




TP04 – ELIMINANDO BLOQUEOS

Subsection_5-4-2: Código bloqueante

Setear Fecha y Hora - Bloqueante

```
static void pcSerialComCommandUpdate( char receivedChar )
{
    switch (receivedChar) {
        case '1': commandShowCurrentAlarmState(); break;
        case '2': commandShowCurrentGasDetectorState(); break;
        case '3': commandShowCurrentOverTemperatureDetectorState(); break;
        case '4': commandEnterCodeSequence(); break;
        case '5': commandEnterNewCode(); break;
        case 'c': case 'C': commandShowCurrentTemperatureInCelsius(); break;
        case 'f': case 'F': commandShowCurrentTemperatureInFahrenheit(); break;
        case 's': case 'S': commandSetDateAndTime(); break;
        case 't': case 'T': commandShowDateAndTime(); break;
        case 'e': case 'E': commandShowStoredEvents(); break;
        default: availableCommands(); break;
    }
}
```



Seteo de fecha y hora

Al seleccionar la opción 's' de actualizar la fecha y la hora, se realiza un llamado a la función *commandSetDateAndTime()*.

commandSetDateAndTime()

```
static void commandSetDateAndTime()
{
    char year[5] = "";
    char month[3] = "";
    char day[3] = "";
    char hour[3] = "";
    char minute[3] = "";
    char second[3] = "";

    pcSerialComStringWrite("\r\nType four digits for the current year (YYYY): ");
    pcSerialComStringRead( year, 4);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Type two digits for the current month (01-12): ");
    pcSerialComStringRead( month, 2);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Type two digits for the current day (01-31): ");
    pcSerialComStringRead( day, 2);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Type two digits for the current hour (00-23): ");
    pcSerialComStringRead( hour, 2);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Type two digits for the current minutes (00-59): ");
    pcSerialComStringRead( minute, 2);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Type two digits for the current seconds (00-59): ");
    pcSerialComStringRead( second, 2);
    pcSerialComStringWrite("\r\n");

    pcSerialComStringWrite("Date and time has been set\r\n");

    dateAndTimeWrite( atoi(year), atoi(month), atoi(day),
        |   atoi(hour), atoi(minute), atoi(second) );
}
```

Esta función es un bloque de código bloqueante.

Una vez dentro de la función, el programa se queda 100% avocado a que el usuario actualice la fecha y la hora. Durante este proceso no se verifica el estado de los sensores ni suena la alarma. Una vez elegida la opción de actualizar la fecha y la hora, hasta que no se terminan de ingresar los datos, la alarma no suena independientemente del estado de los sensores.

Actualizar contraseña

```
static void pcSerialComSaveNewCodeUpdate( char receivedChar )
{
    static char newCodeSequence[CODE_NUMBER_OF_KEYS];

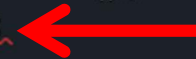
    newCodeSequence[numberOfCodeChars] = receivedChar;
    pcSerialComStringWrite( "*" );
    numberOfCodeChars++;
    if ( numberOfCodeChars >= CODE_NUMBER_OF_KEYS ) {
        pcSerialComMode = PC_SERIAL_COMMANDS;
        numberOfCodeChars = 0;
        codeWrite( newCodeSequence );
        pcSerialComStringWrite( "\r\nNew code configured\r\n\r\n" );
    }
}
```

En contraste, al ingresar una nueva contraseña la misma se actualiza un carácter por vez, y solo cuando se recibe un carácter. No se queda esperando input del usuario. Recién una vez ingresados los cuatro caracteres de la nueva contraseña se actualiza la contraseña llamando a la función *codeWrite()*.

Esta implementación NO es bloqueante.

Propuesta no bloqueante

```
static void pcSerialComCommandUpdate( char receivedChar )
{
    switch (receivedChar) {
        case '1': commandShowCurrentAlarmState(); break;
        case '2': commandShowCurrentGasDetectorState(); break;
        case '3': commandShowCurrentOverTemperatureDetectorState(); break;
        case '4': commandEnterCodeSequence(); break;
        case '5': commandEnterNewCode(); break;
        case 'c': case 'C': commandShowCurrentTemperatureInCelsius(); break;
        case 'f': case 'F': commandShowCurrentTemperatureInFahrenheit(); break;
        case 's': case 'S': commandEnterNewDateAndTime(); break;
        case 't': case 'T': commandShowDateAndTime(); break;
        case 'e': case 'E': commandShowStoredEvents(); break;
        default: availableCommands(); break;
    }
}
```



1. Cuando se selecciona la opción de actualizar la fecha y la hora, se llama a la función *commandEnterNewDateAndTime()*

Propuesta no bloqueante

```
static void commandEnterNewCode()
{
    pcSerialComStringWrite( "Please enter the new four digits numeric code " );
    pcSerialComStringWrite( "to deactivate the alarm: " );
    numberOfCodeChars = 0;
    pcSerialComMode = PC_SERIAL_SAVE_NEW_CODE;
}

static void commandEnterNewDateAndTime() ←
{
    pcSerialComStringWrite("\r\nType four digits for the current year (YYYY): ");
    numberOfDateAndTimeChar = 0;
    pcSerialComMode = PC_SERIAL_SET_DATE_AND_TIME;
}
```

2. Dicha función simplemente actualiza el Estado del SerialCom para reflejar que se está actualizando la fecha y hora. Dispara el prompt al usuario y inicializa numberOfDateAndTimeChar indicando que se debe ingresar el primer carácter.

Propuesta no bloqueante

```
void pcSerialComUpdate()
{
    char receivedChar = pcSerialComCharRead();
    if( receivedChar != '\0' ) {
        switch ( pcSerialComMode ) {
            case PC_SERIAL_COMMANDS:
                pcSerialComCommandUpdate( receivedChar );
                break;

            case PC_SERIAL_GET_CODE:
                pcSerialComGetCodeUpdate( receivedChar );
                break;

            case PC_SERIAL_SAVE_NEW_CODE:
                pcSerialComSaveNewCodeUpdate( receivedChar );
                break;
            case PC_SERIAL_SET_DATE_AND_TIME:
                pcSerialComDateAndTimeUpdate(receivedChar);
                break;
            default:
                pcSerialComMode = PC_SERIAL_COMMANDS;
                break;
        }
    }
}
```

3. Al ingresarse un carácter, estando el SerialCom en estado de actualización de fecha y hora, se registra el nuevo carácter utilizando la función:

pcSerialComDateAndTimeUpdate()

Propuesta no bloqueante

```
static void pcSerialComDateAndTimeUpdate(char receivedChar)
{
    static char newDateAndTime[DATE_AND_TIME_NUMBER_OF_CHARS];

    newDateAndTime[numberOfDateAndTimeChar] = receivedChar;
    pcSerialComStringWrite(receivedChar);
    numberOfDateAndTimeChar++;
    if(numberOfDateAndTimeChar >= DATE_AND_TIME_NUMBER_OF_CHARS) {
        pcSerialComMode = PC_SERIAL_COMMANDS;
        numberOfDateAndTimeChar = 0;
        commandSetDateAndTime(newDateAndTime);
        pcSerialComStringWrite("\r\n");
        pcSerialComStringWrite("Date and time has been set\r\n");
    } else if (numberOfDateAndTimeChar == 4) {
        pcSerialComStringWrite("\r\n");
        pcSerialComStringWrite("Type two digits for the current month (01-12): ");
    } else if (numberOfDateAndTimeChar == 6){
        pcSerialComStringWrite("\r\n");
        pcSerialComStringWrite("Type two digits for the current day (01-31): ");
    } else if (numberOfDateAndTimeChar == 8) {
        pcSerialComStringWrite("\r\n");
        pcSerialComStringWrite("Type two digits for the current hour (00-23): ");
    } else if (numberOfDateAndTimeChar == 10) {
        pcSerialComStringWrite("\r\n");
        pcSerialComStringWrite("Type two digits for the current minutes (00-59): ");
    } else if (numberOfDateAndTimeChar == 12){
        pcSerialComStringWrite("\r\n");
        pcSerialComStringWrite("Type two digits for the current seconds (00-59): ");
    }
}
```

```
static void commandSetDateAndTime(char dateAndTimeStr[])
{
    char year[5] = dateAndTimeStr.substr(0, 4);
    char month[3] = dateAndTimeStr.substr(4, 2);
    char day[3] = dateAndTimeStr.substr(6, 2);
    char hour[3] = dateAndTimeStr.substr(8, 2);
    char minute[3] = dateAndTimeStr.substr(10, 2);
    char second[3] = dateAndTimeStr.substr(12, 2);

    dateAndTimeWrite( atoi(year), atoi(month), atoi(day),
        |   atoi(hour), atoi(minute), atoi(second) );
}
```

4. Una vez registrados todos los caracteres necesarios, se actualiza la fecha y hora con la función *commandSetDateAndTime()*

Conclusión

- La propuesta no resulta bloqueante pues al ingresarse la nueva fecha y hora, se registra de a un carácter por vez.
- Con esta implementación, no se frena el código esperando input del usuario en ningún momento. La actualización se realiza únicamente cuando el usuario ingresa un carácter.