Título: Aspiradora Robot

LPC845 - “Aspiradora automática”

Ciclo Lectivo:	2022	Curso:	R2002

Integrantes	Apellido Nombres	Legajo	Calificación individual	Fecha
MARTÍNEZ, Agustín Damián		2032790		
SALERNO, Santiago Nahuel		2033112		
SPATARO, Guido Franco		2033227		

Calificación grupal:		Fecha:

Profesor:	TRUJILLO, Marcelo Ángel
Auxiliar/es Docente/s:	BUA, Federico Nicolás

Canje por 2° parcial	**SI** **NO**

Observaciones primera entrega	
Observaciones segunda entrega	

ÍNDICE

Desarrollo de la idea fuerza:

Nuestra idea fuerza consiste en la realización de una aspiradora automática que pueda ser manejada a distancia desde una aplicación mediante una PC o de forma "manual" con el teclado incorporado en la misma. Ésta deberá evitar chocar con las paredes u obstáculos con los que se encuentre y generará una base de datos con los tiempos de uso que se le fueron dando. Como última adición a nuestra idea, la aspiradora deberá poder moverse en las cuatro direcciones posibles (adelante, atrás, izquierda y derecha) de forma remota con el uso de la aplicación. Esto para entregar al usuario un control de seguridad en caso de que se encuentre atrapada o no pueda ir hacia ningún lado.

Introducción:

Objetivos:

Los objetivos de este trabajo práctico son los siguientes:

- Realizar una aplicación que permita controlar nuestro microcontrolador.
- Realizar un programa en un sistema embebido que cumpla con todos los requerimientos de nuestro desafío.
- Crear de forma apropiada los drivers de nuestro microcontrolador utilizando la hoja de datos correspondiente.
- Realizar algún método de comunicación entre la aplicación de computadora y el microcontrolador.

En cuanto a nuestro desafío en particular, los objetivos serán los siguientes:

- Poder detectar las paredes tanto delante como a los costados de la aspiradora.
- Realizar un algoritmo de barrido que funcione en la mayoría de los casos posibles.
- Realizar una base de datos que permita guardar todas las sesiones de aspirado realizadas.

Diagrama en bloques:

Para nuestro programa de computadora se utilizaron las siguientes páginas:

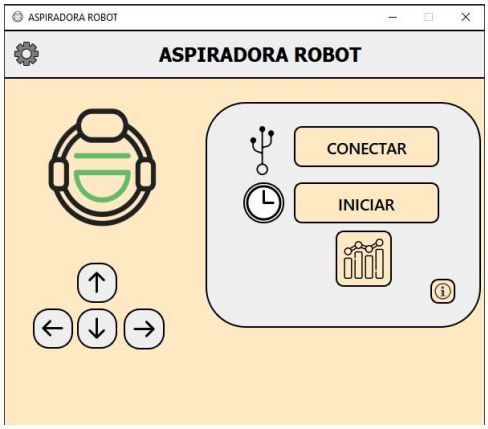


Imagen 1: Página principal.

Esta página es la principal. Desde acá se iniciará nuestra aplicación y se irá al resto de las opciones disponibles. Posee un botón para conectar con el NodeMCU (servidor Wi-Fi), uno para iniciar una limpieza, uno para ver el historial de limpiezas previas, un engranaje que lleva a la configuración y cuatro flechas. Estas flechas son las que permitirán el movimiento "manual" de la aspiradora.

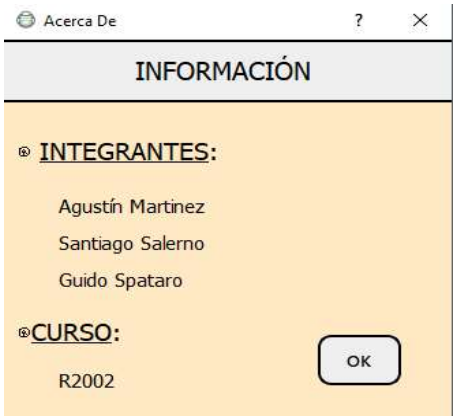


Imagen 2: Página de información.

La página de información es la más sencilla de todas. Simplemente mostrará a los integrantes y curso de la aplicación. No posee ninguna funcionalidad especial.

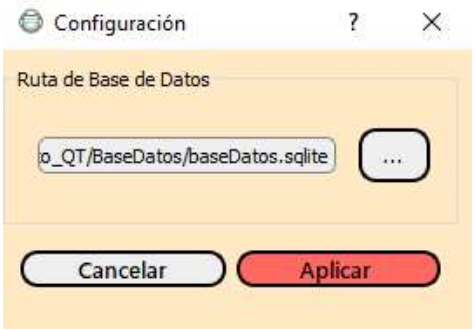


Imagen 3: Página de configuración.

Esta página permitirá al usuario elegir dónde querrá guardar la base de datos en la que se guarda el historial de limpiezas. Por defecto se encontrará junto al lugar de instalación de la aplicación.

Imagen 4: Página con el historial.

En esta página se podrán visualizar todas las sesiones de aspirado en forma de un historial. Podrán ser eliminadas a voluntad y actualizadas en caso de que hayan ocurrido cambios en medio de su lectura.

Imagen 5: Página de inicio de aspirado.

En esta página se inicia la sesión de aspirado. Se puede indicar que se realice en este preciso momento (inicio rápido) o dentro de un rato (inicio programado). La orden es guardada en la base de datos y enviada a la aspiradora de forma inmediata. Como protección, el programa verificará utilizando la base de datos, que la aspiradora no se encuentre aspirando en los horarios pedidos.

Para nuestro programa del sistema embebido se utilizó de base el siguiente diagrama en bloques:

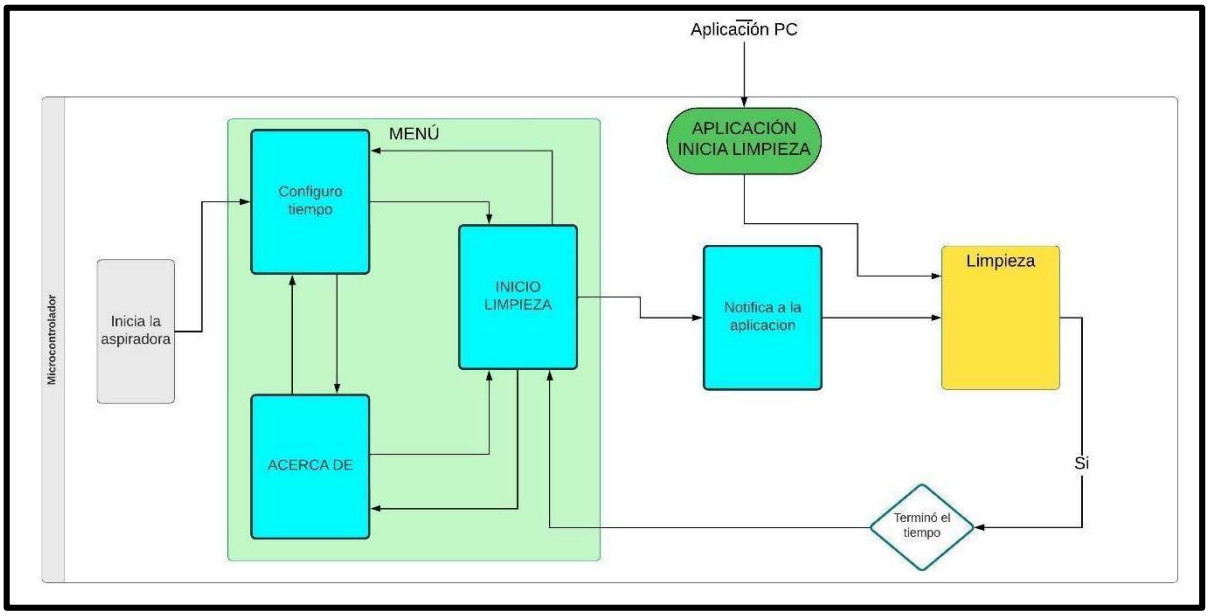


Imagen 6: Diagrama en bloques del LPC845.

Descripción detallada:

Bloques:

Inicio de la aspiradora:

La aspiradora iniciará encendiendo la pantalla y escribiendo en la misma “Aspiradora robot, Iniciando”. Seguido a ello comenzará a escribir puntos suspensivos (…) de forma pausada. Una vez que termine con eso, irá al menú. Mientras se esté ejecutando esta presentación no hará caso de ningún comando del usuario o de la aplicación.

Menú:

El menú está conformado por tres pantallas a las que se puede acceder con el teclado, seleccionando izquierda o derecha para cambiar entre ellas. Estas tres pantallas son: Configuro tiempo, Acerca de e Inicio limpieza.

Configuro tiempo:

En esta pantalla del menú se podrá seleccionar el tiempo que se desea aspirar. Al entrar a esta opción se nos presentará por pantalla el tiempo a elegir y las acciones de sumar o restar minutos. Con el teclado se elegirá el tiempo y quedará guardado el último antes de salir de la configuración. Por defecto el tiempo será de 20 minutos y tendrá un máximo de 255 minutos (equivalente a cuatro horas y quince minutos). En caso de haber realizado una barrida ordenada por la aplicación, el valor del tiempo quedará guardado con el de dicha barrida externa. Este puede volver a ser cambiado en este menú.

Acerca de:

En esta pantalla del menú simplemente se imprimirá por pantalla el nombre de los integrantes del grupo. Se decidió por indicar esto de forma de reconocer a los autores de este.

Inicio limpieza:

Si se selecciona esta opción en el menú, el dispositivo empezará a limpiar utilizando el tiempo que tenga cargado. Es importante saber que el dispositivo no volverá a preguntar cuánto tiempo desea aspirar y

tomará automáticamente el ya elegido en configurar tiempo. También cabe aclarar que, en caso de haber barrido por orden de la aplicación, el tiempo guardado será el de ese barrido y no el que se seleccionó manualmente antes.

Notifico a la aplicación:

Cuando el dispositivo inicia una barrida indicada por hardware, este le avisará de la misma a la aplicación. Simplemente utilizará la comunicación para enviar un mensaje con el protocolo similar a: <L0032> representando esto una limpieza de 32 minutos.

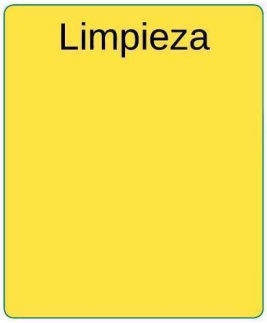
Aplicación inicia limpieza:

Si la limpieza se inicia de forma externa al dispositivo, esta comenzará automáticamente sin importar en qué momento se encuentre. Esto significa que, si la aplicación lo ordena, la aplicación puede salir del menú configurar hora para iniciar la limpieza programada. Esta interrupción del funcionamiento normal del dispositivo será detectada por la máquina de estados encargada de la comunicación.

Si bien el bloque se llama aplicación inicia limpieza, esto es igual de válido para las acciones de movimiento: subir, bajar, izquierda, derecha. Estas acciones interrumpirán cualquier funcionamiento del dispositivo y comenzarán a mover la aspiradora de la forma indicada. Para cancelar estos movimientos y volver al funcionamiento normal simplemente se debe reenviar el movimiento. Ejemplo: si le indico a la aspiradora que avance, luego gire y finalmente retroceda, la acción actual sería retroceder. Para que esta se detenga, simplemente le tengo que volver a decir que retroceda.

Al finalizar los movimientos accionados por el usuario, la aspiradora volverá a la última pantalla de menú que tuvo.

Limpieza:



Si el dispositivo llega a este bloque, comenzará a realizar una limpieza durante el tiempo determinado en el menú o el dado por la interrupción de la aplicación de PC.

La limpieza utilizará el algoritmo de funcionamiento que se encuentra detallado en la sección Máquina de estados.

Especificaciones:

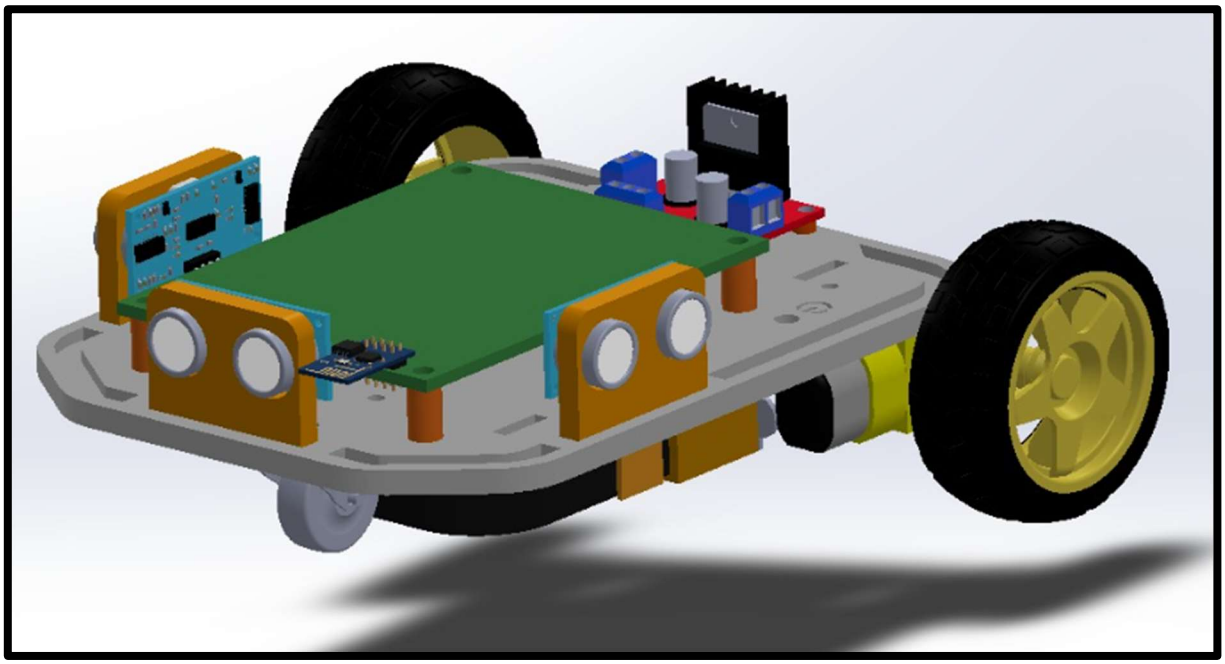


Imagen 7: Maqueta realizada en 3d.

Para la realización de la maqueta en este caso se optó por diseñarla en Solidworks para luego imprimirla con la impresora 3D. Como se puede apreciar en la imagen se buscó tener un Hardware pequeño y robusto al mismo tiempo. Este posee una dimensión de 190x150cm elevada aproximadamente 4cm del suelo.

Para la realización del dispositivo se tuvo que inventar un protocolo de comunicación de forma que se puedan relacionar la aplicación de PC con el dispositivo.

El protocolo es el siguiente:

<	Bit de inicio.	<CU>	Avanzar.
>	Bit final.	<CD>	Retroceder.
C	Indica movimiento de control.	<CR>	Girar a la derecha.
L	Indica limpieza actual.	<CR>	Girar a la izquierda.
P	Indica limpieza programada.		
		<LXXXX>	Limpiar XXXX minutos. (siendo x un número)
		<PXXXX/YYYY>	Limpiar XXXX minutos dentro de YYYY minutos (siendo X e Y números).

Imagen 8: Protocolo de comunicación.

Para el software del dispositivo se utilizó el programa MCUXpresso y para la aplicación de PC se utilizó Qt Creator 5.0.2.

Para tener un historial de todas las sesiones de aspirado fue utilizada una base de datos del tipo SQLite (al ser de Qt pasa a ser QSQLite).

Para la comunicación se utilizó el nodeMCU programado con el software Arduino y sus librerías correspondientes.

Descripción del hardware:

Para la realización de la aspiradora se utilizaron los siguientes elementos:

- Kit de desarrollo del LPC845 realizado por la cátedra.
- Placa Shield para el LPC845.
- Display LCD 16x2.
- Sensor ultrasónico HC SR04.
- Módulo detector de obstáculos infrarrojo FC-51.
- Módulo doble puente H L298n.
- Motores 5V.
- Teclado matricial 1x3.
- Cooler de computadora.
- NodeMCU.

Kit de desarrollo del LPC845:

El Kit de desarrollo del LC845 es una plaqueta realizada por la UTN que nos entrega múltiples periféricos ya conectados y listos para utilizar.

Si bien esto es una ventaja, también nos trae el problema de que la mayoría de las entradas/salidas de nuestro microcontrolador ya se encuentran en uso para distintos actuadores/sensores que no se van a utilizar para este trabajo.

Placa Shield para el LPC845:

La placa Shield del LPC845 consiste en un simple circuito que redirige todos los pines inutilizables del kit de desarrollo a una tira de pines donde podrán ser utilizados. Esto nos permite poder utilizar el kit sin necesidad de crear una plaqueta de gran dificultad para el trabajo.

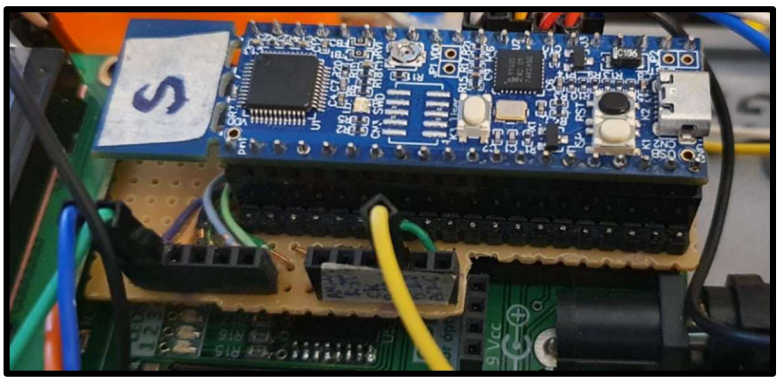


Imagen 9: Shield realizado en placa universal.

Display LCD 16x2:

El display LCD 16x2 es uno de los elementos que se encuentran dentro del Kit de desarrollo. Este componente funciona por medio de 6 salidas digitales del microcontrolador. Cuatro de ellas (D4 a D7) enviarán información al LCD. La salida “E” habilita la lectura de información y la salida “RS” le indica al LCD si se trata de algún comando de configuración o de alguna letra que debe escribir.

Para el funcionamiento se realizó un driver conectado al systick del microcontrolador. Este enviará los comandos de inicialización del LCD en orden. Una vez inicializado, comenzará a imprimir periódicamente por pantalla lo que tenga guardado en su buffer interno.

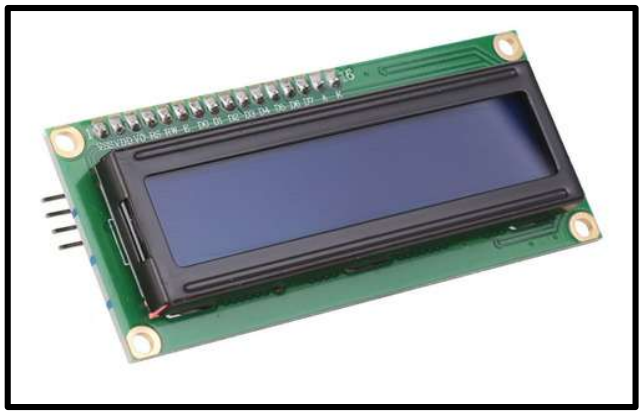


Imagen 10: Display LCD 16x2.

Sensor ultrasónico HC SR04:

El sensor ultrasónico HC SR04 funciona a través de dos patas llamadas Trigger (Trig) y Echo. En la pata Trig se le enviará un pulso periódico de duración igual a 10us. El dispositivo envía ocho pulsos ultrasónicos mientras enciende la pata Echo. El sonido, al rebotar en la pared, regresa al dispositivo. Al recibir de nuevo los ocho pulsos, la pata Echo vuelve a apagarse. La distancia puede ser medida si se cuenta el tiempo de encendido de la pata Echo, teniendo como referencia la velocidad del sonido (340 m/s). Para trabajar con este dispositivo se realizó un driver que activa el pin Trig cada 80 ms (tiempo dado por el fabricante para que no se inicien dos lecturas al mismo tiempo) mientras que se lee el tiempo de encendido de Echo mediante la detección de los flancos ascendente y descendente. Al obtener el valor se realiza automáticamente la conversión a centímetros (cm) del tiempo medido.

Imagen 11: Sensor ultrasónico HC SR04.

Módulo detector de obstáculos infrarrojo FC-51:

El módulo detector de obstáculos consta de dos leds infrarrojos, uno emisor y otro receptor. El módulo utiliza un comparador LM393 para analizar internamente si la lectura de los leds ocurrió lo suficientemente cerca del módulo o no. La distancia umbral de medida puede ser ajustada mediante un potenciómetro incorporado en el mismo. Debido a esto, a diferencia del sensor ultrasónico, este módulo no nos permite medir distancias, sino que, nos permite indicar si algo se encuentra más cerca de una distancia fija. Debido a su sencillo comportamiento, este sensor será utilizado como una simple entrada digital, similar a un pulsador sin rebote.

Imagen 12: Detector de obstáculos FC-51

Módulo puente H L298n:

Para poder conectar dos motores que permitan inversión de marcha se utiliza el módulo de puente H L298n. Esta plaqueta posee un puente H y un conjunto de transistores que nos permiten invertir y controlar la marcha de cualquier dispositivo con tensiones y corrientes relativamente bajas. Para su funcionamiento se utilizan cuatro salidas digitales, dos para cada motor. Dependiendo de cuál de las dos salidas esté encendida, el motor irá en un sentido o en otro. Para simplificar su uso se realizó un driver que combina las cuatro salidas junto a las acciones típicas que se pueden realizar con ellas de forma que pueda responder a los comandos "avanzar, frenar, retroceder o girar".

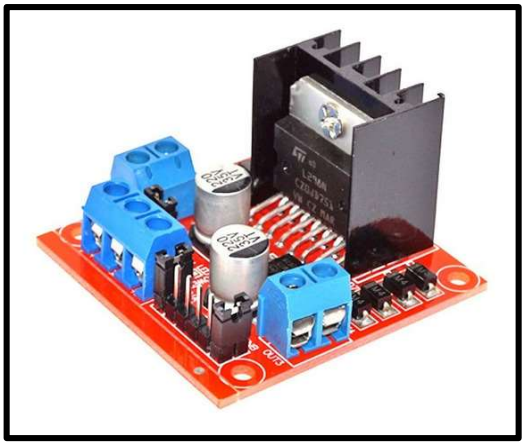


Imagen 13: Puente H L298n

Motores 5V:

Para el desplazamiento de la aspiradora se optó por colocar dos motores de "modelo TT DC Gearbox Moto" de 5V con una rueda por cada motor en el eje. Estos serán conectados al módulo de puente H explicado anteriormente (ver imagen 16) logrando un control sencillo además de aumentar la tensión de los 3,3v entregados por el LPC845 a los 5v correspondientes de los motores. Asimismo, se eligió una rueda universal para que se desplace libremente por la superficie a limpiar.

Imagen 14: Motor de 5v.

Voltaje	Parámetros	3V DC	5V DC	6V DC
Parámetros del motor (sin caja reductora)	RPM	6000 RPM		
	Corriente	80-100mA		
Parámetros de la caja reductora	Reducción	48:1		
	Velocidad sin carga	125 RPM	200RPM	230RPM
	Velocidad con Carga	95RPM	160RPM	175RPM
	Torque de salida	0.8kg.cm	1.0 kg.cm	1.1 kg.cm
	Corriente	110-130mA	120-140mA	130-150mA
	Diámetro máximo de Rueda	6.5cm		
	Dimensiones	70mmx22mmx18mm		
	Peso	50g		
	Ruido	<65dB		

Imagen 15: Especificaciones técnicas del motor.

Conexión con el Puente h:

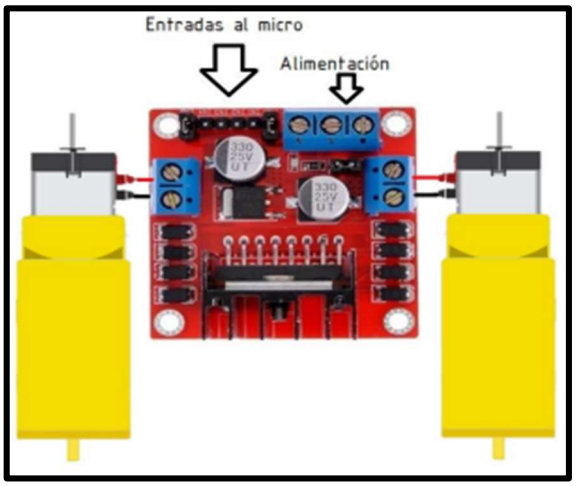


Imagen 16: Conexión entre motores de 5v y puente H.

Teclado matricial 1x3:

Para poder controlar nuestra aspiradora se decidió utilizar un teclado de tres pulsadores, siendo estos: izquierda, ok, derecha. Gracias al kit de desarrollo, ya tenemos a disposición un teclado con formato matricial de 2x3. Para su funcionamiento se utilizan tres salidas digitales y dos entradas. Se enciende una única salida a la vez y se lee cuál de ambas entradas está siendo presionada. Luego de analizar dicha lectura se repite el proceso, realizando un barrido de las salidas seleccionadas. Para finalizar, se guarda el valor presionado en un buffer listo para la lectura por parte de la aplicación. Como el teclado a utilizar es más grande de lo necesario, se tendrán en cuenta solo los primeros tres valores de este y se inicializará el teclado con tres salidas y una entrada.

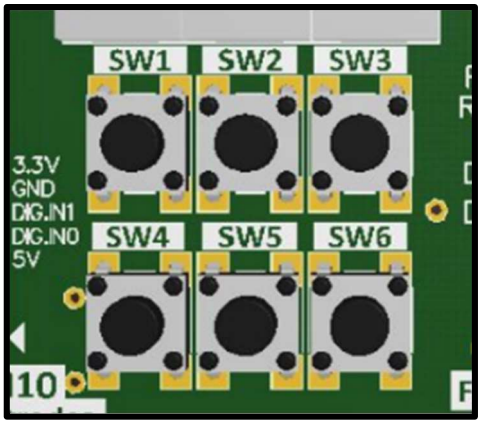


Imagen 17: Teclado matricial utilizado.

Cooler de computadora:

Para simular el aspirado sin tener que meternos en la realización de Hardware innecesario, se optó por utilizar como motor un cooler de computadora. Este motor se encenderá, aspirando lo que se encuentre debajo y lo que recoja se almacenará en un compartimiento de basura. Debido al tipo de motor elegido, su potencia de aspirado no es muy eficiente y solo podrá limpiar migajas o basura relativamente pequeña. Asimismo, se optó por utilizar este motor para poder conectarlo directamente al kit de desarrollo como una salida digital, ya que la diferencia entre usar este motor y uno más potente, no se encuentra en el software sino en el hardware que permita aumentar la corriente que entrega el LPC845.

Imagen 18: Cooler de computadora.

NodeMCU:

Para lograr una comunicación exitosa por Wi-Fi utilizamos el microcontrolador NodeMCU programado utilizando el software Arduino. Su funcionamiento es sencillo, crea un servidor Wi-Fi conectado a la red al que se va a conectar nuestra aplicación de QT y abre un puerto serie que se comunicará de forma cableada con el LPC845. Su código simplemente reenviará todo lo que le llegue por el puerto serie al servidor Wi-Fi y viceversa. Además de recibir y enviar la información, también se comporta como acondicionamiento del puerto serie del LPC845. Por limitaciones del hardware, nuestro microcontrolador no puede recibir más de un byte de información a la vez. Por lo tanto, el NodeMCU enviará la información al puerto serie de forma pausada, un byte a la vez. De esta forma, la información llega al LPC845 en perfectas condiciones para ser leída.

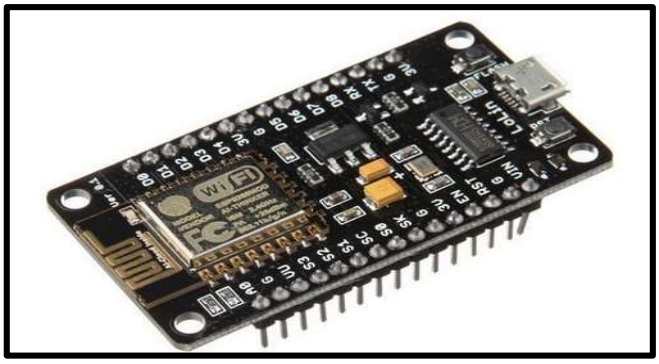


Imagen 19: NodeMCU.

Circuitos:

Para la realización de nuestro dispositivo, solo fue creada una única plaqueta por nosotros. Esta es la "placa shield del LPC845" (ver imagen 9) y su esquemático es el siguiente:

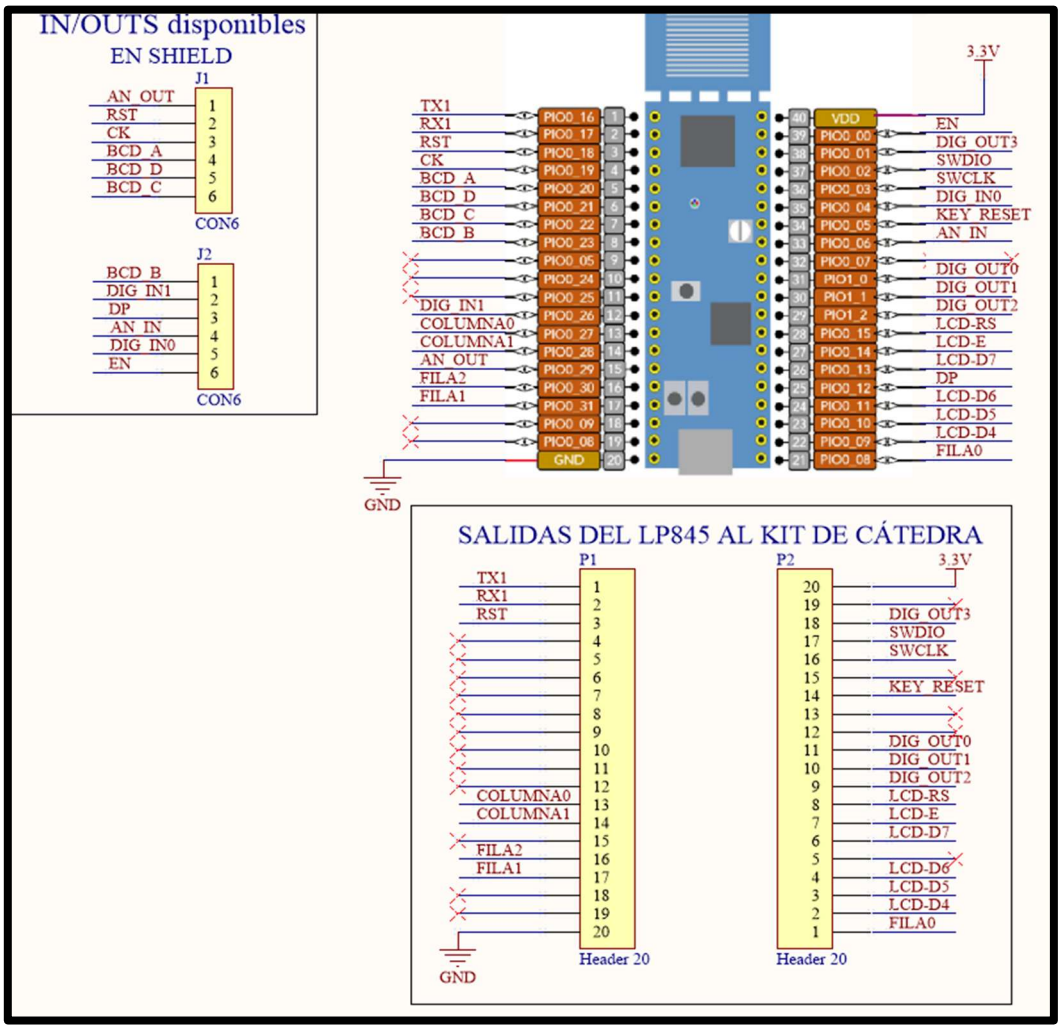


Imagen 20: Esquemático de nuestro Shield.

Hojas de datos:

ESP8266 (módulo en el que se basa el NodeMCU):
https://drive.google.com/file/d/1uy5BJjAh1xArDw2yduMLEBGEFbyT90oy/view?usp=sharing
HCSR04: https://drive.google.com/file/d/1ReJksdO_1ZBupPSjWWA13wK0mxg6pt5A/view?usp=sharing
LPC845: https://drive.google.com/file/d/1bTe09-hPSeEuFnjOtEv3O0i4bMvIvZlL/view?usp=sharing
HD44780u (LCD):
https://drive.google.com/file/d/1dtSg23D6qg0GKwZj_VNNWgOipXMpgPW9/view?usp=sharing
FC-51: https://drive.google.com/file/d/1H_zxyJdc9C3KulbjBeTTdEtkCGQa5tON/view?usp=sharing
Motor 5v: https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/3777_Web.pdf

Máquina de estados:

Debido a la complejidad del dispositivo y las múltiples interrupciones a su funcionamiento normal que pueden ser enviadas por la aplicación de PC se optó por fragmentar la máquina de estados en varias imágenes. De esta forma, se explicarán distintos aspectos de la máquina en cada una de las fotos, eliminando las múltiples transiciones que impedirían la fácil lectura.

Funcionamiento normal del dispositivo:

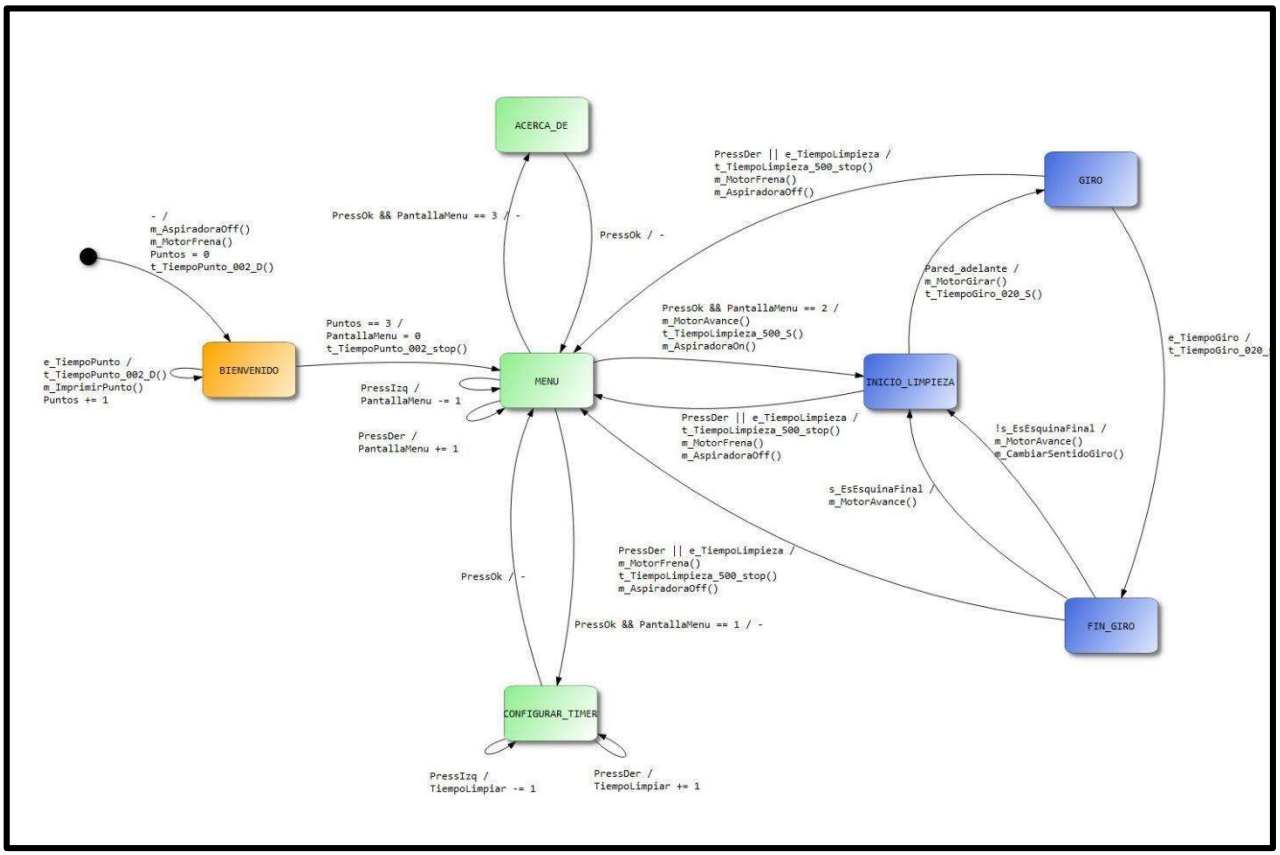


Imagen 21: Máquina de estados de la aspiradora.

En caso de no recibir ninguna instrucción por parte de la aplicación de PC, el embebido respetaría esta máquina de estados. Al iniciar imprime en la pantalla un mensaje de inicialización que irá cargando, utilizando timers. Cuando esa presentación esté terminada, la máquina irá al menú.

El menú podrá imprimir tres pantallas distintas, siendo estas “iniciar limpieza”, “configurar tiempo” y “acerca de”. Esto se ve reflejado en la variable PantallaMenu. Dependiendo de la pantalla del menú, este irá los estados CONFIGURAR_TIMER, ACERCA_DE y INICIO_LIMPIEZA respectivamente.

El estado ACERCA_DE simplemente imprimirá por pantalla los nombres del grupo y se irá si el usuario vuelve a presionar OK.

El estado CONFIGURAR_TIMER utilizará las teclas de izquierda y derecha para modificar el tiempo de limpieza de la aspiradora.

El estado INICIO_LIMPIEZA junto a GIRO y FIN_GIRO en conjunto, son los encargados de realizar el algoritmo de limpieza. Los tres estados regresan al menú si el tiempo de aspirado expiró o si el usuario cancela el proceso (presionando la tecla derecha). El objetivo final es llegar a un algoritmo similar al de la siguiente imagen:

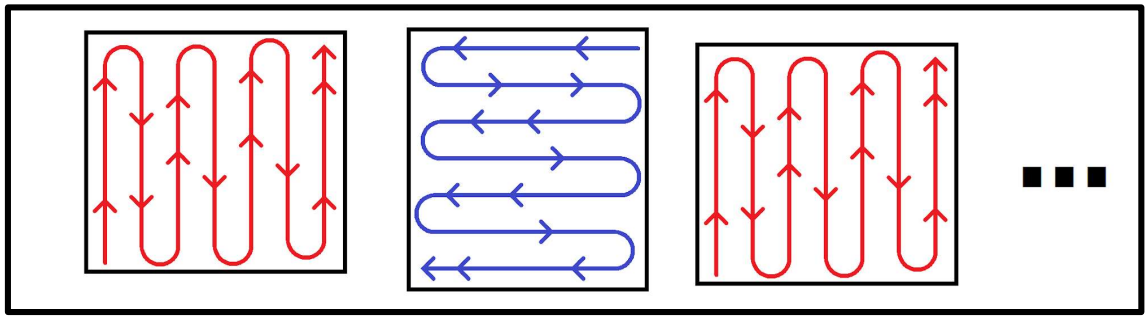


Imagen 22: Funcionamiento del algoritmo de limpieza.

Como se puede ver, la aspiradora irá en línea recta hasta chocar con una pared enfrente suya. Al detectar la pared realizará un giro de 180° hacia una dirección. Luego repetirá este proceso cambiando únicamente el sentido de giro de izquierda a derecha. Cuando llega a la esquina final (aquella donde hay pared del mismo lado del que tiene que girar), la aspiradora debe realizar un giro de 90° en la misma dirección que tomó por última vez. Al realizar este giro, pasará de dar el recorrido rojo al recorrido azul. Cabe resaltar que el primer giro que toma al iniciar el recorrido azul va en el mismo sentido que su giro de 90°.

Traduciendo esto a código, el estado INICIO_LIMPIEZA realizará el recorrido en línea recta de la aspiradora. Al detectar una pared adelante, irá al estado GIRO. Si no se encuentra en la esquina final, el giro lo realizará en el sentido que corresponde, por lo que no debe realizar ninguna acción más. Pero si se encuentra en la esquina final, el sentido de giro no es el que corresponde, por lo que, irá al estado GIRO cambiando el sentido al que debe ir. El estado GIRO simplemente realiza la acción de girar durante un determinado tiempo (correspondiente a 180° o 90°) e irá a FIN_GIRO al finalizar. FIN_GIRO es un estado auxiliar que nos ayuda a visualizar mejor todo el proceso (ya que todo lo realizado en FIN_GIRO podría colocarse a la salida de GIRO). Teniendo eso en mente, FIN_GIRO simplemente guardará el sentido del próximo giro a tomar y volverá automáticamente al estado INICIO_LIMPIEZA repitiendo todo el proceso.

Mientras todo este algoritmo de limpieza está ejecutándose, en pantalla se verá el tiempo restante de la sesión de aspirado, junto a una señal para cancelar el proceso presionando derecha.

Interrupciones de la aplicación de PC:

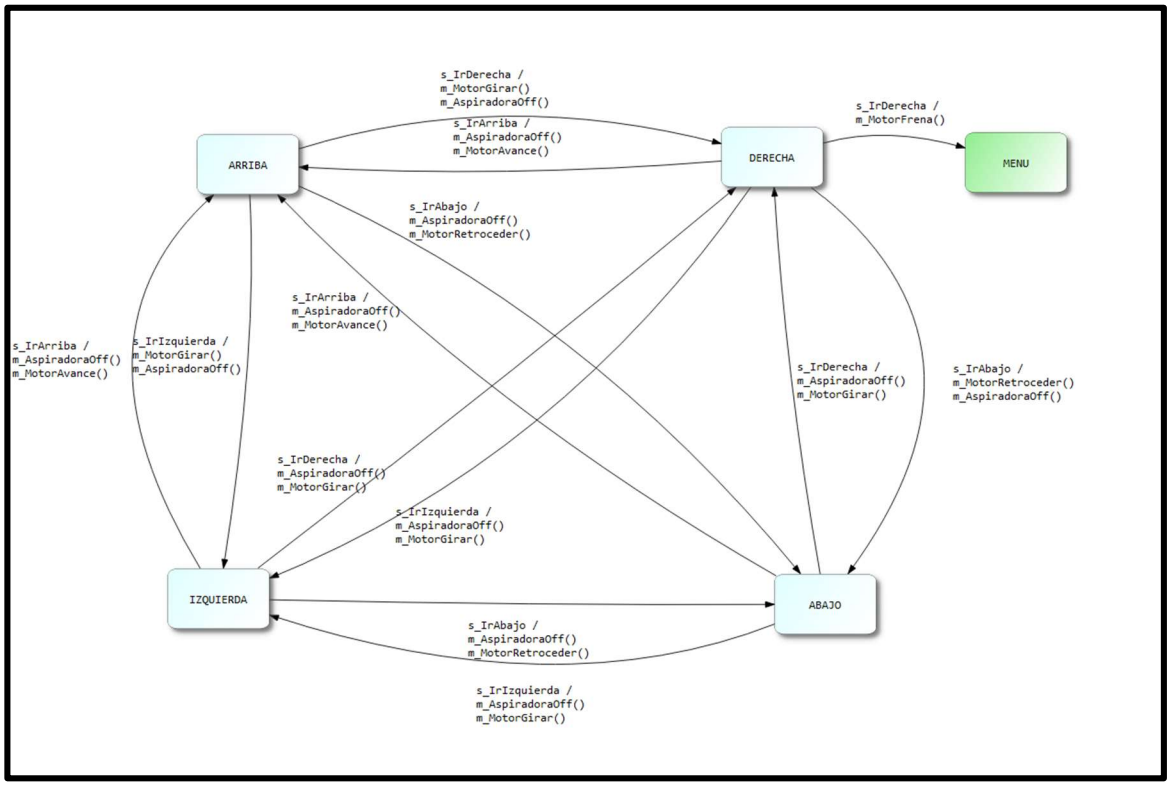


Imagen 23: Estados de interrupción de la aspiradora

La aplicación de PC puede enviar cuatro señales de movimiento a la aspiradora, siendo estas arriba, abajo, izquierda y derecha. Dependiendo de la señal recibida, la máquina de estados irá automáticamente a su estado correspondiente, sin importar en qué momento del programa se encuentre. La única excepción a esta regla es el estado BIENVENIDO. Si la aplicación se encuentra en este estado quiere decir que está iniciando, por lo que no tendría sentido que lea interrupciones externas. Para salir de estos estados especiales, solo debe enviarse la misma señal del estado en el que se encuentra. Cuando finaliza regresa al estado MENU, sin importar lo que estaba haciendo anteriormente.

La última interrupción se da cuando la aplicación pide que la aspiradora inicie una sesión de barrido. Si se envía este mensaje, la aspiradora pasa al estado INICIO_LIMPIEZA automáticamente.

Comunicación (Segunda máquina de estados):

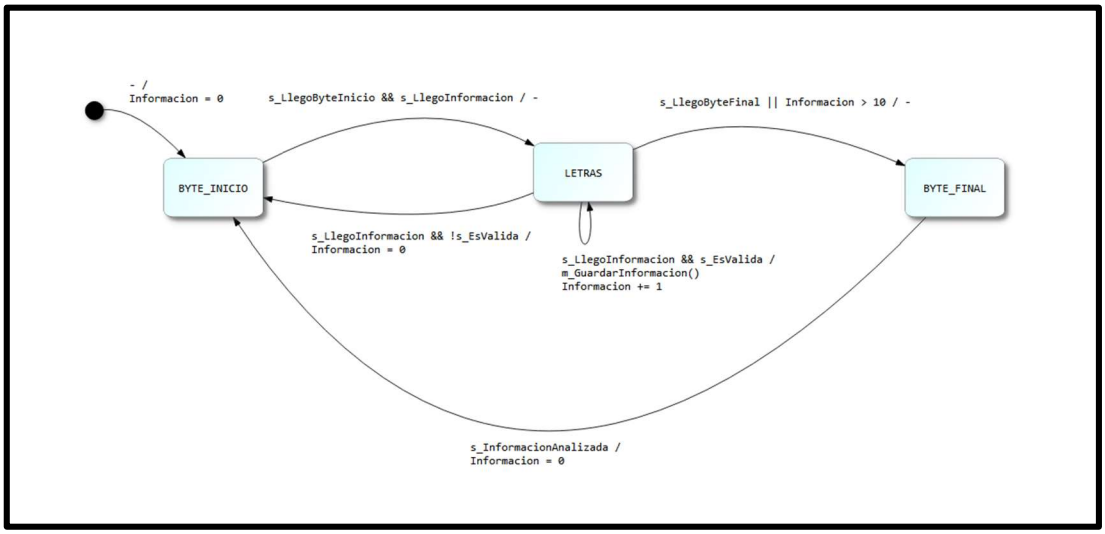


Imagen 24: Máquina de estados de comunicación

La comunicación de nuestro embebido se realiza en paralelo con esta máquina de estados. Su funcionamiento es sencillo y se intentó realizar de la forma más genérica posible de forma que pueda modificarse con sencillez.

La máquina espera a que llegue información en el estado BIT_INICIO. Cuando la información corresponde con el bit de inicio de nuestra trama de comunicación, el estado cambia a LETRAS. En letras se irán guardando en un buffer todos los bytes recibidos. Si la información no es válida (llegan cosas que jamás podrían llegar) se vuelve al estado BIT_INICIO. Si llega el bit final o la trama de información supera el límite, pasa al estado BIT_FINAL. En este último estado, se debe analizar el buffer que se fue generando. Cuando el análisis del mensaje termina, se vuelve al BIT_INICIO y se limpia el buffer.

Problemas encontrados:

Uno de los primeros problemas que surgieron fue la realización del algoritmo de limpieza. Según la investigación, algunas aspiradoras simplemente avanzan hasta chocar con algo logrando cubrir todo el terreno si se lo deja el tiempo suficiente. Esto solucionaba el problema de una forma muy poco eficiente. La otra opción que surgió de la investigación fue la de realizar un mapeo de la zona a barrer. Esto, si bien es altamente eficiente, requería de mucho hardware adicional haciéndolo una opción muy compleja. Terminamos creando un algoritmo propio, que se encontraba en el medio de ambas ideas.

El siguiente problema que nos encontramos fue el uso del sensor ultrasónico. Este requiere detectar el tamaño de pulsos y generar un PWM con un dato extremadamente pequeño (del 1% o menos). Realmente se produjeron muchos cambios en la realización del driver para este sensor, utilizando múltiples periféricos que tenía el LPC845. Además, nuestro contador principal (el systick) se encontraba configurado a 1ms, tiempo

extremadamente grande para los parámetros que maneja el ultrasónico. Parte de la solución se encontró en utilizar un segundo contador, mucho más veloz, dedicado únicamente al ultrasónico.

Otro de los grandes problemas con el que nos enfrentamos está relacionado con la comunicación. Al realizarse por medio de Wi-Fi tuvimos que utilizar microcontroladores externos que tengan esta capacidad (como el ESP01 o el NodeMCU). Tuvimos que aprender a manejar las librerías entregadas por Arduino y realizar algunos programas auxiliares con los que podamos probar los códigos. Para sumar a esto, la comunicación fue uno de los últimos temas vistos en el año, por lo que el tiempo de realización fue relativamente menor a los demás periféricos.

Con respecto al hardware tuvimos un pequeño inconveniente. Los dos motores de 5v, por alguna extraña razón giraban a distintas velocidades. Lo revisamos en distintas condiciones (probar por separado, con alimentación externa, midiendo con tester) y el resultado siempre seguía siendo el mismo. Para corregirlo intentamos reemplazar el más lento de los dos motores por uno nuevo. Fueron recolocados en la maqueta asegurando una posición más paralela entre los motores. El código fue rehecho múltiples veces hasta encontrar la que mejor anduviera. Finalmente logramos un resultado que seguía sin ser perfecto, pero bastante funcional a distancias cortas. El problema se ve muy claramente cuando la aspiradora recorre grandes distancias en línea recta. Al hacer esto, su trayectoria se va desviando levemente debido a la diferencia de velocidades. Consideramos que este problema no afectaba en gran medida nuestro trabajo, puesto que las aspiradoras automáticas con diseños más simples, recorren el piso sin ningún algoritmo o manejo, por lo que, la desviación se puede compensar al dejar a la aspiradora más tiempo funcionando.

Finalmente, el último problema que encontramos fue debido al compilador. En las últimas pruebas del código intentamos agregar algunas líneas y nos empezó a marcar un error que decía “MFLASH64 overflower”. Aparentemente nuestro código llenaba la memoria del microcontrolador. Fue solucionado luego de una investigación en la que descubrimos que el compilador poseía un modo de optimización a la hora de realizar el compilamiento del programa. Al aumentar la optimización, el MCU disminuía la memoria utilizada, pero aumentaba la cantidad de código generado en lenguaje máquina. Como eso no era problema en nuestro caso, se aumentó levemente la optimización con tal de poder seguir trabajando normalmente.

Beneficios encontrados:

El TPO nos pareció una experiencia muy interesante ya que combina todo lo visto en la materia con el objetivo de realizar un dispositivo propio. La creación de drivers con el fin de satisfacer nuestro objetivo es una muy buena práctica para terminar de cerrar dicho tema. Si bien se fue dilatando a medida que pasaban los meses, el tiempo dado nos parece correcto para realizar el trabajo y poder seguir estudiando las demás materias de la carrera. A nuestro entender las entregas parciales pudieron ser mejores, obligándonos a tener que hacer algo de trabajo y marcando un ritmo. También hubiera estado bueno tener un tiempo destinado a simplemente realizar el trabajo durante la clase, de forma tal que, se puedan plasmar las dudas que a veces son difíciles de explicar por el foro o mail. En particular la comunicación, se sintió como algo muy externo a la materia y los temas enseñados, en especial si se quería realizar por Wi-Fi o bluetooth.

El desarrollo de la aplicación de escritorio con QT nos pareció muy entretenido pero tardío, teniendo que aprender e investigar cómo funciona cada uno de los distintos botones y widgets para dejarlo lindo y prolijo. Por eso creemos que es perfecto para el TPO ya que haberlo evaluado en un parcial probablemente lleve muchísimo tiempo.

También nos pareció que hubo temas importantes a desarrollar que se explicaron a fin del segundo cuatrimestre, lo que nos entorpeció el avanzar correctamente (o como nos hubiese gustado). Algunos ejemplos de esto son la utilización de la máquina de estado y la comunicación serie.

Conclusiones:

Tras la realización del TPO pudimos llegar a las siguientes conclusiones:

La programación de un dispositivo se debe realizar de la forma más clara y coherente posible. Simplificar los problemas en varios más pequeños fue un procedimiento muy beneficioso. La realización de drivers y herencias nos permitió ir aumentando la complejidad de estos, realizando tareas sencillas que se volvían cada vez más específicas al proyecto realizado. La realización de la máquina de estados es muy útil al comienzo, ya que nos fija un horizonte hacia el que tenemos que llegar. Si bien su implementación en código no se realizó hasta casi el final, si nos sirvió como una herramienta auxiliar a nuestro trabajo.

Si bien esta materia es Informática II, muchas de las tareas realizadas podrían haber sido completadas utilizando un hardware correspondiente. Esto se vio reflejado principalmente en la realización del ultrasónico, donde la mayoría de los controles deben estar en microsegundos, una velocidad mucho mayor a la recomendable y utilizada por nuestro timer principal. Una combinación entre programación y placas de hardware (como un PWM o filtros para leer el ultrasónico de forma analógica) probablemente hubieran sido más sencillos.

La organización y control de los objetivos fueron primordiales a la hora de realizar el trabajo en tiempo y fecha, así como la creación de código que pueda ser reutilizado en futuras aplicaciones. Cuanto más genérico era el código, más sencillo fue detectar errores y modificarlo para que se comportase como el trabajo necesitaba. Además, muchas veces tenían funcionalidades que no sabíamos que íbamos a utilizar hasta más adelante. Esto también aplica al LPC845 ya que varios periféricos no habían sido vistos en clase y realmente ayudaron a simplificar el trabajo. Leer la hoja de datos y saber las capacidades y limitaciones de nuestros microcontroladores era una ventaja fundamental.

Finalmente, igual que cualquier dispositivo, siempre queda lugar para mejoras. Aunque nos sentimos satisfechos con el resultado final, sabemos que siempre se pueden mejorar las cosas. Probarlo en otras condiciones, agregar más medidas de seguridad, incrementar la compatibilidad de nuestro código son cosas que probablemente aún se puedan seguir haciendo. A medida que más códigos realicemos, mayor será la cantidad de soluciones a estas mejoras que encontremos.

Bibliografía:

Links de investigación utilizados:
https://community.nxp.com/t5/LPCXpresso-IDE-FAQs/Compiler-Optimization/m-p/468819

https://community.nxp.com/t5/LPCXpresso-IDE/AN11538-SCTimer-PWM-cookbook-file-broken/td-p/574911

https://unelectronica.github.io/Led_wifi_esp8266/