Proyecto Final - Trabajo Práctico 3

Circuitos Digitales y Microprocesadores - Facultad de Ingeniería UNLP

Estanislao Crivos Gandini 68449/0

28 de enero de 2021

Introducción

En este informe se detalla lo realizado en el trabajo final de la materia. El proyecto consistió en el diseño de un sistema embebido basado en el microcontrolador ATmega328P y su posterior implementación virtual o real. En este caso se optó por la implentación real. La propuesta para el proyecto se basa en una de las ideas brindadas por la cátedra, en particular, la de diseñar una cerradura electrónica. Para darle un contexto y complejizar un poco más la idea se pensó en un sistema integral de automatizado para puertas deslizantes corredizas. En este contexto, la cerradura electrónica pasa a ser un sistema que destraba, abre, detecta presencia, cierra y traba la abertura, todo en un ciclo de funcionamiento debidamente diseñado. Se pensó este proyecto como un subsistema electrónico, parte de un sistema mayor, por lo que la implementación completa vendría dada por la adición de un sistema electromecánico encargado de realizar la apertura y cierre del paño de la abertura y toda la estructura y montaje de la puerta automática propiamente. Este tipo de puertas podrían pensarse dentro de ambientes sensibles que requieran seguridad y control de la gente que entra y sale del establecimiento, como pueden ser bancos, oficinas, laboratorios, centros de datos, oficinas gubernamentales, etcétera.

En la siguiente sección se detalla el funcionamiento del sistema y se da una descripción detallada del funcionamiento de cada uno de sus periféricos. En las últimas secciones se detallan las tareas futuras a realizar, las conclusiones y el *link* del video en el que se muestra el funcionamiento del mismo.

1. Implementación del sistema

A continuación se detallan en una lista los periféricos utilizados para la implementación del proyecto, controlados a través del microcontrolador ATmega328P. En la figura 1 puede verse una posible implementación real del sistema en una puerta de paño corredizo, con el detalle de la ubicación de los periféricos y en la figura 2 se muestra la implementación circuital del prototipo.

- Servomotor SG-90
- Sensor de ultrasonido HC-SR04
- Relés de baja tensión (2)
- Zumbador
- Display 16×2
- Teclado de membrana 4×4

1.1. Descripción general del funcionamiento

La cerradura electrónica consta principalmente de un sistema interactivo, compuesto por el teclado, el display y el zumbador, que interactúa con el usuario. A través de este sistema el usuario puede ingresar la contraseña de acceso que luego el sistema comprueba para dar o no acceso. Además, permite acceder a un menú de raíz en el cual se pueden tomar acciones de superusuario. El resto de los periféricos (relés, servomotor y sensor) son actuadores internos al sistema y sirven para realizar las acciones de apertura, cierre y detección. El sensor de ultrasonido, por ejemplo, juega un papel clave al momento del cierre de la puerta, detectando obstáculos en el paso de la misma. El servomotor cumple la función de trabar y destrabar el paño corredizo, mientras que los relés son los encargados de dar la señal en alto de apertura o cierre al sistema externo encargado de realizar el movimiento del paño. En la figura 4, al final de este informe, puede observarse un diagrama de flujo que describe, en líneas generales, el funcionamiento de la cerradura electrónica.

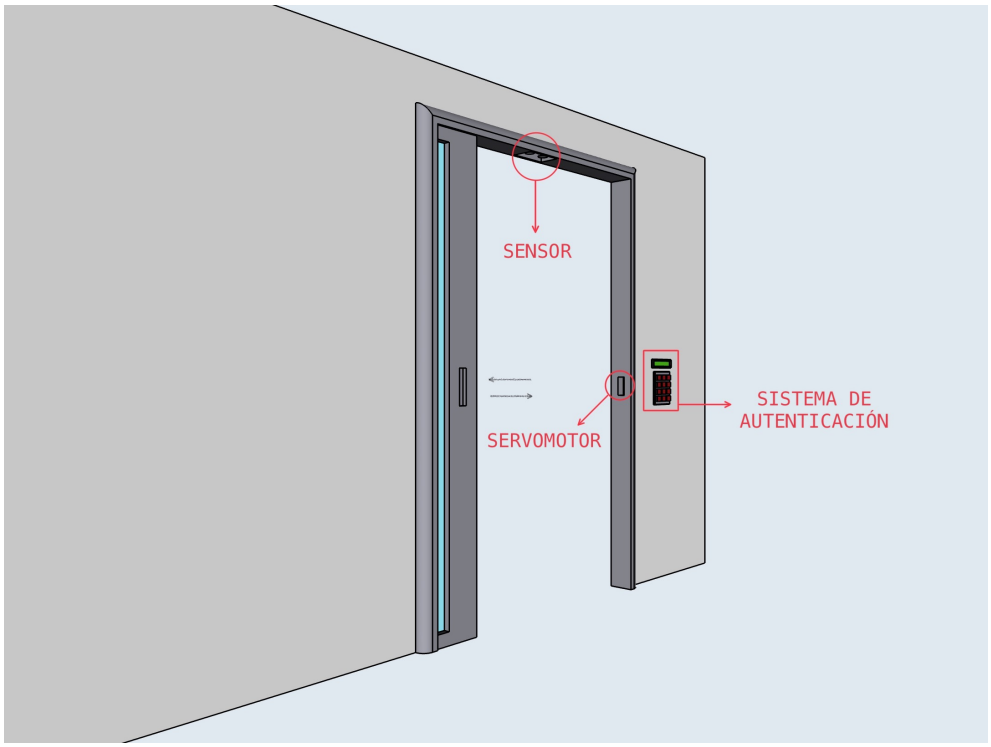

Figura 1: Detalle de la ubicación de los periféricos del sistema en una implementación real.

1.2. Servomotor

Para poder trabajar con el servomotor SG-90 encargado de trabar o destrabar la puerta, primero se consultó la hoja de datos para conocer el funcionamiento del mismo. El motor se alimenta a 5V por dos de sus entradas y necesita una señal pulsada de unos 20ms de período en la tercer entrada. Esta señal es del tipo PWM (*pulse-width modulation*) y dependiendo del ancho de pulso de la misma el motor posiciona su eje en un determinado ángulo entre 0 y 180 grados. Para esta aplicación se pensó directamente en utilizar uno de los tres temporizadores que brinda el microcontrolador. Particularmente se hizo uso del Timer 2. Se lo configuró en el modo *fast PWM* con valor a comparar en el registro *OCR2A* y máximo en *ICR1*, es decir que el temporizador pondrá en alto el puerto *OC2A* (puerto B3) por un determinado tiempo hasta que logre la igualdad entre el número de *ticks* del contador y el numero de *ticks* presente en el registro *OCR2A*. Se configuró el prescaler en 1024 de forma tal que, considerando $f_{clk} = 16MHz$, el período del timer es de $64\mu s$:

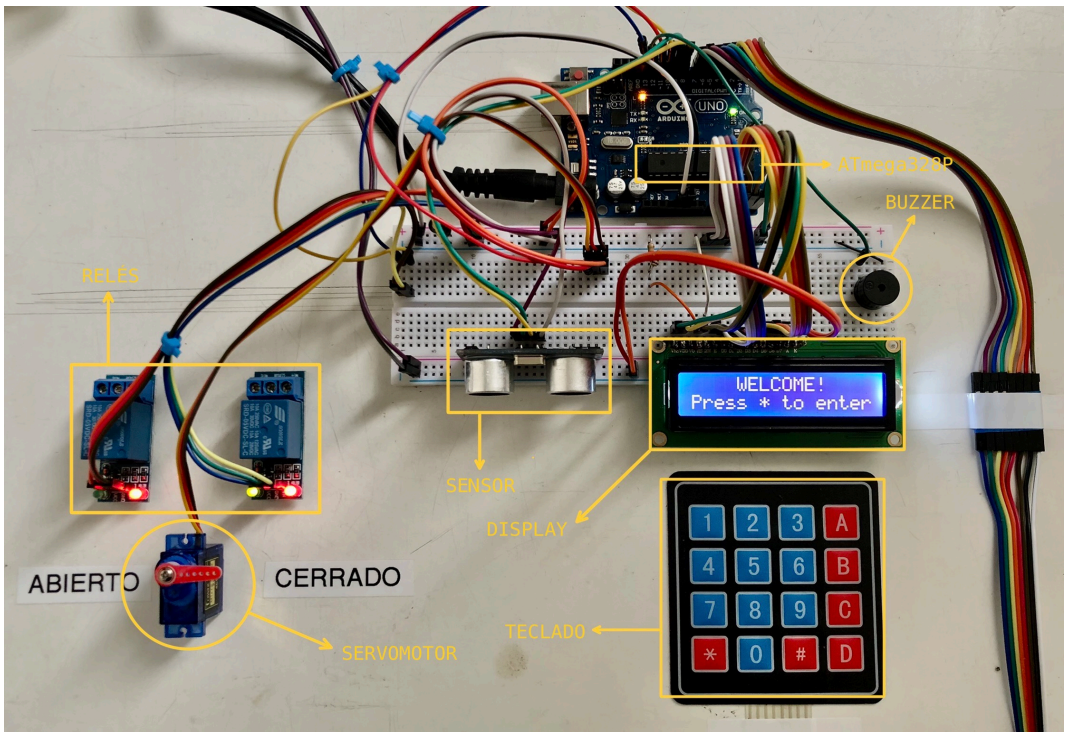

Figura 2: Prototipo del sistema implementado y detalle de los periféricos utilizados.

$$T_{timer} = \frac{1024}{16MHz} = 64\mu s \tag{1}$$

En el registro *ICR1* se coloca el número de *ticks* totales por período para lograr que se reinicie el contador llegado este punto. En este caso, dado que el período del contador es de $64\mu s$, entonces se configuró este registro en 250 de forma tal de lograr un período de 16ms, suficiente para el correcto funcionamiento del servomotor (dado que este es un registro de 8 bits al igual que el temporizador utilizado, no podría conseguirse un período mayor a este).

$$T_{PWM} = \frac{1024}{16MHz} * 250 = 16ms \tag{2}$$

En cuanto al ancho de pulso necesario para lograr cada posición angular, se obtuvo de hoja de datos los valores necesarios para lograr que el eje se posicione en 0 y 180 grados, estos son:

$$T_{alto} = 1ms - 2ms \;\Rightarrow\; \alpha = 0^o - 180^o \tag{3}$$

Haciendo T_{alto}/T_{timer} se obtiene el número de *ticks* del contador necesarios para lograr cada posición del eje. Verificándolo con el servomotor se debieron ajustar mínimamente los valores obtenidos para el registro *OCR2A* de forma tal de obtener la posición adecuada en 0º (posición cerrado) y 180º (posición abierto).

1.3. Sensor de ultrasonido

Para hacer uso del sensor de ultrasonido HC-SR04, nuevamente se refirió al manual del fabricante donde se describe el funcionamiento del mismo. En este caso el sensor es utilizado como sensor de barrera para detectar la presencia en el paso de la puerta automática. Este sensor posee cuatro líneas, por dos de ellas se alimenta y por otras dos recibe y emite señales que son utilizadas por el microcontrolador para calcular la distancia medida. Una de estas líneas es la de *trigger* y la otra es la de *echo*. Como puede suponerse por los nombres que llevan, una línea se encarga de dar la señal

de disparo al sensor y por la otra línea el sensor devuelve un pulso, el cual es utilizado por el μC para calcular la distancia registrada.

Particularmente, la señal de *trigger* debe ser un pulso de $10\mu s$ de ancho. Este pulso le avisa al sensor que debe realizar el disparo de 8 pulsos de ultrasonido de 40kHz. En este momento la señal de *echo* se pone en alto. Estos pulsos de ultrasonido rebotan en la superficie que tenga en frente el sensor y vuelven a él. Al volver, la señal de *echo* se pone en bajo y se reinicia el ciclo de medición. El tiempo que la señal de *echo* duró en alto se puede utilizar para calcular la distancia medida por el sensor, de la forma:

$$D[cm] = \frac{T}{2} * 34300cm/s \tag{4}$$

Donde T representa el tiempo que estuvo en alto la señal de *echo*. Para medir esta señal de *echo* se hizo uso del Timer 1. Este temporizador tiene la posibilidad de ser configurado para recibir una interrupción externa por el pin *ICP1* (puerto B1) y que esta sirva como señal de disparo para el contador. La línea *echo* del sensor fue conectada a este pin y la línea de *trigger* al pin B0. Para generar la señal de disparo se hizo uso de una rutina bloqueante con un delay de $10\mu s$. Para medir el tiempo transcurrido una vez disparado el contador a partir de la señal de *echo*, se hizo uso del Timer 1 configurado en el modo normal de funcionamiento, el cual hace que el contador cuente desde 0 hasta el máximo valor (2^{16}) y se reinicie. Este conteo lo hace con un período de $1/16MHz$ ya que no se utiliza ningún *prescaler*. Al reiniciarse, el Timer ejecuta una interrupción, en la cual la variable *timer_ overflow* es incrementada. Al ponerse en bajo la señal de *echo*, se hace uso del valor almacenado en esta variable para contar el número k de *ticks* transcurridos, de la forma:

$$k = 2^{16} * timer\_overflow \tag{5}$$

$$T = \frac{k}{16MHz} \tag{6}$$

De esta forma entonces puede obtenerse la distancia medida por el sensor de ultrasonido.

1.4. Teclado matricial

Para hacer uso del teclado matricial se siguió parte de lo visto en las clases prácticas de la materia. El teclado posee 8 líneas de control, 4 correspondientes a las filas y 4 correspondientes a las columnas. Estas líneas se conectan entre sí al presionar una tecla, por lo que se conforma una especie de matriz (de allí el nombre) de 16 posiciones. En este caso, las filas del teclado fueron configuradas como salidas y las columnas como entradas. Para escanear la teclas y poder detectar el dígito ingresado en cada caso se hace uso de la función *keypad_ scan*, la cual utiliza una estructura *for* que va poniendo en bajo una fila (salidas) a la vez, mientras que las columnas (entradas) son todas configuradas como entradas en *pull-up*. Al presionar una tecla, se pone en bajo la columna correspondiente. Conociendo qué fila se puso en bajo al momento del ingreso, puede definirse la tecla que se presionó. Esto se hace utilizando dos estructuras *case* anidadas. La primera selecciona el caso según la columna que se puso en bajo y el segundo *case* decodifica la tecla ingresada, asignándole a la variable de salida de la función el valor correspondiente según el número que esta impreso en cada tecla del teclado matricial, basándose en la variable de iteración (la cual define implícitamente la fila que se puso en bajo en primer lugar). De esta forma la función devuelve el número ingresado del 0 al 9 (se omitió el uso de las teclas alfabéticas).

1.5. Display

El display es una parte fundamental del sistema ya que es el encargado de interactuar con el usuario. Para controlarlo se hizo uso del puerto C completo (6 pines), por lo que se utilizó la conexión

de 4 bits de datos y 2 bits de control. Para comunicarse se implementó una librería con 4 funciones: de inicialización, de pasaje de comandos, de posicionamiento del cursor, de escritura de un caracter y de escritura de una cadena de caracteres. Para la función de inicialización se siguieron los comandos propuestos por el fabricante (Hitachi en este caso). Esta inicialización prepara el chip para escribir los caracteres almacenados en su memoria. La rutina tanto de pasaje de comandos como para escribir un caracter constan en tomar el caracter o comando de 8 bits, separarlo en su parte alta y baja respectivamente y mandar cada una por vez a los 4 pines de entrada de datos, mientras que se mantiene el pin *Enable* en alto y el pin *RS* en alto o bajo según sea un caracter o un comando lo que se envía. Los retardos que se utilizan para la señal de habilitación o de selección se implementan utilizando la función *_delay_ms()* de la librería *util/delay.h*. La función de escritura de una cadena de caracteres no es más que una estructura *for* que llama iterativamente a la función de escritura de caracteres. La función que posiciona al cursor en un determinado lugar del display se implementó siguiendo lo propuesto por el fabricante.

Este display se alimenta de 5V, tanto para su funcionamiento como para el LED que funciona de *backlight*. Los pines utilizados para su conexión como también para el resto de los periféricos se detallan en la figura 3.

1.6. Relés

Los relés utilizados cumplen la función de dar una señal de aviso en alto o bajo al sistema externo electromecánico (no implementado) para notificarle que realice el cierre o la apertura de la puerta. Son accionados por nivel bajo (*normally-open)*. Al ser dos, uno se encarga de dar aviso de apertura y el otro de cierre. El que se mantenga en alto (llave cerrada) es aquel que da la señal de aviso al sistema externo. Estos relés están diseñados para manejar tensiones de alterna de hasta 250V y de continua de hasta 30V y corrientes de hasta 10A en ambos casos. Son alimentados con 5V y son controlados por el microcontrolador utilizando una línea que, al ponerse en bajo, acciona el interruptor, cerrándolo.

1.7. Buzzer

El zumbador se encarga de dar señales sonoras de confirmación o aviso al usuario. Es alimentado directamente desde uno de los pines del microcontrolador.

2. Implementación en C

A continuación se describe en forma general el funcionamiento del algoritmo implementado para el proyecto. El código se encuentra debidamente comentado, por lo que puede seguirse fácilmente.

Para lograr un código más estructurado y fácil de mantener, se armaron librerías para todas aquellas partes del código que puedan ser llamadas desde el *main* y puedan funcionar independientemente del mismo. Esto se hizo para las funciones que involucran al teclado (librería *keypad.h*), al display (librería *LCD_display.h*) y al servomotor (librería *servomotor.h*). Al inicio del código se incluyen todas estas librerías, se definen las funciones que son utilizadas en el mismo código y se definen variables y arreglos globales que serán leidos y modificados a lo largo de la ejecución.

Dentro del *main*, previo al *loop* principal, se configuran los puertos utilizados, seteándolos como entrada o salida, en bajo, en alto o en *pull-up*. Finalmente se llama a todas las funciones que inicializan el display, el teclado, el servomotor y el modo bajo consumo. El display es inicializado y puesto en espera con el texto *Welcome! Press * to enter*. El servomotor es inicializado con el eje en 0 grados (estado cerrado) y los relés en modo cerrado también. El teclado numérico es inicializado de forma tal que al ingresar el dígito * se genere una interrupción por el pin *INT1* (sólo pin D4 en bajo). El modo de bajo consumo es inicializado al final. Se optó por utilizar el modo *Power Down* ya que

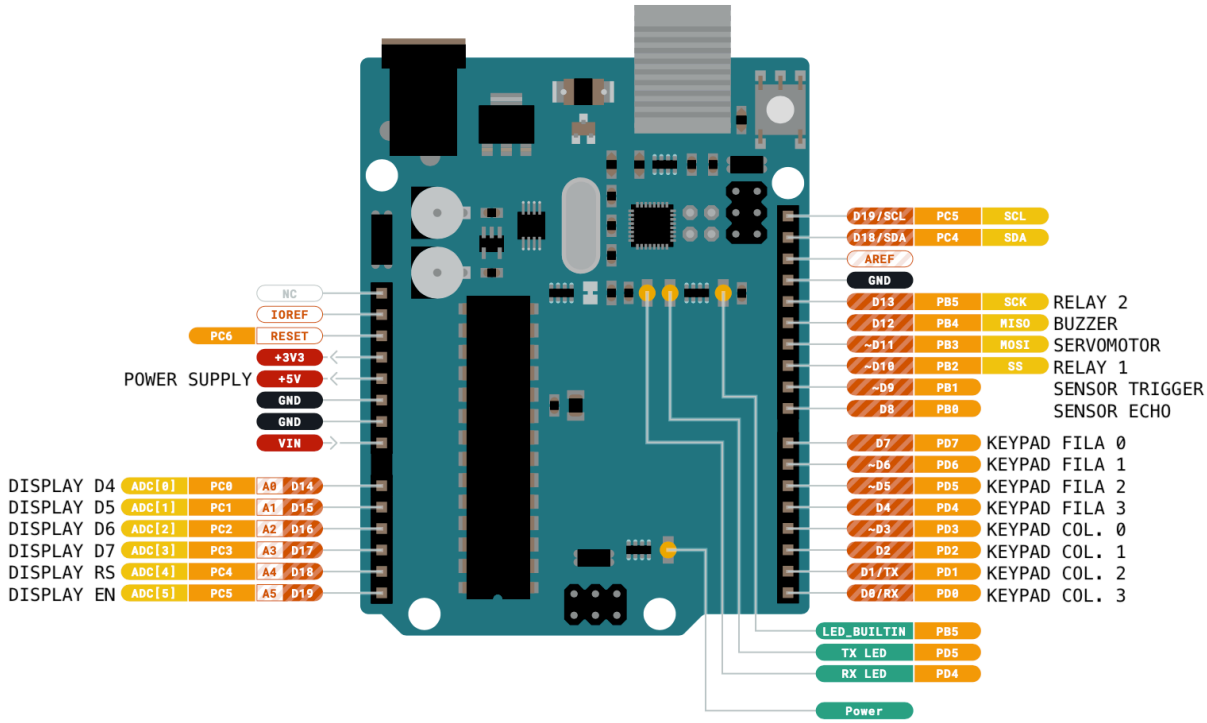

Figura 3: Configuración de puertos utilizada para los periféricos.

es uno de los más ahorrativos y permite que se interrumpa por nivel bajo en el pin *INT1*. De esta forma, el sistema queda en *stand-by* a la espera de una interrupción por teclado.

Dentro del *loop* principal (*while(1)*) se encuentra todo el código que será llamado constantemente. Dentro hay otra estructura *while* que lee el valor de la variable *key_flag*. Esta variable es puesta en alto en la ISR (*interrupt service routine*) correspondiente al *INT1*. Al ponerse esta variable en alto, se pide por pantalla que se ingresen los 4 dígitos de la contraseña. Para ello, dentro del *while* se llama a la función *keypad_scan* constantemente. Al ingresar cada dígito, estos son almacenados en un arreglo. Ingresados los cuatro dígitos, el programa entra a una estructura *if* que comprueba si la contraseña ingresada es la correcta o no de acuerdo a la que tenga almacenada en memoria por defecto, utilizando la función *compare*. Si es correcta, se admite el acceso llamando a la función *admit_access*, sino, se niega el acceso llamando a la función *deny_access*. Existe un caso especial que es cuando se ingresa el código de *superusuario*. Este código es único y viene dado por el fabricante. A continuación se detalla el accionar de cada subrutina de acceso.

La rutina *admit_access* ejecuta una serie de acciones que permiten el acceso al peatón. Esto es, accionar el servomotor, los relés y el sensor. Luego del sensado procede a cerrar la puerta siguiendo los pasos inversamente. La rutina *deny_access* simplemente avisa por display que el acceso fue denegado. Para el modo especial de superusuario, se armó la función *root_menu* la cual brinda las opciones ya mencionadas, permitiendo sobreescribir la contraseña por defecto o abriendo indefinidamente la puerta, permitiendo luego restablecer el sistema. Por una cuestión de simplicidad, estas rutinas fueron implementadas como funciones dentro del código principal debido a que leen y escriben variables o arreglos globales que son utilizados por otras funciones o dentro del *main* mismo.

Al terminar de ejecutarse cualquiera de estas 3 rutinas de acceso, el sistema se restablece haciendo uso de la función *standby_mode*. Esta resetea las variables globales utilizadas, muestra la pantalla de bienvenida, reinicia el teclado, habilita interrupciones y pone al μC en modo bajo consumo a la espera de una nueva interrupción.

En cuanto al código utilizado para leer la distancia desde el sensor, se implementó la función *measure_distance* que en pocas palabras realiza todos los pasos ya descriptos en la sección correspondiente. Dado que la implementación en código requiere leer constantemente la variable *timer_overflow*

(modificada dentro de una interrupción), no se logró aislar la función *measure_ distance* como una librería por lo que se incluye dentro del código principal al final del mismo. Para determinar la presencia de obstáculos en el paso de la puerta, se utiliza una estructura *while* que compara el valor medido por el sensor con un valor umbral, el cual puede ser ajustado dependiendo de dónde se encuentre colocado el sensor. En el caso de la implementación mostrada en la figura 1, este valor umbral sería la distancia desde el sensor al suelo.

3. Demostración del funcionamiento y medición de parámetros

Se adjunta el *link* del video en el que puede verse el funcionamiento del subsistema implementado: `https://youtu.be/Nhiu9yL8T1A`. Lo que se puede ver en el video es el prototipo circuital dispuesto a modo demostrativo donde puede verse el accionar de los actuadores y sensores en cada modo de funcionamiento. Cabe destacar que para utilizar el sensor de ultrasonido se utilizó un valor umbral de 20cm como límite admitido por el microcontrolador para realizar el cierre de la abertura y se utilizó un obstáculo simple para simular un obstáculo real en la implementación real del sistema.

Se realizó una medición de la corriente consumida por el Arduino UNO, sin periféricos y con periféricos, en modo *sleep* y en modo normal de funcionamiento para determinar el ahorro de corriente mientras el microcontrolador se encuentra en *stand-by*. Para energizar el Arduino se utilizó una fuente de 12V 500mA. El consumo del Arduino UNO sin periféricos conectados es de 42mA. El consumo del mismo pero con todos los periféricos conectados y en modo *sleep* es de 118mA, lo que entonces da lugar a una corriente de 76mA de consumo por parte de los periféricos en modo bajo consumo. Al despertar el μC se obtiene una corriente neta consumida de 130mA, lo que da lugar a una corriente de 88mA, por lo que la diferencia de consumo entre el modo bajo consumo y el modo normal es de 12mA. Obviamente que este es un cálculo aproximado ya que no tendría en cuenta los consumos variables que existen durante el funcionamiento, como pueden ser los picos de corriente que consume el servomotor al mover su eje (consumo de hasta 200mA). El LED que ilumina el display fue conectado aparte pero su consumo es de unos 20mA.

En cuanto a la performance del sistema, no se creyó necesario realizar la medición del conjunto debido a que .

4. Tareas futuras

Uno de los aspectos que tal vez se podría modificar a futuro es el tipo de sensor que se utiliza para detectar presencia. En ese sentido con un simple sensor de movimiento bastaría y hasta podría ser más eficiente. Otra cuestión que podría agregarse es una funcionalidad anti-pellizco, que mientras se ejecute el cierre de la puerta pueda aún detectar presencia y reabrirla si fuese necesario.

En cuanto al algoritmo, podría mejorarse para hacerlo más estructurado aún y dividirlo aún más en distintos bloques más fáciles de mantener a largo plazo (por ejemplo las funciones ejecutadas en cada modo de funcionamiento).

En cuanto a la comunicación con el display o el teclado, podría pensarse a futuro en utilizar algún módulo I^2C para interactuar y así ahorrar puertos y conexiones.

En cuanto a la forma de acceso, sería útil poder almacenar distintos códigos de acceso para que distintos usuarios puedan tener el suyo. Además, podría a futuro pensarse en alguna forma de encriptar estos códigos y almacenarlos en memoria no volátil para así volver más seguro al sistema.

Por último, también sería necesario diseñar los montajes donde se ubicaría cada periférico en una instalación real y cómo quedarían aseguradas las partes más sensibles para que no puedan ser intervenidas.

5. Conclusiones

Personalmente, quedé conforme con lo logrado. Creo que hacer este proyecto fue una buena forma de darle un cierre a la materia y comprender el real uso y funcionamiento de los microcontroladores, dando noción de lo importante que son en los sistemas automáticos o semi-automáticos con los que uno interactúa en el día a día.

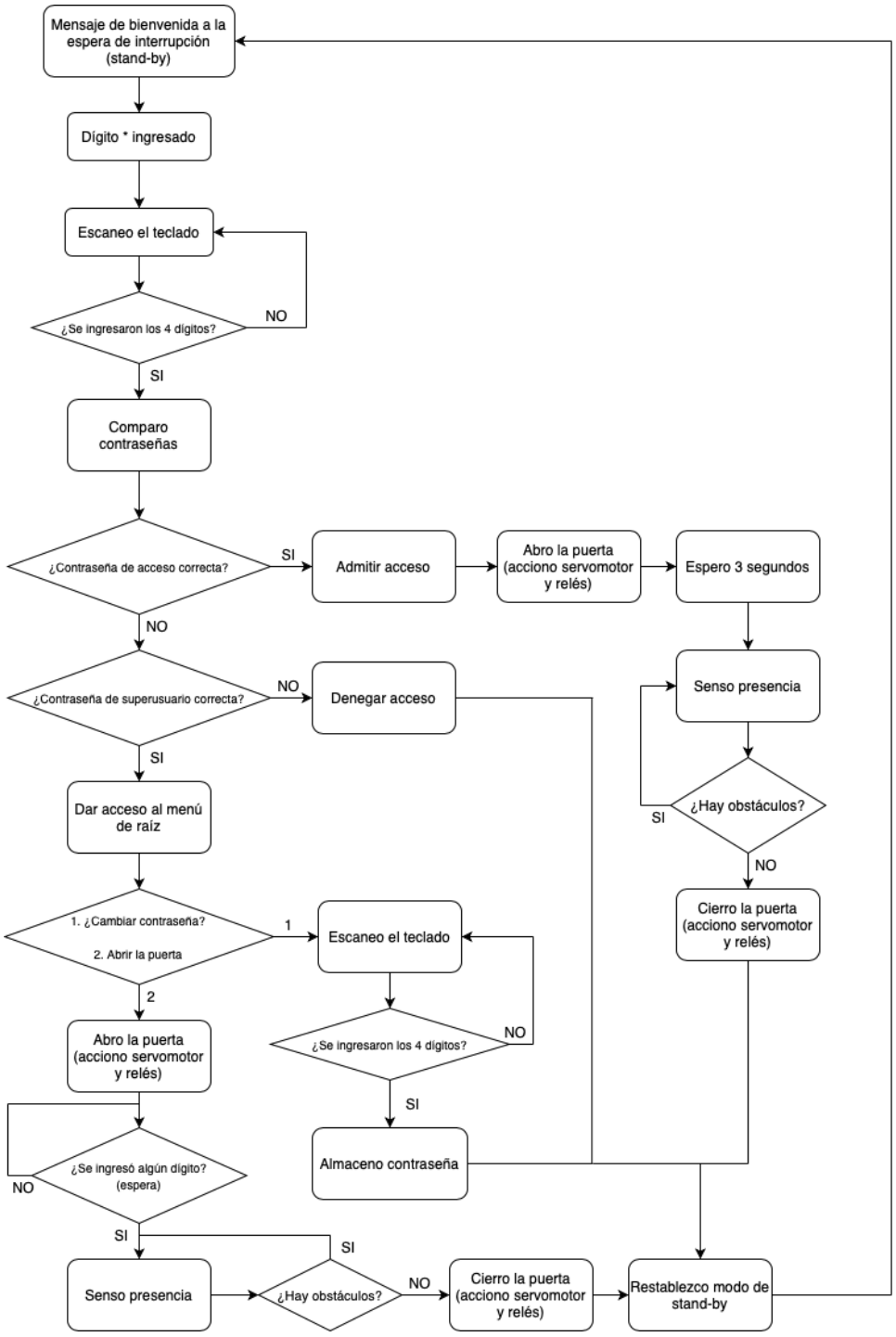

Figura 4: Diagrama de flujo que describe el funcionamiento del sistema.