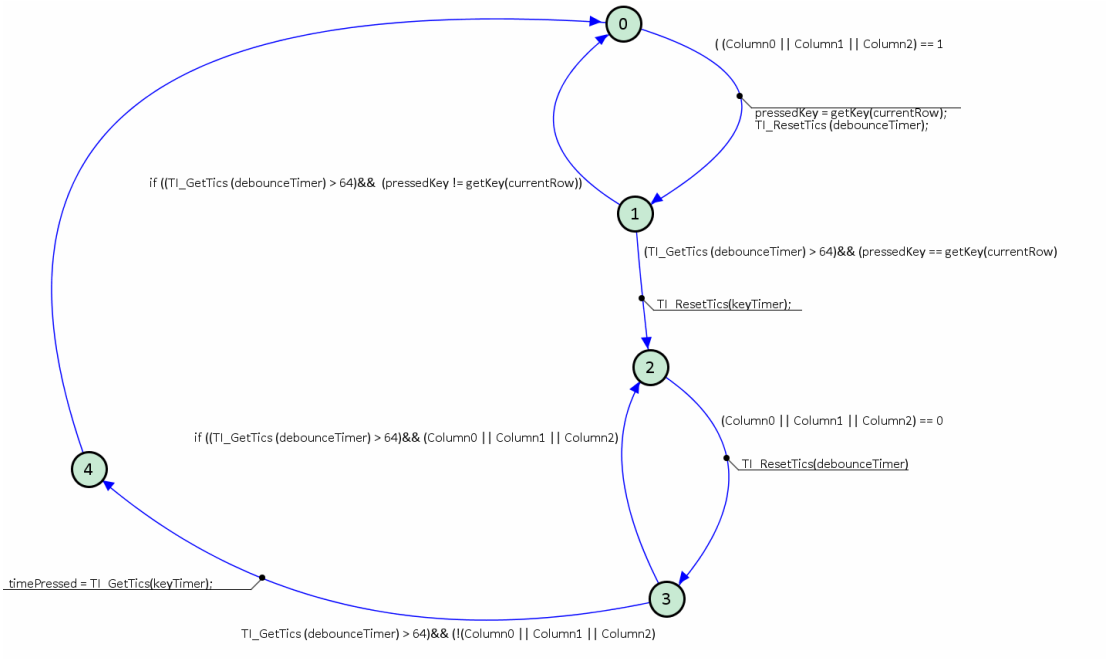
unsigned char receivedID(unsigned char ID[]);

//Post: If the ID input is valid, returns 1. Else returns 0.

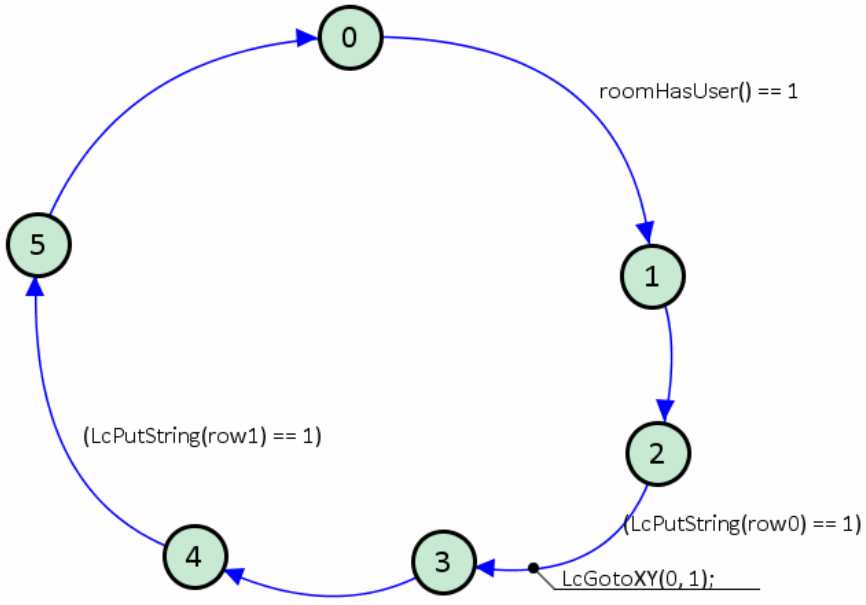
Updates the matchedUser variable to the input.

5. Engine diagrams

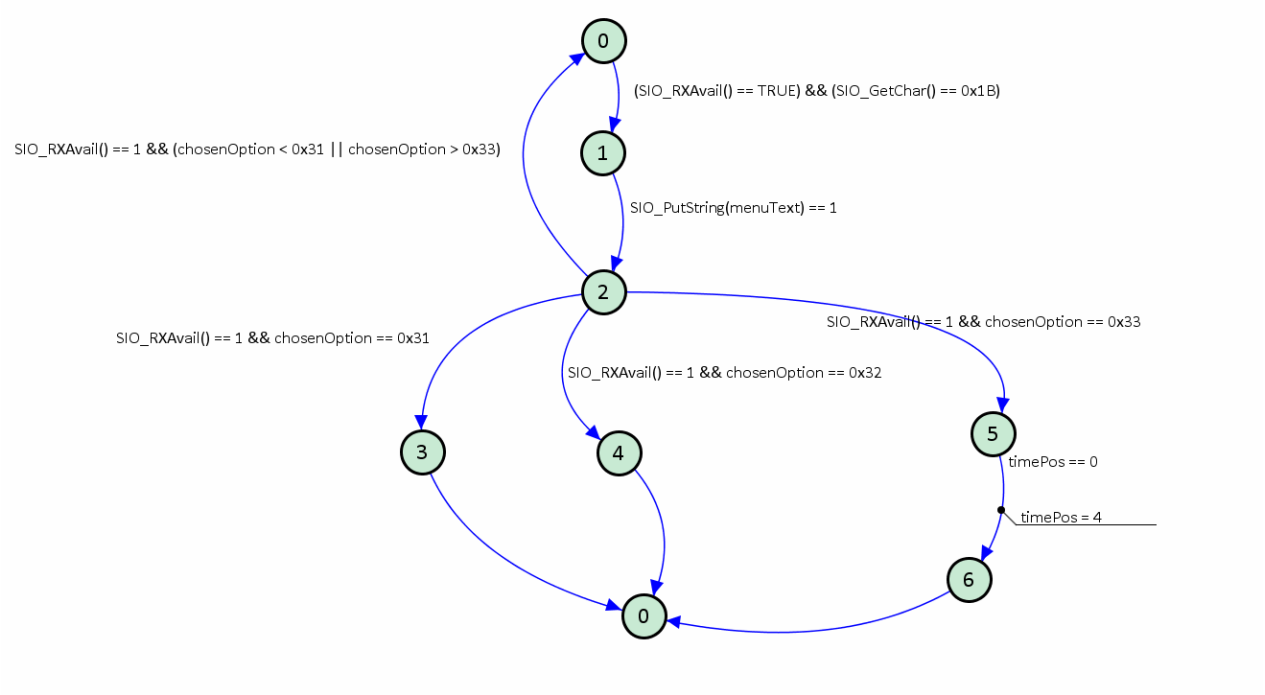
5.1 Keypad engine



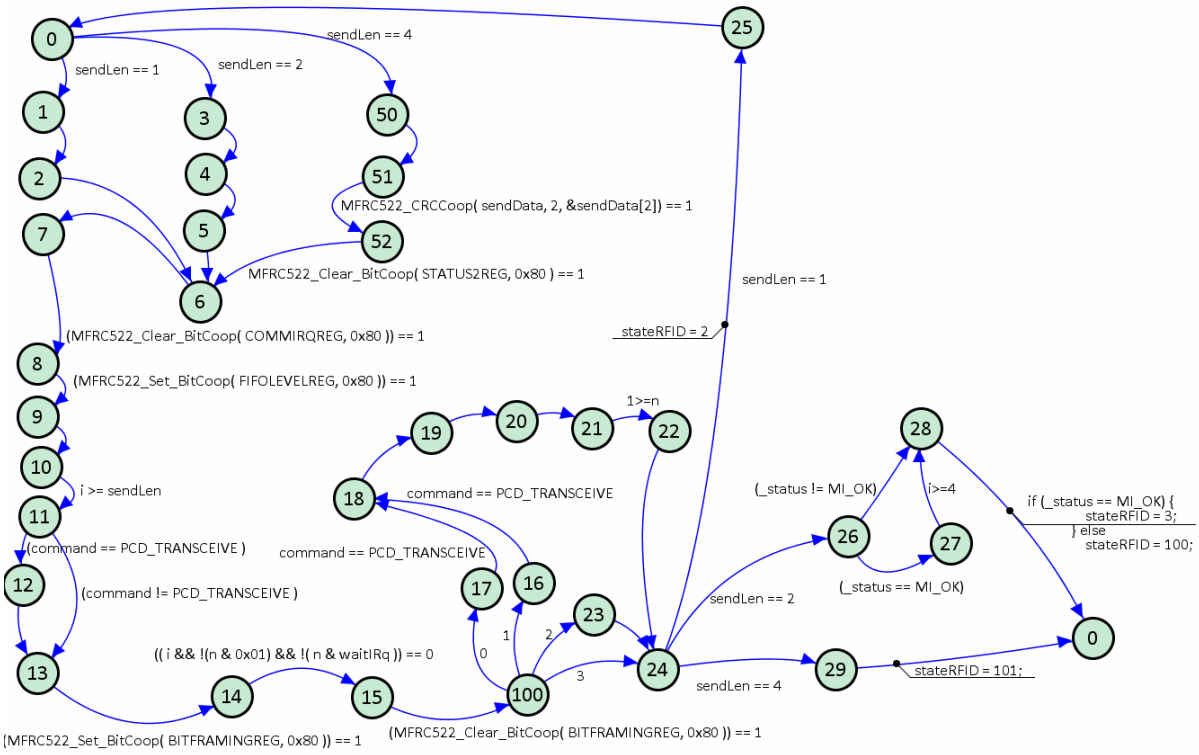
5.2 LCD engine



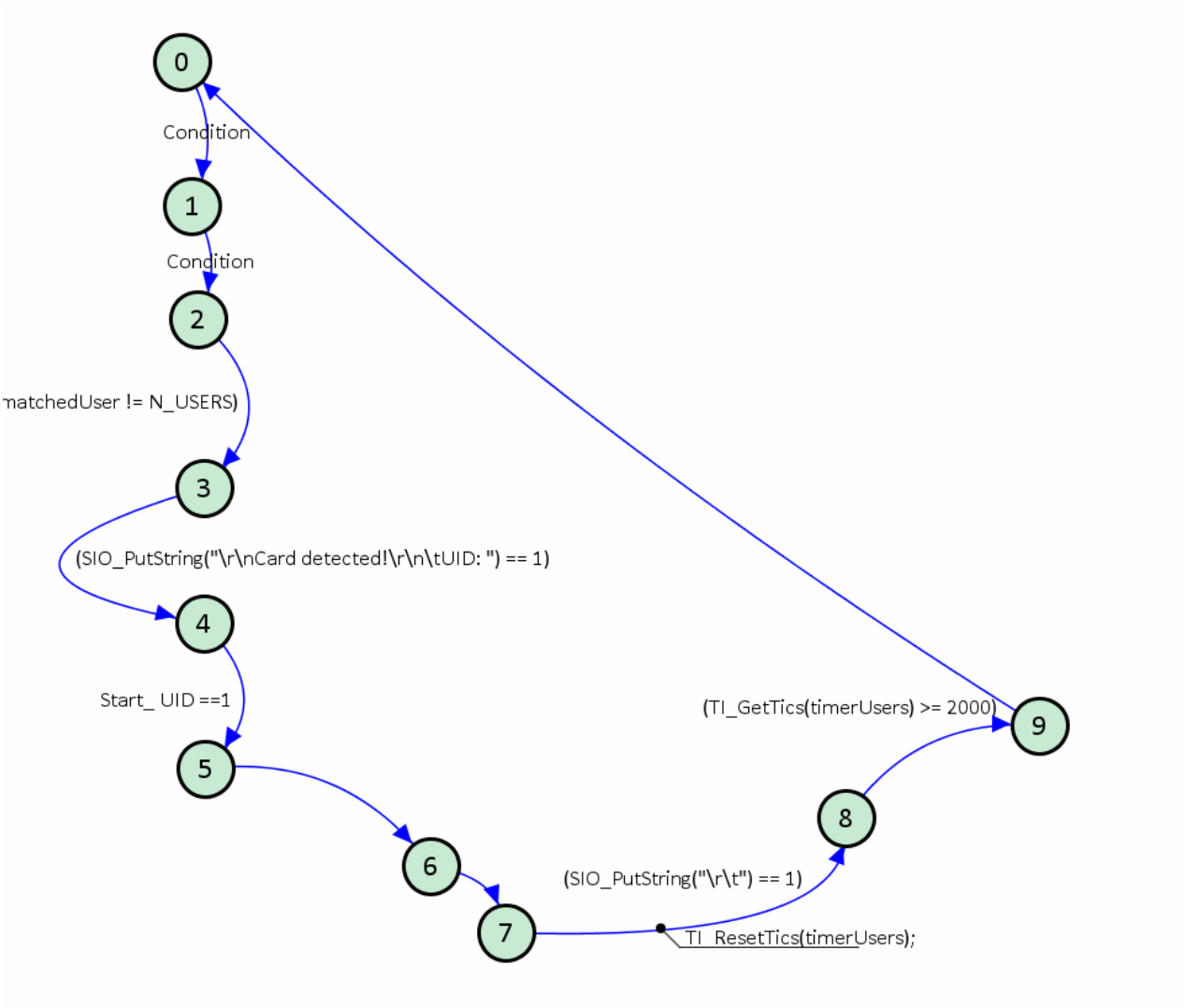
5.3 Menu engine



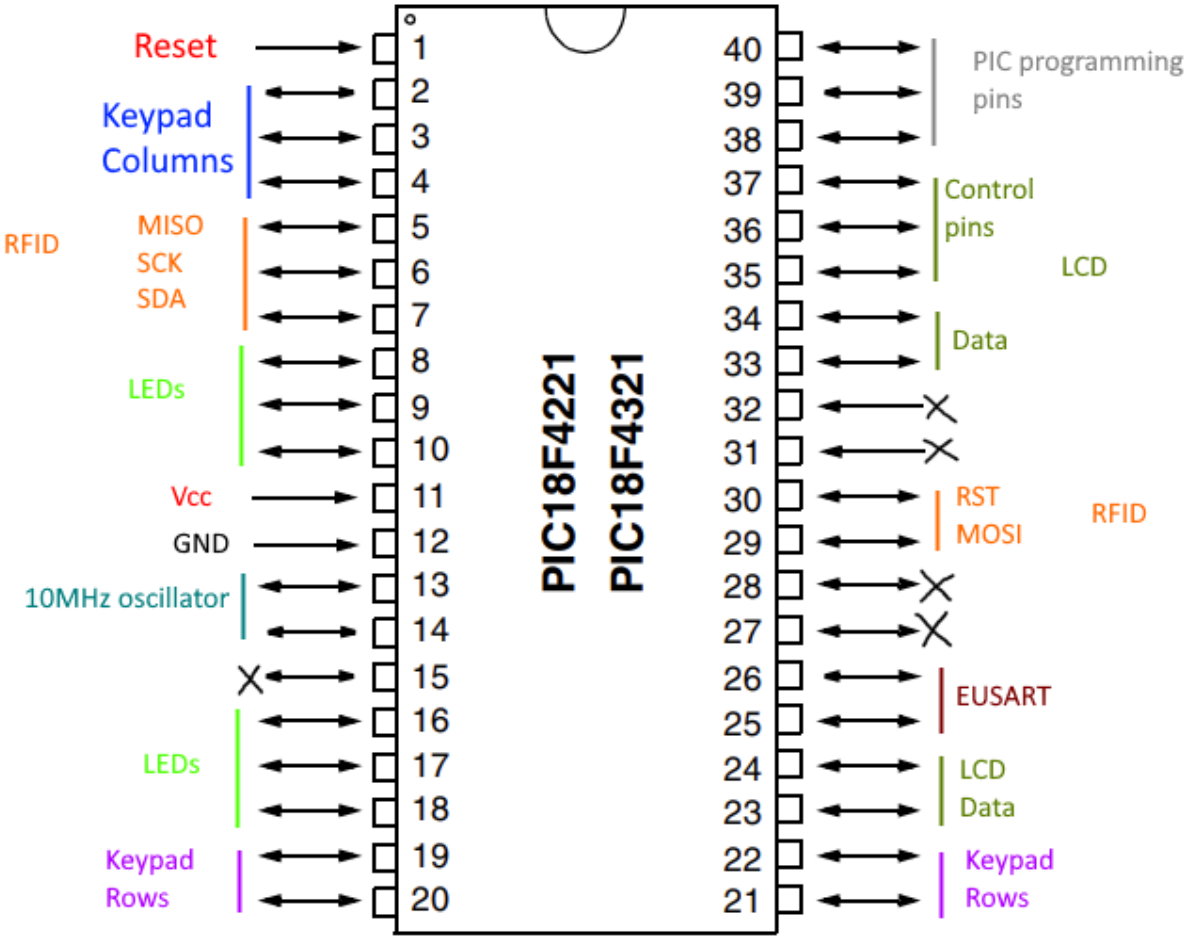
5.4 RFID engine



5.5 Users engine



6. Microcontroller configurations



7. Observed problems

We had many issues during the process of solving this board, and we considered that many of them were out of our control. For example, our PIC wasn't receiving the inputs from the keypad, we just changed the pin assignation and it started working with the exact same code, and something similar happened with the RFID module. These were the biggest time wastes in the entire project.

We also had some issues with memory but it was easy to solve.

Also, our PIC got burned and we took a while to realise that that was the issue.

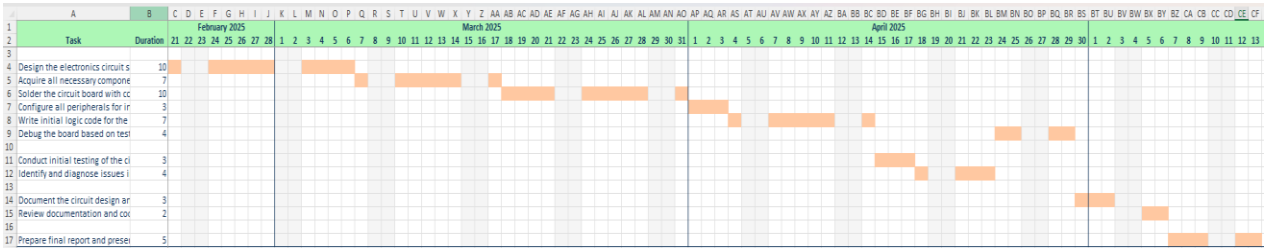
Finally, we had some issues with the PicKIT programmer, until we realised that it had something to do with the USB-TTL (when it was connected it wouldn't program).

8. Conclusions

This practical assignment was a great way to gain a deeper understanding of many functions of the microcontroller.

Firstly, we learned to design the software with TADs and engines. Secondly, we learned about cooperative systems, how to design them and keeping track of the cooperative uncertainty, as well as understanding its' importance. We also used peripherals that we had never used (16x2 LCD and RFID reader), which means we also had to learn different communication protocols than EUSART. Overall, we gained some hands-on experience on embedded systems using the concepts we learned in class, but we had to go much further due to the assignment being more complex than the problems in class.

9. Planning



Task	Duration
Design the electronics circuit s	10
Acquire all necessary compone	7
Solder the circuit board with cc	10
Configure all peripherals for ir	3
Write initial logic code for the	7
Debug the board based on test	4
Conduct initial testing of the ci	3
Identify and diagnose issues i	4
Document the circuit design ar	3
Review documentation and coc	2
Prepare final report and presei	5