Practical assignment of Digital Systems and Microprocessors Course 2024-2025

Practice 2

Phase B

LSServerControl

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

	Board	Report	Grade
Delivery			

Date	**8th of July 2025**

Cover of the report

TABLE OF CONTENTS:

1. Summary of the statement

This project utilizes our PIC18F4321 microcontroller to watch over a room's temperature and keep the user updated about said temperature using some peripherals (an RBG led and two fans). Our main goal is log every temperature reading the system executes, and letting users tweak the system settings through a simple Java graphical panel.

Our components, aside from our microcontroller include two fans driven by a PWM signals, a TMP36 sensor, a RGB led, a 62256 RAM chip that collects long-term logs, which is controlled by 74LS chips (4 counters) and a DS3231 RTC that stamps can both receive and output time stamps, as well as count in real time with the input date it receives. When the system detects a critical temperature, the information on both the temperature data and the time and date of this occurrence need to be saved in our internal EEPROM memory in the microcontroller, which needs to be able to store up to 15 logs.

When the unit first powers up, the system will remain idle until the user inputs the time, date, our temperature thresholds to detect what state the temperature is in (low, moderate, high or critical), and finally the seconds between each temperature reading.

Each temperature state has actions associated to it, for example, when the most temperature reading corresponds in the low threshold, our LED will turn green, and one single fan will start operating in moderate intensity. With the moderate threshold, the LED turns blue, and both fans turn on. When the temperature is detected as high, the LED turns red and both fans start operating at maximum power.

Finally, in the critical range, the LED starts alternating between red and magenta, while both fans turn off. Each critical reading will be timestamp and stored in our internal EEPROM, which needs to persist without voltage.

2. System design

Hardware used:

- PIC18F4321 → microcontroller
- DS3231 → RTC
- RGB LED
- 2 x fans
- 4 x 74LS161 → Counters for the RAM address
- 62256 → RAM
- TMP36 → Temperature sensor

Communication protocols:

- SIO
- I2C

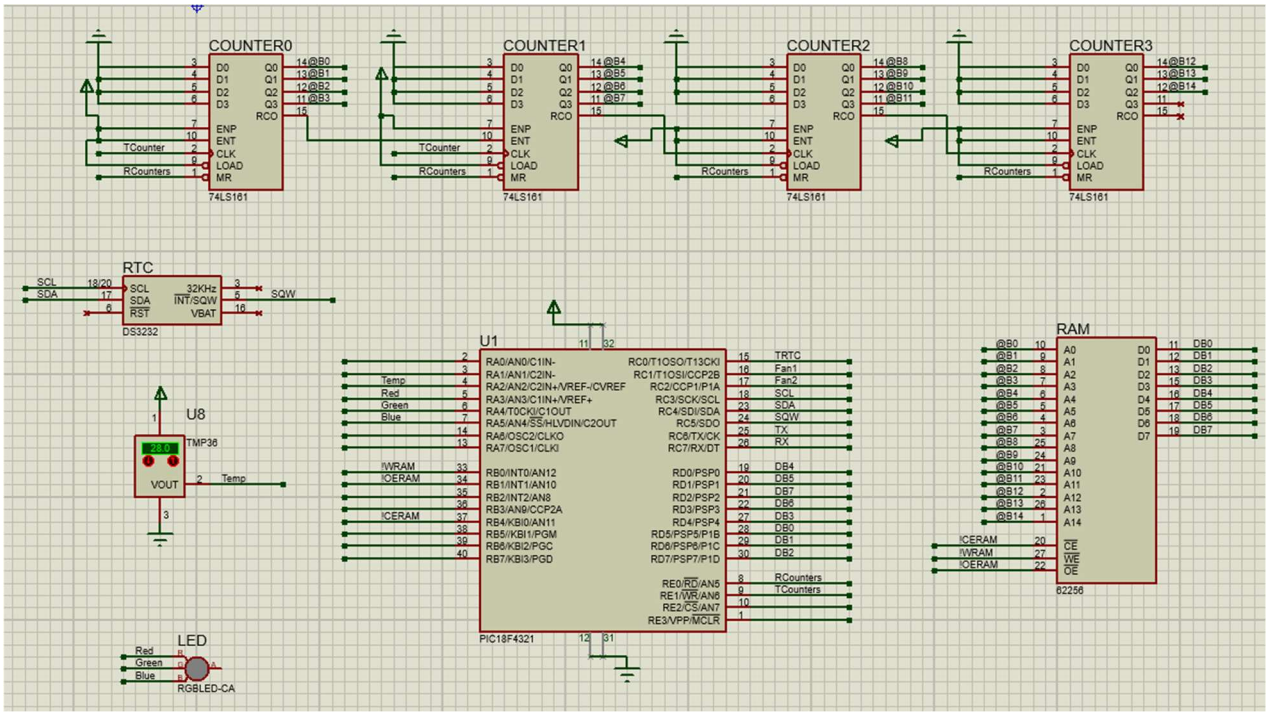
User Inputs:

- All inputs are received by the Java interface using our internal SIO of the microcontroller

System outputs:

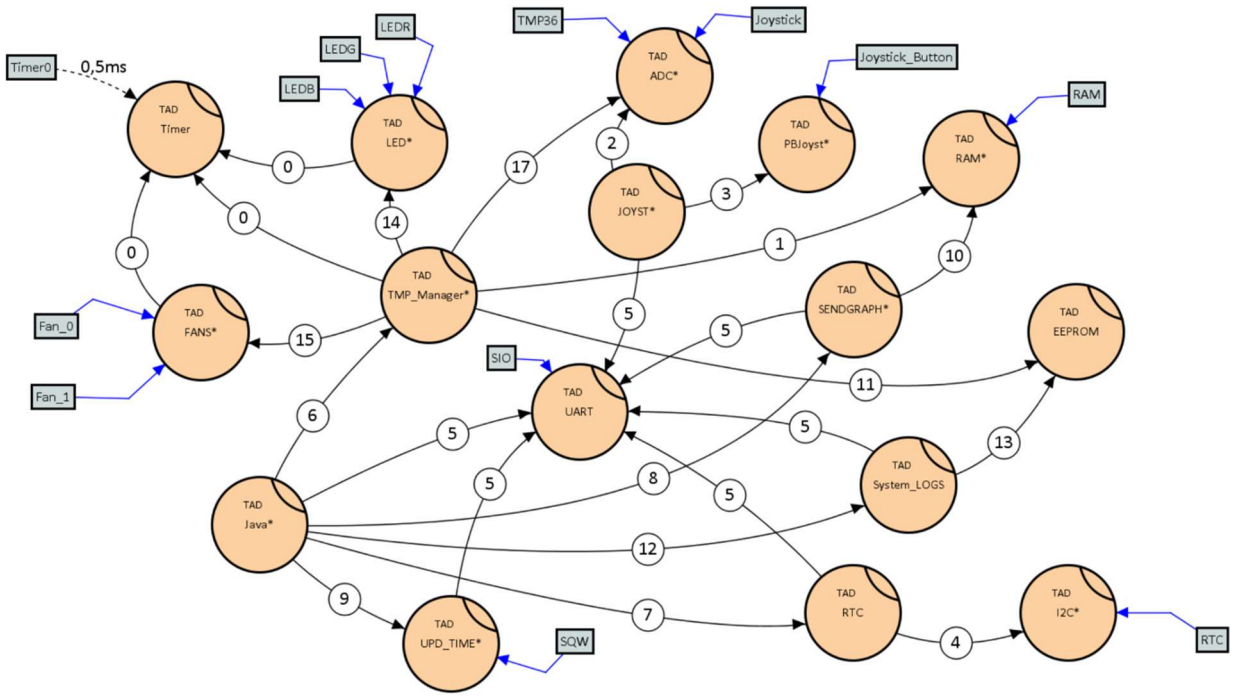
- SIO: The user can request data, and it will be shown in the Java interface
- LED and Fans: Will reflect the temperature state

3. Electrical schematic



//TODO POSAR CAPACITORS I TAL AMB EL GOODNOTES

4. Diagram of TADs



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

void TI_ResetTics (unsigned char TimerHandle);

// Pre: TimerHandle comes from Ti_NewTimer.

// Post: Stores the time reference associated to TimerHandle.

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.

1:

char getRamIsBusy(void);

//Post: returns 1 if the RAM is busy (already performing an operation), 0 if it's available

void writeToRam(unsigned char);
//Pre: RAM address is the one where data will be written (new data will overwrite old data).
// RAM is not busy
//Post: Writes the input data in the current @ of the RAM.

2:
unsigned char getDirectionJoystick(void);
//Post: Returns current direction of the joystick, varying from UP, DOWN, LEFT, RIGHT or CENTER

3:
unsigned char getButtonIsPressed(void);
//Post: Returns 1 if button is pressed, 0 otherwise

4:
char I2C_Write(unsigned char data);
//Post: Sends information using i2C to the DS3231 chip (RTC)

char I2C_Read(char flag, char* done);
//Post: Reads data sent from I2C communication, and returns byte

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: SIO_TXAvail has returned true
//POST: Transmits a whole string of characters, returns 1 when it's done, 0 otherwise

6:

void setThresholdLow(unsigned char temp[]);
//Post: Recieves a number, and sets it as the threshold for low temperature

void setThresholdMod(unsigned char temp[]);
//Post: Recieves a number, and sets it as the threshold for moderate temperature

void setThresholdHigh(unsigned char temp[]);
//Post: Recieves a number, and sets it as the threshold for high temperature

void setThresholdCrit(unsigned char temp[]);
//Post: Recieves a number, and sets it as the threshold for critical temperature

void setPollingRate(unsigned char time[]);
//Post: Recieves a number, and sets it as polling rate in seconds in which the temperature will be read, analysed and stored

7:

void setYearToRTC(unsigned char data[]);
//Pre: RTCisAvail() has returned 1
//Post: Sets year to be sent to RTC

void setMonthToRTC(unsigned char data[]);
//Pre: RTCisAvail() has returned 1
//Post: Sets month to be sent to RTC

void setDayToRTC(unsigned char data[]);
//Pre: RTCisAvail() has returned 1
//Post: Sets day to be sent to RTC

void setHourToRTC(unsigned char data[]);
//Pre: RTCisAvail() has returned 1
//Post: Sets hour to be sent to RTC

void setMinToRTC(unsigned char data[]);
//Pre: RTCisAvail() has returned 1
//Post: Sets minute to be sent to RTC

unsigned char RTCisAvail();
//Post: Returns 1 if RTC is available for use, 0 otherwise

unsigned char WriteInitialRTCinfo();
//Pre: RTCisAvail() has returned 1
//Post: Sends saved date to the RTC (if no date has been set, 0/0/0 00:00 is sent)

8:
void startSendingGraph(void);
//Post: Starts process to read and send all the stored information in the RAM, so all the temperature readings

9:
unsigned char startAlarm();
//Post: Starts reading the alarms for every minute

10:
unsigned int getRamAddress(void);
//Post: returns the current address of the ram

unsigned char getWRRound(void);
//Post: returns if the round had started yet, if it's the first or if it's at least the second

char getRamIsBusy(void);
//Post: returns 1 if the RAM is busy (already performing an operation), 0 if it's available

void writeToRam(unsigned char);
//Pre: RAM address is the one where data will be written (new data will overwrite old data).
// RAM is not busy
//Post: Writes the input data in the current @ of the RAM.

11:
unsigned char isEEPROMAvail(void);
//Post: returns 1 if EEPROM can be used, 0 otherwise.

unsigned char writePhraseEEPROM(unsigned char data[], unsigned char size);
//Pre: isEEPROMAvail() returned TRUE
//Post: It writes data in the following empty addresses, until it finished, returning a 1, 0 otherwise.

12:

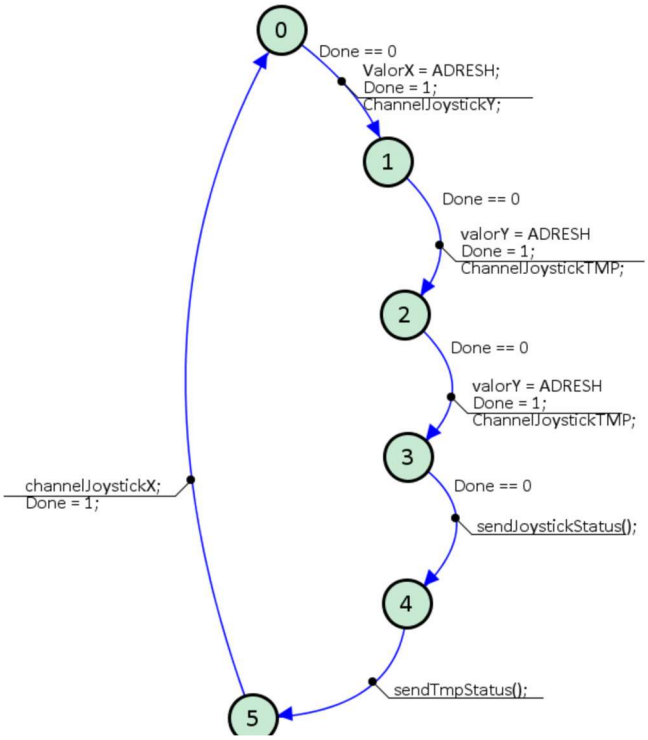
13:
unsigned char isEEPROMAvail(void);
//Post: returns 1 if EEPROM can be used, 0 otherwise.

14:
void recieve_temp(unsigned char temp);
//Post: Recieves new temperature reading, and sets the state of the room, changing colour of the light

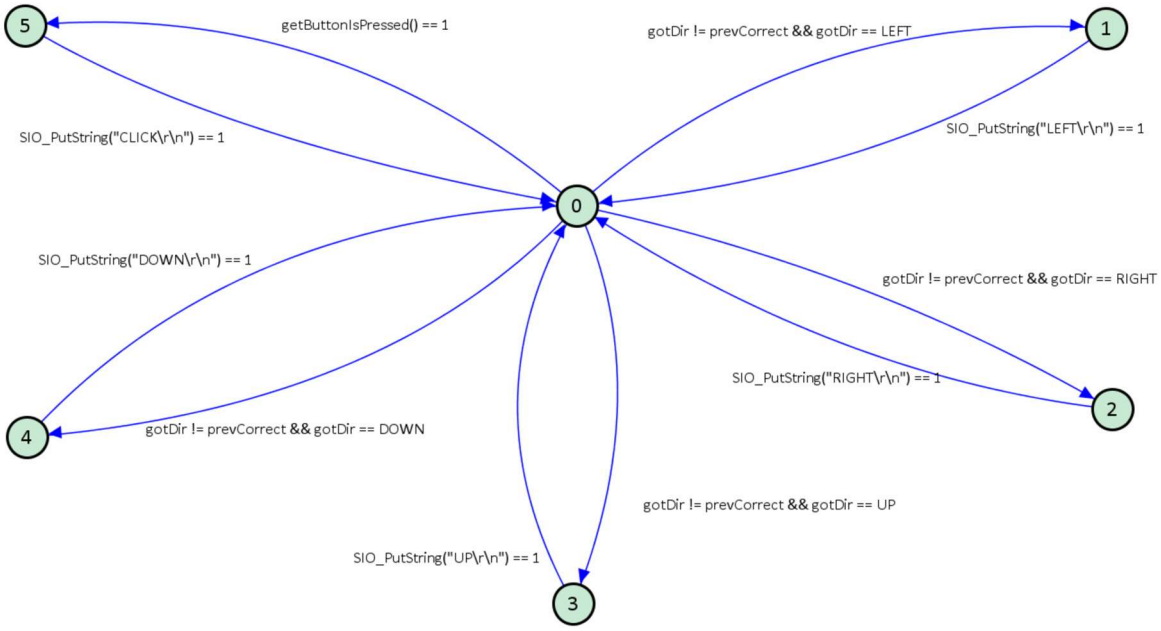
15:
void setFansState(char newState);
//Post: Sets temperature state for fan motor, activating corresponding action to temperature

5. Engine diagrams

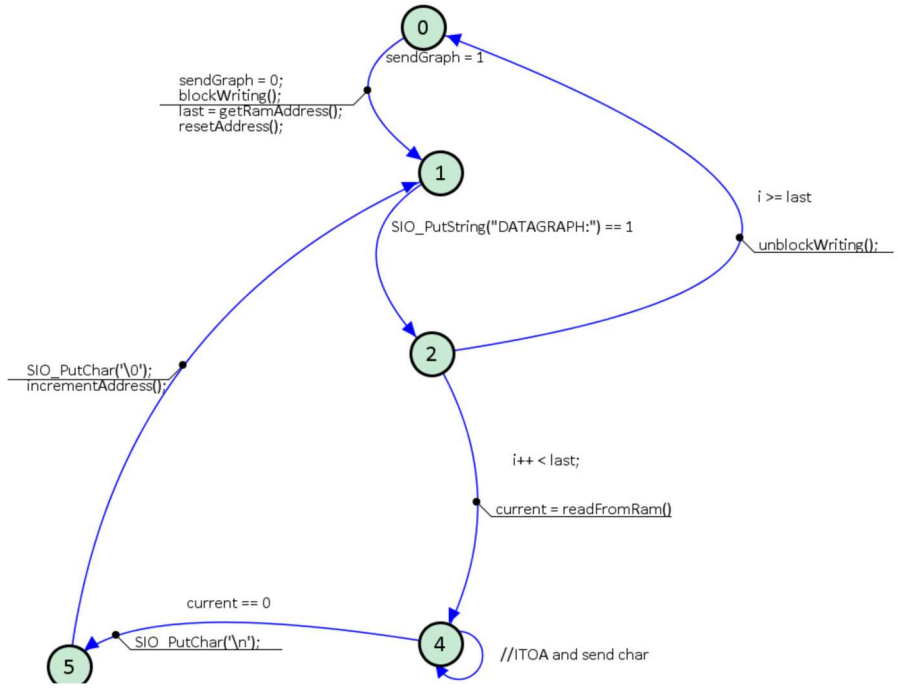
ADC



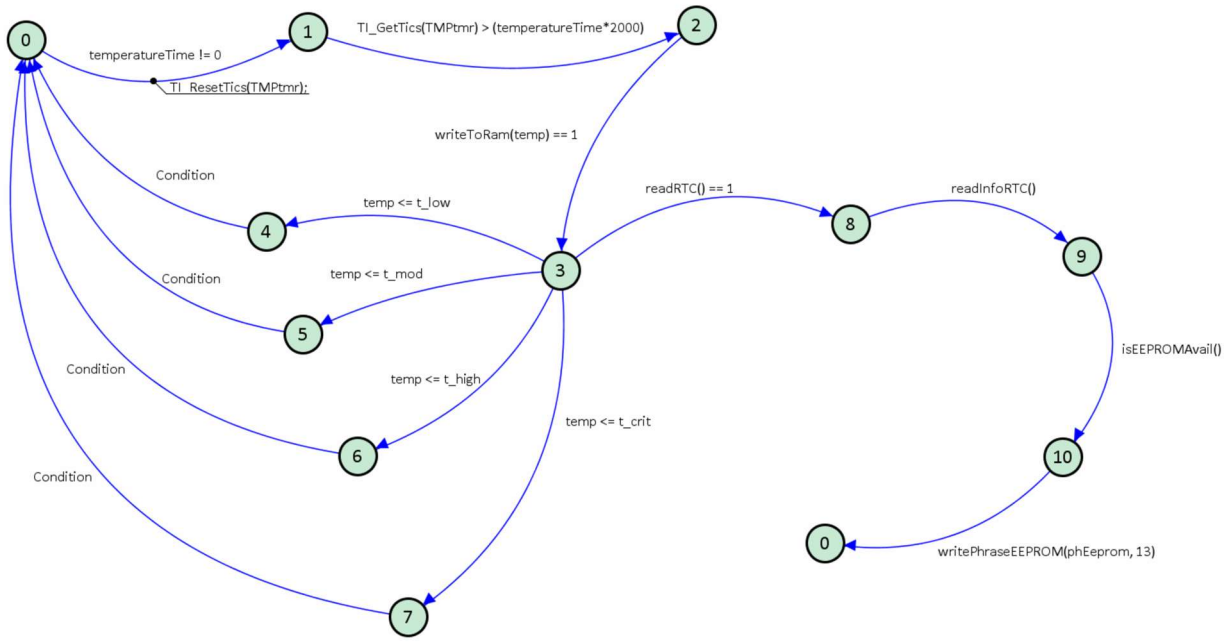
JOYSTICK



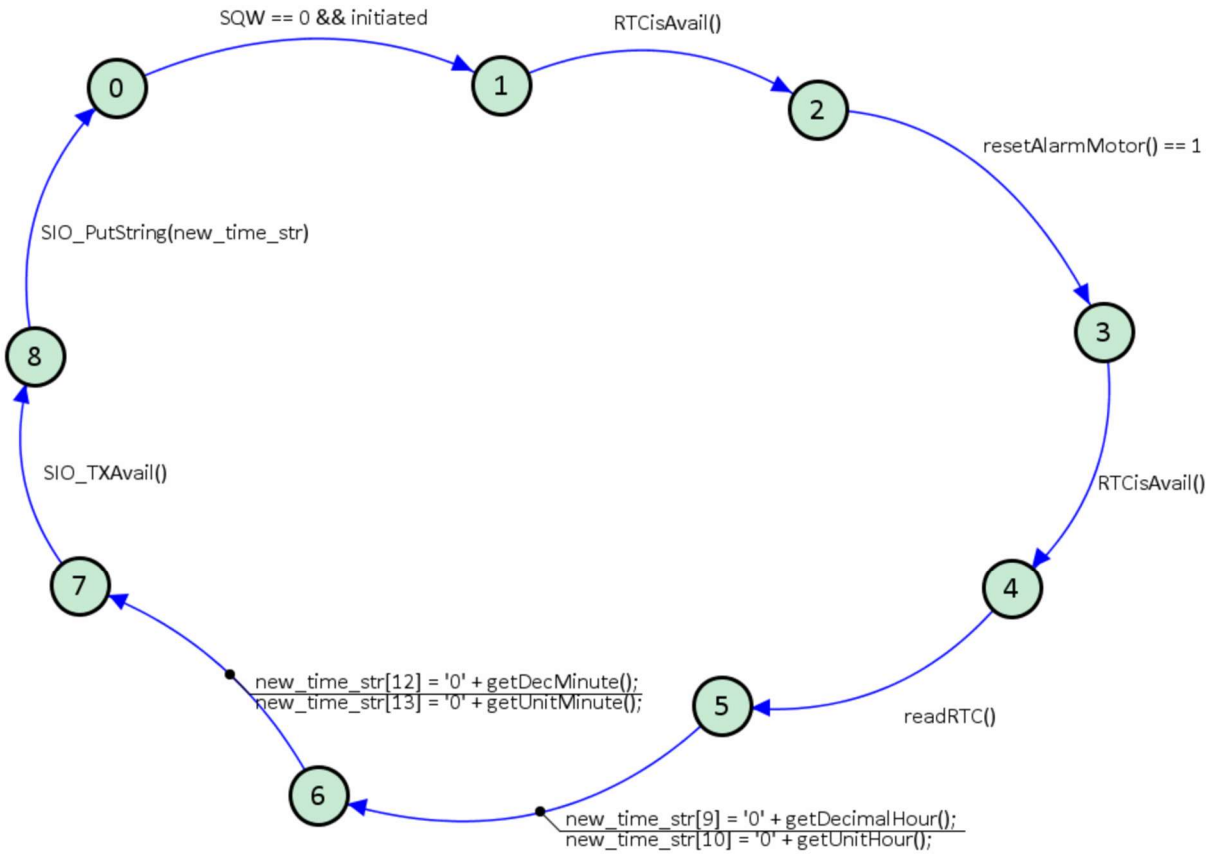
SEND GRAPH



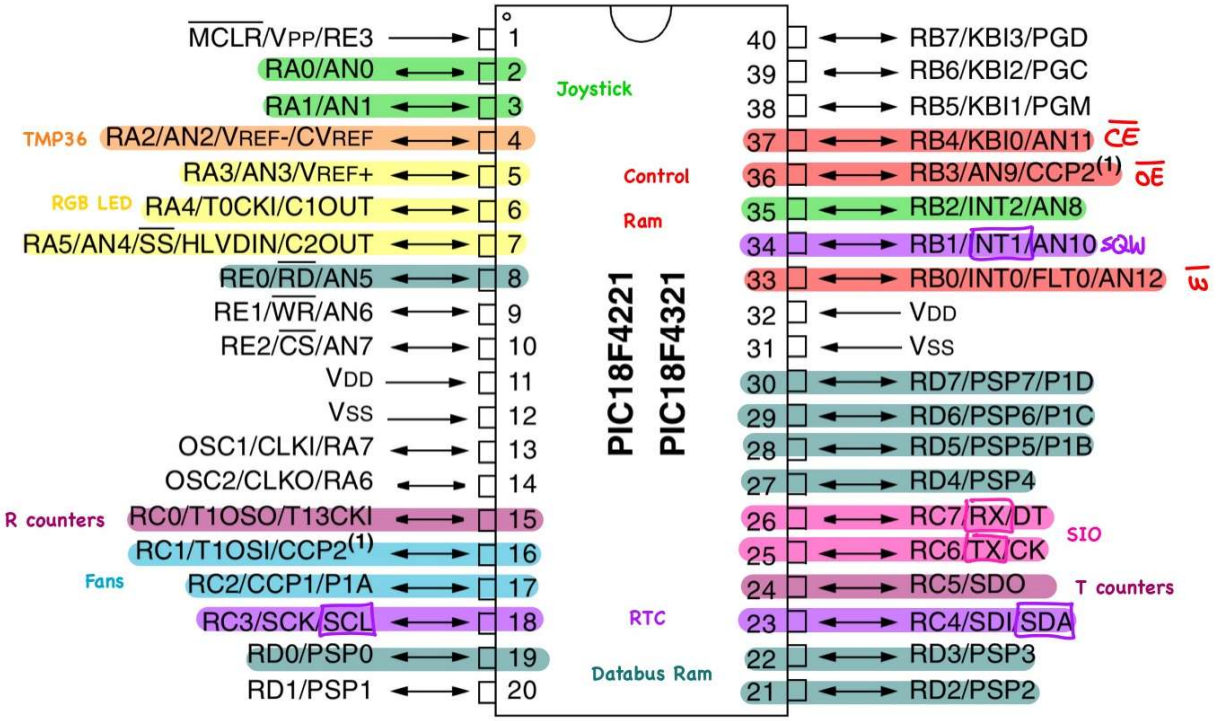
TEMP MANAGER



TAD UPDATE TIME



6. Microcontroller configurations



7. Observed problems

We encountered some issues with our RAM databus, because some times one bit would be changed when sent to the PIC, but then it would go back to the correct bit afterwards. It didn't look like the problem was the RAM, since the information was correct after a couple of reads, so we decided to change the pin from RD1 to RE0, and that fixed the problem.

Another challenged we encountered, was that once we soldiered our fans, the chip TEMP36 would give us weird readings when both fans where off, such as really low readings compared to any other fan state. We fixed this by powering the fans with our USB-TTL 5V pins, which fixed the problem.

Another issue we encountered related to both previous issues, was that when we touched the TEMP36 chip, we would have issues reading the RAM afterwards, due to our contact with the legs of the pin.

8. Conclusions

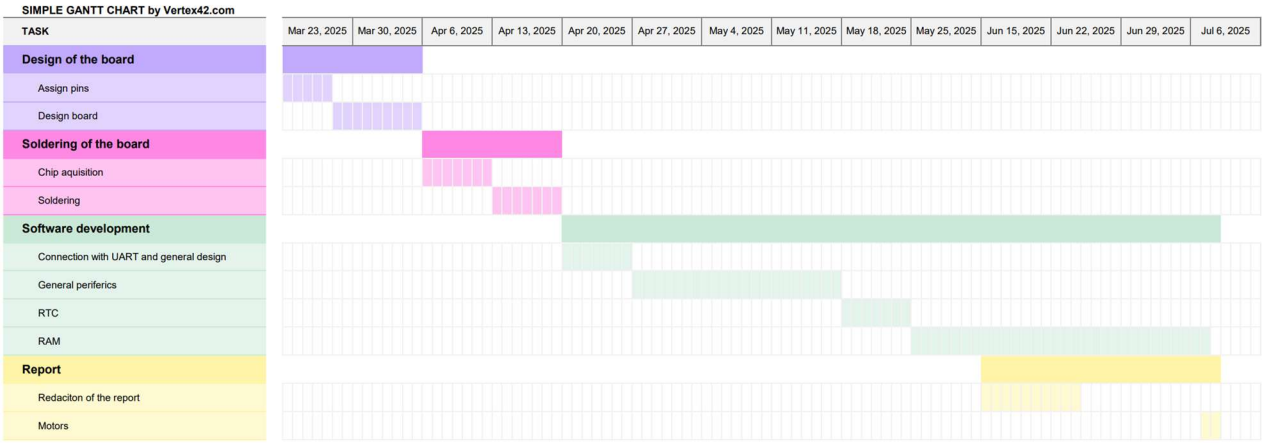
This assignment helped us put to practice concepts seen in class, such as the EEPROM, EUSART or our ADC convertor. We also got to learn how to work with an external RAM, and apply the I2C module of the PIC as master, and our RTC as the slave device.

Developing a code that wouldn't take too much space was also a challenge due to the amount of features this practice had. The part that took the most amount of code was the java interface, since we had to check for the whole string if they matched with the command, and we didn't dynamize it enough, so we ended up creating a function to check each different command, which took a lot of code space.

We could've also organised the code a bit better, since we ended up having a lot of TAD's, which had different functionalities but could've been combined together.

In conclusion, we learned a lot from this experience, as we can see by the fact that just at the moment of finishing we are able to see the improvements that could've been made along the way.

9. Planning


SIMPLE GANTT CHART by Vertex42.com
TASK
Mar 23, 2025
Mar 30, 2025
Apr 6, 2025
Apr 13, 2025
Apr 20, 2025
Apr 27, 2025
May 4, 2025
May 11, 2025
May 18, 2025
May 25, 2025
Jun 15, 2025
Jun 22, 2025
Jun 29, 2025
Jul 6, 2025
Design of the board
Assign pins
Design board
Soldering of the board
Chip aquisition
Soldering
Software development
Connection with UART and general design
General periferics
RTC
RAM
Report
Redaciton of the report
Motors