Estoy viendo el programa ahora en el trabajo

Está muy prolijo, Te felicito!!!.

El programa no tiene fallas a simple vista.

Aquí lo que veo es algo que no se ve en el papel y es la velocidad.

Intentaré ser lo más claro posible.

Para escribir en una pantalla si no va a haber cambios no hace falta reescribirla si nada cambió.

No tiene sentido volver a escribir siempre lo mismo

Vamos a ver un poco los tiempos

Serial.begin(9600);

Esto significa que vamos a escribir o comunicarnos a una velocidad de 9600 baudios o bits por segundo esa es la frecuencia de trabajo.

Cuando yo saco una (A)= numero 65 en ASCII en el Serial.print por poner un ejemplo , para que esa A sea interpretada por una PC establecemos la reglas de comunicación o Protocolo hay algo implícito que no vemos pero por ahí lo has visto en algún lugar 8N1

8 = BIts de datos

N = sin bit de paridad

1 = Bit de stop

¿Qué quiere decir esto? que vamos a escribir datos a una velocidad de 9600 Bps y que para escribir una letra tenemos que hacerlo transmitiendo 9 Bits si tuviera bit de paridad serían 10 Bits

El periodo es 1/9600 = 0.000104 segundos = a 104 milisegundos por cada bit,

104 \* 9 = 937 milisegundos

Si escribimos 20 letras = 18750 milisegundos = a 0.01875 segundos ya nos vamos acercando a unidades grandes paro un microcontrolador.

OJO estoy hablando de la función Serial.print que es para sacar datos serie.

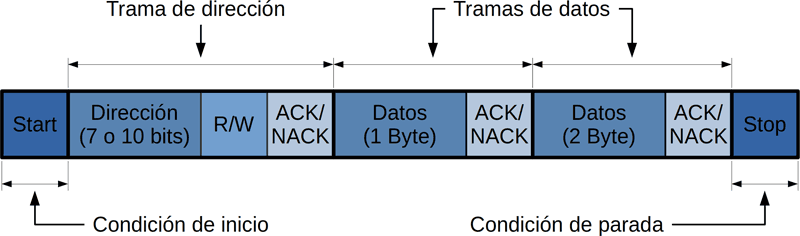
¿Te vas dando cuenta lo que está pasando?

Ahora la función que vos estás usando es para imprimir el **DISPLAY**

lcd.print

Esa transmite por datos serie a la plaqueta del display por **I2C**

¿Sabes cuánto ocupa el protocolo I2C para enviar una A ?



<https://programarfacil.com/blog/arduino-blog/comunicacion-i2c-con-arduino/>

Para que veas con tiempo aquí está muy bien explicado lo mío es un resumen para entender

Fijate la cantidad de bits que se necesitan para transmitir un solo dato cerca de **40 bit**s y tendría que buscar cual es la velocidad de comunicación, pero ahora para escribir **20 letras** necesitamos **20\*40 = 800 bits**

Vos estás en tu programa haciendo esto

lcd.clear();

    lcd.home (); // Mover el cursor a la fila (0) columna (0)

    lcd.print("FECHA:"); // Imprime en Display "FECHA:"

    lcd.setCursor(6,0); // Mover el cursor a la fila (0) columna (6)

    lcd.print(dia); // Imprime en Display la variable "dia"

    lcd.setCursor(8,0); // Mover el cursor a la fila (0) columna (8)

    lcd.print("/"); // Imprime en Display "/"

    lcd.setCursor(9,0); // Mover el cursor a la fila (0) columna (9)

    lcd.print(mes); // Imprime en Display la variable "mes"

    lcd.setCursor(11,0); // Mover el cursor a la fila (0) columna (11)

    lcd.print("/"); // Imprime en Display "/"

    lcd.setCursor(12,0); // Mover el cursor a la fila (0) columna (12)

    lcd.print(anio); // Imprime en Display la variable "anio"

    lcd.setCursor(0,1);  // Mover el cursor a la fila (1) columna (0)

    lcd.print("HORA:"); // Imprime en Display "HORA:"

    lcd.setCursor(5,1); // Mover el cursor a la fila (1) columna (5)

    lcd.print(hora); // Imprime en Display la variable "hora"

    lcd.setCursor(7,1); // Mover el cursor a la fila (1) columna (7)

    lcd.print(":"); // Imprime en Display ":"

    lcd.setCursor(8,1); // Mover el cursor a la fila (1) columna (8)

    lcd.print(minuto); // Imprime en Display la variable "minuto"/\* code \*/

    delay (6000);

Son 20 FUNCIONES que ocupan un tiempo importante en términos electrónicos, claro está pero que no se ven en el papel y si tienen que estar en todo momento en la mente del programador.

Ahora vos no solo estas escribiendo estas llamando a funciones ()

Ejemplo **lcd.home()** para hacer esto debe hacer mucho más que escribir un digito si mal no recuerdo cuando esto se hace a mano se deben enviar varios comandos con ciertos retardos porque el display necesita esos tiempos para ejecutar las ordenes internas y por lo menos 5 o 6 datos al display con sus tiempos.

Así que cada función tardará lo suyo, pero el gran problema sumado a esto son los

6 segundos que le estas agregando al final

Para saber cuánto tarda hay un truco se trata se prender un led al comienzo y apagarlo al final con un retardo de 1 segundo entonces con un osciloscopio se puede ver cuánto tarda en hacer todo esto sin utilizar los 6 segundos adicionales.

Aquí te busque en la librería parte del home

void LiquidCrystal\_I2C::home(){

    command(LCD\_RETURNHOME);  // set cursor position to zero

    delayMicroseconds(2000);  // this command takes a long time!

}

Ya ahí tenés un retardo de 2000 microsegundos = 2 milisegundos

Pero esto llama a otra función

**command(LCD\_RETURNHOME);**

Que hay que buscar, pero seguro hace lo suyo con retardos también en particular los del display.

Vuelo al programa para que vuelva a leer un pulsador hacen falta por lo menos 6 segundos más los otros retardos.

Por eso no te lee tan rápido los pulsadores.

Para eso hay un truco pero quise explicarte bien por que ocurren ciertas cosas difíciles de ver en papel pero que molestan en la práctica.

Entonces los pulsadores hay que leerlos rápido y el display hay que escribirlo de vez en cuando es por eso que en tu caso seguramente viste algún ejemplo le dan el retardo porque si no, no verías nada si constantemente le estás mandando información.

Si puedo hoy te escribo lo que hay que hacer. Una de las tantas posibles soluciones.

Perdón por lo extenso, aunque no lo creas es solo mini resumen, no para desalentarte sino para que comiences a abrir los ojos a los detalles que los programadores nunca ven.

Una cosa es programar otra cosa es saber que estamos programando del punto de vita de lo electrónico del hardware. Ahí es donde los técnicos hacemos la diferencia.

Y eso es lo que les tenes que enseñar a tus alumnos.