**Tecnológico de Costa Rica**

**Ingeniería en Computadores**

**Autores:  
Alejandro Vásquez Oviedo  
Emanuel Marín Gutiérrez**

**Profesor:  
Milton Villegas Lemus**

**Segundo y Tercer Proyecto: Fórmula E CE TEC**

**Grupo 2**

**Fecha de entrega 7 de junio del 2019**

**Introducción**

Los carros de juguete siempre han tenido gran popularidad entre los niños de diversas edades, y esto se destaca sobre manera cuando dichos carros son controlados por control remoto, ya que es notorio el entusiasmo que estos generan en los miembros más jóvenes de los hogares, sin embargo, los controles que ofrecen estos juguetes suelen ser limitados (adelante, atrás y a los lados), además trabajan con radiofrecuencia, lo cual requiere un control adicional que requiere de alimentación independiente, elevando así su costo. Entonces, ¿qué pasaría si se le agregan más funcionalidades a los carros y que estos sean controlados por medio de una red inalámbrica, a través de una computadora portátil? Justamente eso es lo que busca este proyecto, crear, a partir de un carro a control remoto, uno que sea capaz de manejarse a través de una red WiFi, que permita controlar las luces delanteras, traseras, direccionales, esto claramente además de las funciones normales de dirección y tracción. Con el objetivo adicional de que cuenta con comandos especiales que no suelen poseer ningún otro carro disponible en el mercado, entre estos la realización de un círculo hacia la izquierda o hacia la derecha, un movimiento en zigzag, así como la formación de un símbolo de infinito. A estos tres movimientos especiales se le añadió uno adicional cuyo nombre corresponde al nombre de la marca elegida por los desarrolladores para el carro (Katarsys), el cual consiste en la realización de cuatro círculos, uno por cada  
punto cardinal. Además, para facilidad del manejo del usuario se construyó una interfaz en la cual se pueda visualizar el estado del carro, que además posea un registro histórico de los anteriores carros y pilotos de temporadas pasadas de la escudería cuyos datos pueden ordenarse según el REP o el RGP en el caso de los pilotos y eficiencia en el caso de los carros. Además el usuario podrá seleccionar uno de los pilotos de la temporada actual para poder realizar pruebas en una ventana de Test Drive, la cual fue creada con el fin de facilitar el manejo del carro y sus funciones para el usuario. Así como se le dará la posibilidad al usuario de cambiar el logo y los patrocinadores de la Escudería a la que pertenece el carro, incluso podría ingresar un auto nuevo. Esto se logró gracias a la utilización de distintas herramientas que permitieron la construcción del carro, así como su conexión a WiFi y los comandos necesarios para su control. En lo que respecta a la interfaz que utiliza el usuario, fue creada en Tkinter, utilizando las bibliotecas *threading, os, time, tkinter, ttk y winsound*, además de que incluye otra librería llamada *WiFiClient* cuya función es específicamente conectarse con el NodeMCU (el controlador utilizado para conectarse a WiFi), tanto el código para la interfaz donde se digitan los comandos como el que crea la clase que permite conectar el NodeMCU a la interfaz de Tkinter fueron brindados por el asistente del curso, se especifica  además que los códigos fueron manejados en Python 3.8.0. y Python 3.7.1. Otra parte esencial requerida para el funcionamiento del carro fue la programación del código en Arduino, para esto, se utilizó Arduino 1.8.9. Dicho código se compone de distintos módulos que se encargan de procesar el comando y ejecutar la acción según el comando digitado. En lo que respecta a hardware, se ocupó de la utilización de protoboard, jumpers de tipo macho-macho, macho-hembra, y hembra-hembra, estaño, placa preperforada, resistencias, LEDs de distintos colores, cautin, desoldador, cable UTP, alambres, fotoresistencia, transistores, reguladores de voltaje, multímetro, cutter, cuchillas de precisión, silicon caliente, baterias, alicates. Para la parte gráfica del proyecto se utilizó tkinter en su versión disponible en Python 3.7.3 donde se aprovecharon los diferentes widgets tales como pestañas, botones, creación de ventanas, create\_text, create\_image, selección por medio de botones, entry y detección de teclas. Y para el rediseño de las imágenes utilizadas se utilizaron los programas PhotoShop CC 2019, y Paint ver. 10.0.

Se logró una completitud de 95% en el proyecto.

**Conclusiones**

1. La protoboard es una herramienta que resulta en extremo útil para montar circuitos de manera preliminar.
2. El hecho de contar con dos motores es muy práctico ya que permite controlar dirección y tracción de manera independiente.
3. El editor de Arduino 1.8.9 posee una amplia gama de compatibilidad con diversos chips controladores.
4. La versión de Arduino 1.8.9 funcionó de manera correcta a lo largo de toda la elaboración del proyecto.
5. La biblioteca ESP8266WiFi.h es de uso imprescindible en este proyecto ya que es con la cual se conecta y controla el NodeMCU desde Arduino.
6. Instalar una nueva biblioteca en Arduino es un proceso realmente sencillo, a pesar de no tener experiencia previa en ello.
7. Los operadores lógicos permiten un mayor control de los bits en el corrimiento, obteniendo así un resultado esperado de manera práctica.
8. La función *shiftOut* usada en el control del registro de corrimiento que enciende y apaga la luces es en extremo útil pues permite manejar todo un byte con solo dos inputs.
9. El método *switch-case* sugerido por el asistente facilita la legibilidad del código al ahorrar el uso excesivo de *if*s.
10. Al cargar el código .ino al NodeMCU, este no debe recibir alimentación del carro, ya que se genera un error de puerto.
11. Montar todo el circuito electrónico bajo una tierra en común resulta ser muy práctico.
12. Los rectificadores de un motor que redireccionan la corriente completando el ciclo con el negativo general, usando el diagrama del circuito brindado, aplican de igual manera para uno que para el otro.
13. Es posible crear códigos de prueba para cada función en específico, dando certeza así se su funcionamiento o fallo.
14. Debido a que Arduino funciona con una derivación del lenguaje C, es un lenguaje compilado, por tanto no hay forma de verificar error línea por línea sino, únicamente luego de completar el código.
15. Al trabajar con la fotoresistencia utilizar transistor npn en lugar de uno que sea pnp resulta ser más conveniente debido al tipo de circuito.
16. *Str.format* (en Python) no es semejante a *sprintf* (en C) en ningún caso.
17. Al trabajar con valores tan cambiantes como el voltaje se trabajó con el promedio de 10 datos de este para obtener un valor más preciso.
18. Hay fórmulas que permiten obtener el voltaje a partir de las resistencias empleadas.
19. El sensor de luz debe recibir un mayor voltaje a solo 3V, con 7.5V y las resistencias adecuadas funciona de manera correcta.
20. Un buen método para controlar lo que realmente está ocurriendo en el código es realizar *prints* adecuados en el monitor serial.
21. La función *sprintf* es útil para reescribir strings según variables declaradas de una manera más simplificada.
22. Las funciones preconstruidas *analogRead, analogWrite, digitalRead* y *digitalWrite* son en extremo necesarias para el manejo de pines, entradas y salidas, digitales y analógicas.
23. Para evitar errores de conexión es preferible tener los dos archivos .py abiertos.
24. El primer mensaje que se muestra al usuario es siempre un *Timed Out*, comienza a funcionar correctamente a partir del segundo.
25. Cada cierto tiempo ocurre un error de conexión que se soluciona enviando nuevamente el comando.
26. Utilizar la red WiFi del hogar da problemas de conexión e imposibilita el control del automóvil a través de esta.
27. Utilizar una red WiFi creada con el celular demostró ser eficiente a grandes rasgos.
28. El pin ldr y su modificación sobre el byte que controla las luces domina sobre las demás funciones que modifican dicho byte.
29. La fotoresistencia tiene una sensibilidad que depende de la tensión que esté recibiendo.
30. PhotoShop CS6 funcionó de manera ideal como herramienta en la creación de las etiquetas de la marca del carro.
31. El pin D8 (asociado a la fotoresistencia) afecta el funcionamiento general del NodeMCU, al estar operando bloquea el funcionamiento del NodeMCU al reiniciarse.
32. Es conveniente tener el NodeMCU disponible en caso de ser necesario alguna actualización del código.
33. El estaño en su estado líquido es incontrolable lo cual requiere extremo cuidado al momento de realizar las distintas soldaduras.
34. El removedor de soldadura es una herramienta muy útil para retirar el estaño de donde no debería estar.
35. El modo de continuidad del multímetro permite asegurarse de un correcto flujo de corriente a lo largo de todo el circuito.
36. El modo para medir la polaridad de los diodos del multímetro es útil sobre todo cuanto estos tienen borrada la marca que indica donde está el positivo y donde el negativo.
37. En ciertos puntos donde quedó poca cantidad de estaño que no debería estar se removió con las cuchillas de precisión.
38. Para no sobre cargar la placa preperforada se midió la longitud de cable requerida en los puntos específicos y se corto a ese largo para usar únicamente el cable necesario.
39. Las conexiones de los cables de las LEDs del carro resultaron ser de muy mala calidad por lo que se terminaron cambiando casi en su totalidad.
40. Se utilizó el silicón en gran parte del carro y los cables debido a su propiedad adhesiva y aislante.
41. Las diferentes aberturas que se realizaron en el carro fueron hechas con cuchillas de precisión de manera satisfactoria.
42. En el momento de montaje, el  L298 es muy poco seguro y no garantiza el contacto con los cables que se hayan conectado, es necesario asegurarlos de otra forma.
43. Es imposible realizar ángulos de 90 grados con este carro, siempre realizará movimientos curvos o rectos.
44. El NodeMCU no puede recibir más de 3.3 V, si se le envía más se quema.
45. Se cortó de manera satisfactoria la placa preperforada utilizando una cegueta.
46. La placa preperforada no es conductora en sí misma, por esta razón no se calienta al acercar el cautín, más sí se puede quemar.
47. El cautín es capaz de derretir el plástico de una LED separando sus conexiones dejándola inutilizable.
48. La cinta aislante, si bien asegura que no se toquen dos cables no garantiza que se queden en su posición.
49. El número más pequeño que aún envía suficiente potencia para mover los motores es 500.

**Conclusiones III Proyecto**

1.       El formato .gif para imágenes reduce considerablemente la calidad de imagen por tanto no es tan eficiente para trabajar.

2.       Las imágenes png trabajan con mayor calidad y permiten una imagen sin fondo, esto ayuda a lograr una mayor estética.

3.       Python 3.7.3 tiene fallos que evitan la correcta ejecución del código los cuales son enteramente provocados por el lenguaje y no por el desarrollador.

4.       Tkinter en la versión de Python 3.7.3 posee gran cantidad de widgets, muy útiles, que lo hacen una herramienta muy útil en la construcción de interfaces gráficas.

5.       La creación de pestañas permite una interfaz de sencilla navegación y de estilo simple y elegante.

6.       Los labels son eficaces pero poco eficientes, además su manejo resulta más complejo. Y no son compatibles con imágenes en png ya que siempre cargan un fondo.

7.       El método readlines facilitó el manejo de archivos de texto al convertirlo en lista y poder manipularlo en índices.

8.       Al tener una lista del archivo es posible recorrerlo a conveniencia con un índice de la fila y otro de la columna. Muy práctico para seleccionar un fragmento del texto.

9.       El método de ordenamiento por selección aplicado en listas es muy eficaz.

10.   Los datos externos que deban ser usados para alguna función específica conviene tenerlos en lugares de sencillo acceso.

11.   Es posible cargar imágenes tomando el nombre de la imagen de un archivo externo, incluso usando una notación de índices.

12.   Cargar las imágenes y texto tomando la fila como el texto garantiza que se muestren respectivamente en la posición y conjunto correspondiente.

13.   Debido a que el usuario no puede acceder a cambiar el nombre de imagen se puede fijar la posición de lectura en el programa.

14.   El uso de create\_image y create\_text es el método más eficiente y eficaz para mostrar texto e imágenes en la interfaz hasta ahora conocido.

15.   El método de ordenamiento y el de actualizar datos en pantalla son procesos que deben trabajarse de forma independiente

16.   Al crear botones y añadir argumentos a la instrucción command, esta se ejecutará automáticamente.

17.   La función lambda permite crear una función "oculta" de esta forma es posible enviar argumentos a una función de un botón sin que se inicie automáticamente.

18.   Modificar la variable con la lista del documento no es equivalente a modificar el documento mismo.

19.   Abrir un documento en modo "w" borra todos los datos existentes dejando así un documento vacío.

20.   El comando writelines permite escribir todas las líneas de la lista en un archivo abierto en modo "w".

21.   La modificación de datos en el documento no altera de manera instantánea la muestra de estos datos en pantalla.

22.   Utilizar seek para delimitar la reescritura o la muestra de los datos en pantalla es impráctico pues los espacios son constantemente modificados por el usuario al editar la información.

23.   Intercalar los datos entre los que son de letras o numéricos permite delimitar el inicio de un dato y el otro mediante funciones diseñadas concretamente para esto.

24.   Python 3.7.3 puede tomar dos entrys como uno solo y manejar ambos de igual manera, este es un error propiamente de la ejecución del lenguaje Python 3.7.3 ya que al reescribir las líneas no ocurre el error.

25.   Si se realizan dos insert en un mismo entry que se cree dos veces de manera instantánea solo uno de los entry contará con el insert correspondiente.

26.   Es posible modificar las acciones de los botones de la ventana predefinidos tales como el botón de salir, minimizar o agrandar.

27.   Deshabilitar todos los botones de una ventana es posible con un comando y es una excelente medida de seguridad cuando no se desea que el usuario pueda interactuar con todos los botones o ventanas a la vez.

28.   Usar for permite ahorrar mucho recurso computacional incluso siendo este cargar diferentes textos, y además volvió el código más simple y legible.

29.      La biblioteca Tkinter provee módulos y métodos que facilitan el desarrollo de interface gráficas.

30.      El uso masivo de threads ralentizan la ejecución del programa, sobre todo cuando estos se ejecutan en tiempos diferentes.

31.      Al cerrar una ventana que corra o implemente un thread, se genera una excepción en la ejecución del programa, a menos que esta se controle con un try-except.

32.      Debido a la ejecución secuencial de las instrucciones, si dentro de una función se llama a otras funciones, estas se irán ejecutando según el orden en que estén definidas dentro de la función.

33.      Modularizar el código principal en subcódigos no necesariamente implica una disminución de recursos computacionales.

34.      Para captar excepciones de ejecución pass y return son equivalentes.

35.      Trabajar sobre canvas es más eficiente que trabajar sobre labels.

36.      El evento key reléase no es del todo indispensable para el buen manejo del carro tanto físicamente, como en la interfaz.

37.      Como cada comando que se envia al NodeMCU está asociado a una tecla específica, es innecesario hacer comprobación de lectura del comando antes de este enviarse al NodeMCU para ser ejecutado sobre el mismo.

38.  Al conocer la longitud de respuesta de cada uno de los que se le envían al NodeMCU, es fácil distinguir cuál pertenece al comando Sense para la interfaz del nivel de batería, el estado del carro y el tiempo(si es de día o de noche dependiendo del valor de lectura del sensor de luz)

39.  A captar la respuesta del comando Sense es poco probable determinar el rango exacto en que se encuentra el valor que devuelve el battery level y el ldr ya que conforme al usuario manda más comandos para manejar el carro, la logitudes de las respuesta irá aumentando en uno por el indicador el número de respuesta.

40.  Es mejor cambiar los rangos del nivel de batería en el NodeMCU pues tenerlos en un rango de 0.1 voltios entre un nivel y otro, vuelve muy volátil la lectura del mismo afectando la estética de la interfaz del manejo del carro.

41.  Al haber funcionalidades automáticas del carro tales como el encendido y apagado de las luces direccionales siempre que el carro gire hacia uno de los dos sentidos es posible que se afecte la lógica manual del mismo.

42.  El sensor de luz es muy sensible puesto que en el circuito electrónico se le fijaron resistencias con una alta tolerancia.

43.  El tipo de batería influyes mucho en la eficiencia y velocidad del carro.

44.  El pin D8 del NodeMCU hace reset del código cargado cada vez que lee un cero al encender el carro.

45.  Varios eventos pueden ser captados en una misma función mediante el método char para identificar las teclas accionadas por el usuario.

46.  El valor inicial del pwm para mover el carro hacia adelante o hacia atrás debe ser mayor igual a 700 0 -700, con baterías de mala calidad; o de 400 y -400 con baterías de buena calidad.

47.  Para el movimiento hacia la izquierda o hacia la derecha del carro es más sencillo consider un valor de pwm de 1023(máximo) y que si se desea mover el carro en línea recta, es mejor asociar una tecla en específico para tal caso.

48.  Es mejor volver a cargar toda ventana que implique la modificación de datos o elementos de la interfaz por el usuario para poder visualizar inmediatamente los cambios hechos por el mismo.

49.  Es imposible realizar dos acciones simultáneamente, siempre se ejecutará primero una y luego la otra, a veces está acción es imperceptible por el humano pareciendo que tal hecho si es posible.

50.  El efecto de animación inicial requiere de buena capacidad de procesamiento del computador para poder ser ejecutado de manera satisfactoria.

51.  Para elementos modificables por el usuario que no implique la alteración total del txt, ordenar los datos del mismo por el método de espaciamiento es buena opción.

52.  Los Entrys generalmente requieren tener una dimensión mayor a  la del canvas para poder visualizarse en su totalidad.

53.  Los flags son perfectos para condicionar acciones repetitivas.

54.  Definir las variables del programa al inicio genera mayor ordenamiento de memoria por el computador.

55.  Es posible nombrar de manera distinta una función de un módulo aparte del código principal.

**Recomendaciones**

* El kit brindado por el asistente contenía los recursos suficientes para el desarrollo del proyecto en su parte electrónica, y debido a que se adquirió entre dos personas su costo no fue muy elevado, por lo tanto, si se compra entre dos personas es totalmente recomendado, al ser adquirido únicamente por una igual se recomienda debido a que es un kit bastante completo.
* Se recomienda el uso de dos o más protoboard ya que esto permite tener montado el circuito principal en una y dedicar la otra únicamente para pruebas, de solo contarse con una no podría dejarse el circuito principal montado, ya que habría que remover y montar constantemente, consumiendo así también mucho tiempo, debido al exceso de trabajo que requiere el contar con solo una protoboard se recomienda, debido a mayor practicidad, contar en total con dos o tres protoboards.
* Tanto el código dado por el asistente en Arduino como los dos códigos en Python cuya función es crear la conexión entre el NodeMCU y la interfaz de Tkinter, es altamente recomendado ya que funciona de manera óptima, además asegura que no ocurran problemas con la IP al asignar una estándar con la cual siempre va a trabajar.
* Un método en extremo eficaz al tratar con registros de corrimiento donde se busca siempre manipular un único bit es el recomendado por el asistente (operadores lógicos), ya que con estos se garantiza que a pesar del corrimiento que se realice siempre se va a operar el o los bits que se quieran, debido a su alta eficiencia y eficacia se recomienda encarecidamente su uso.
* Debido a que se busca poder controlar de manera independiente las 4 direcciones que pueda tomar el carro, se recomienda trabajar con uno que cuente con dos motores, ya que esto permite un mejor manejo del mismo, acercándose así a lo que se quiere lograr.
* Arduino 1.8.9 posee un código, con base en C, pero mucho más facilitado y de mayor legibilidad para el usuario, además de poseer gran cantidad de controladores a disposición del usuario, así como muchos ejemplos acerca de lo que se puede hacer, debido a su facilidad de manejo y gran practicidad se recomienda.
* Arduino 1.8.9 al ser un lenguaje compilado no permite la revisión de errores analizando de manera independiente un fragmento del código, únicamente es posible detectar los errores, ya sean de sintaxis o mala programación, una vez compilado y ejecutado el código, si bien esto resulta ser más eficiente para el desarrollador puede generar conflictos si no se tiene suficiente conocimiento o no se es suficientemente cuidadoso, se recomienda el uso de Arduino, siempre que el desarrollador tenga presentes estos dos aspectos.
* Ya que se buscaba controlar por medio de una red WiFi se decidió trabajar con el NodeMCU se requiere una biblioteca que trabaje con esto de manera adecuada, luego de probar su uso y que funcionara de manera correcta se recomienda la librería ESP8266WiFi.h.
* Al no disponer de suficientes pines que actuén sobre todo lo que se quiere controlar la función *shiftOut* así como el registro de corrimiento ofrecen una excelente alternativa pues permite controlar 8 salidas con un solo pin, se recomienda su uso.
* Utilizar la sintaxis *switch-case* si bien es tan eficiente como una serie de *if*s vuelve mucho más legible el código y lo hace ver mucho más simple y elegante a la vista de quien lo revise, por esta razón se recomienda.
* Se intentó primeramente con el transistor pnp sugerido en el diagrama del circuito, sin embargo, gracias a los conocimientos previos en electrónica del estudiante Emanuel Marín se decidió utilizar en su lugar uno npn, el cual debido a su composición trabaja con el voltaje de una manera más acorde con la que deseada en este proyecto.
* La función *sprintf* es muy útil cuando se quiere reemplazar un string por otro que tome los valores a escribir de una variable o función en específico, solo debe asegurarse que el string donde se va a escribir posea suficiente espacio, debido a su fácil comprensión y practicidad se recomienda su uso.
* Se recomienda utilizar valores promediados en los casos donde el dato que se desee utilizar sea poco estable o muy inconstante, esto para reducir así el margen de error en los cálculos o muestras que requieran de este valor.
* Las fórmulas encontradas en la página **microcontroller-project** resultan sencillas de entender, aplicar y muy precisas en cuanto al cálculo del voltaje y las resistencias según se requiera, debido a su alta efectividad se recomienda su uso.
* El comportamiento de la fotoresistencia varía dependiendo de la cantidad de voltaje que esta recibe, y por esta razón el nivel de voltaje y corriente que libera es distinto de esta manera su comportamiento varía, por esta razón se recomienda mantener un control sobre el nivel de voltaje que recibe para así regularlo de una manera más controlada, o en caso contrario, conocer este comportamiento y asegurar que no ocurra ningún error debido al mismo.
* Como herramienta de gran utilidad en el manejo y control de las distintas salidas y entradas con las que se trabaja se recomienda fervientemente el uso de las funciones preconstruidas de *analogRead, analogWrite, digitalRead* y *digitalWrite,* debido a esto y además de su fácil legibilidad se recomienda el uso de los mismos.
* Las conexiones creadas con dispositivos móviles suelen tener cierto problema de estabilidad, debido a esto se recomienda que dichos dispositivos estén cerca de la computadora empleada como control y que además se encuentren con la pantalla encendida.
* Se recomienda el manejo y creación de imágenes por medio del programa PhotoShop CS6 ya que su interfaz es de manejo sencillo y permite la manipulación de imágenes sin mayor dificultad.
* Se recomienda acondicionar un espacio del carro donde se coloque el NodeMCU y que quede de fácil acceso al usuario o desarrollador en caso de realizarse alguna clase de actualización, si bien requiere más tiempo y trabajo es necesario tener acceso al NodeMCU, incluso en caso de ser necesario arreglar algún error en el código, además se recomienda que dicho espacio se condicione de forma que no arruine la estética del carro, aunque esto lleva más tiempo evita que el impacto visual del carro se vea afectado.
* Se recomienda posicionar la placa preperforada de manera que el estaño líquido no se mueva hacia un punto que afecte todo el comportamiento del  circuito, generando así problemas que afectan en su totalidad la finalidad del circuito en construcción.
* Se recomienda encarecidamente disponer de un removedor de soldadura pues el estaño es de un manejo bastante complejo y tiende a alcanzar lugares no deseados, por esta razón es preferible contar con una herramienta que facilite su remoción.
* Se recomienda el uso del multímetro como herramienta de uso esencial en la comprobación de flujos de corriente, continuidad del estaño y polaridades en general, así como en la cantidad de voltaje que se recibe, suelen ser de un costo mayor al de los demás elementos utilizados, sin embargo facilita mucho la construcción de todo el circuito así como también da certeza sobre tu correcto funcionamiento y permite la fácil detección de errores a nivel de hardware y conexiones.
* Para un mejor manejo de la placa preperforada se recomienda trabajar con la longitud estrictamente necesaria de los cables, al hacerse de esta manera se reduce el exceso de cables por lo que se crea un circuito que es más sencillo de entender, seguir y en último caso replicar.
* Debido a que existen conexiones que vienen de fábrica se recomienda revisar la estabilidad de las mismas y asegurar así que no se vayan a romper con facilidad, en caso de verificar que se tengan malas conexiones de fábrica se recomienda cambiarlas en su totalidad y así crear conexiones más seguras que eviten a largo plazo problemas en algún funcionamiento del carro, ya que de no hacerse esto el costo en tiempo y trabajo aumentaría de manera no deseada.
* En caso de requerir adherir cables expuestos a alguna superficie y asegurarse que no ocurra un mal contacto se recomienda con total seguridad el uso del silicón caliente, esto debido a la propiedad aislante que posee el silicón y debe ser caliente debido a su rápido secado, el silicón frío no se recomienda ya que toma demasiado tiempo secándose, tiempo que podría aprovecharse en otra tarea y agilizar el proceso de desarrollo.
* Las modificaciones realizadas a la carrocería del carro fueron hechas con cuchillas de precisión, hacer las modificaciones con estas herramientas resulta bastante tedioso debido a el grosor del plástico, a pesar de que toma mucho tiempo los resultados, si se saben usar, son bastante estéticos y cumple de manera esperada sus resultados, por esta razón, a pesar de consumir mucho tiempo se recomienda su uso pues se consiguen modificaciones que no afecten de manera negativa la estética del carro.
* Se recomienda encarecidamente utilizar algún pegamento potente que asegure la conexión correcta de los distintos cables del L298 ya que de no asegurarlos pueden no desconectarse y deshabilitar la acción de distintas funciones del carro.
* Debido a las altas temperaturas del cautín se recomienda no acercarlo de manera excesiva a la placa preperforada pues llega a quemarla, y si se quema a un punto donde no sea posible crear el circuito deberá comprarse otra por lo que los gastor incrementarían de una forma no deseada y evitable, de igual manera se debe tener esta precaución al acercar el cautín a los demás componentes.
* En caso de quererse aislar algún cable o alambre se recomienda el uso de la cinta aislante, pues esta garantiza que no se genere ningún tipo de contacto no deseado, sin embargo no es útil para fijar componentes en algún lugar en concreto, en caso de querer fijar algo se recomienda algún pegamento de mayor potencia, pues la cinta aislante no posee la capacidad de sostener.

**Recomendaciones III Proyecto:**

* Se recomienda evitar a toda costa el trabajar con imágenes en formato gif, esto debido a que empeora la calidad de la imagen, es un formato que posee menor compatibilidad que otros como el jpg o png, y al siempre acarrear un fondo evita que sea posible un mejor diseño, debido a poseer tantos aspectos negativos no se recomienda su uso.
* El uso de imágenes en formato png conserva la calidad de la imagen original, además de no contar con un fondo lo cual permite una mayor estética en el diseño de la interfaz, por estas razones se recomienda encarecidamente su uso, además de ser uno de los formatos permitidos por tkinter.
* Python 3.7.3 es un lenguaje de alto nivel que posee una sintaxis sencilla por lo que programar en él no resulta tan complicado como en otros lenguajes, sin embargo, posee errores que se deben únicamente al lenguaje como tal. Demostró ser un lenguaje que cumple con las necesidades para este proyecto, por tanto se recomienda su uso pero se advierte de que pueden ocurrir errores que se solucionan al reiniciar el editor.
* Debido a que el proyecto consistía en desarrollar una interfaz gráfica se recomienda utilizar la biblioteca de tkinter en la versión de Python 3.7.3, pues a lo largo del proyecto demostró contar con gran cantidad de herramientas muy potentes que permiten una mayor versatilidad en la creación de interfaces, al ser una herramienta tan potente se recomienda altamente su uso.
* El uso de pestañas permite ahorrar toda la creación de una nueva ventana y de los botones para poder entrar y salir de ella, además que vuelve la interfaz más manejable para el usuario, siendo más práctica para el usuario, y una excelente forma de ahorrar recurso computacional se recomienda su uso.
* El uso de labels puede resultar eficaz, pero no es eficiente debido a la cantidad de líneas que requiere, además de limitar la edición al cargar siempre un fondo que puede arruinar la estética o efecto deseado, por estas razones no se recomienda su uso en la creación de interfaces.
* Al trabajar con archivos de texto siempre resulta más sencillo realizar la lectura y cambio con el documento en tipo lista, para esto se utiliza *readlines*, que además permite recorrer el archivo por medio de índices y subíndices, por tanto, al facilitar el manejo de los datos se recomienda encarecidamente su uso.
* El método de ordenamiento por selección, al comparar un elemento menor con otro manejando índices, es de sencilla implementación a cualquier tipo de lista, por esta razón se recomienda su uso siempre que se quieran ordenar listas.
* Al requerir acceder a un dato específico de un archivo de texto, ya sea para cargar imágenes u algún ordenamiento con base en este se recomienda colocar dichos datos en alguno de los extremos de cada fila de modo que sea sencillo acceder a ellos.
* Al requerir mostrar imágenes ubicadas en posiciones específicas en la interfaz, que además deben cambiar según se modifique el ordenamiento, se recomienda no cargar directamente el nombre de la imagen, en lugar de esto, se recomienda fuertemente cargar la fila con el texto correspondiente al nombre de la imagen, y así al cambiar el nombre, la fila sigue siendo la misma y se muestra la nueva imagen en la fila correspondiente, por esta razón se recomienda este método de carga.
* Ya que create\_image y create\_text permiten la creación de estos elementos en tan solo una línea de código, además de crearlos sin ningún fondo que afecte estéticamente, se recomienda encarecidamente su uso por sobre otros métodos menos eficientes para mostrar texto e imágenes en la GUI.
* Para actualizar los datos el método de eliminar y crear la ventana resulta ser eficaz, mas no del todo eficiente, ya que no es un método sutil ni elegante, por esta razón no se recomienda su uso. Es preferible encontrar otro método que realice esta acción.
* Si se desean enviar argumentos en una función asociada a un botón se debe utilizar lambda, ya que de otro modo la función se ejecutaría de manera inmediata, sin siquiera presionar el botón, al ser este un comportamiento no deseado se recomienda utilizar el método lambda.
* Al reescribir los elementos de la lista del documento y querer modificar el documento como tal se recomienda abrir el archivo en modo “w” y utilizar *writelines*, pues es un método altamente eficaz, que no requiere de mayores recursos computacionales.
* Al dar acceso al usuario que edite los datos del archivo se debe tomar en cuenta que los espacios pueden variar, cambiando así la cantidad de caracteres designados originalmente, por esta razón utilizar seek para delimitar donde iniciará la escritura resulta en extremo impráctico ya que siempre será un número distinto, por esta razón no se recomienda su uso.
* Mientras los datos se encuentren acomodados de manera que sea uno compuesto de letras seguido de uno numérico se recomienda el uso de las funciones creadas para este propósito pues resultan ser totalmente eficientes y eficaces para estos casos.
* Se recomienda el uso de insert si se desea que un entry se muestre con un texto previo, pues cumple en gran medida el comportamiento esperado, mas cabe destacar que pueden ocurrir errores con este método donde el insert se aplique de manera equívoca, por tanto, se recomienda utilizar con la debida precaución.
* Debido a que mostrar dos ventanas relacionadas a la vez puede provocar problemas de funcionamiento o comportamientos no deseados, se recomienda deshabilitar todos los botones y posibles interacciones de la ventana según convenga para así evitar cualquier tipo de fallo inesperado.
* A pesar de que la recursividad es eficaz, suele consumir, en gran cantidad de ocasiones, una considerable cantidad de recursos computacionales, los cuales se pueden ahorrar al utilizar algún método iterativo, por esta razón se recomienda utilizar iteración.
* Cuando se requiere que el usuario interactúe directamente con la interfaz con mecanismo tales como la selección, ya sea de una imagen, un piloto, una canción, entre muchas otras más; usar radio button simplifica dicha lógica pues en lugar de tener que asociar botones para cada opción seleccionable, se asocia el click del mouse.
* Como hay funciones que constantemente necesitan estar ejecutándose para comprobar el nivel de batería, el estado del carro, si es de día o de noche; usar Threads es la mejor opción ya que mediante está herramienta se puede establecer la cantidad de tiempo entre una comprobación y la otra y así evitar colisiones.
* Si se desea modularizar partes del código principal, se recomienda hacerlo para líneas que se usen de vez en cuando y sean indispensable para la lógica general del programa. De esta forma lo que se modularice podrá ejecutarse más rápido, y así devolver valores más exactos y en menor tiempo.
* Al haber funciones que corren mediante un Thread, es mejor implementar try-except para captar posibles excepciones en la ejecución del programa y este pueda seguir ejecutándose sin ningún problema.
* Debido a que desarrollar una interfaz gráfica implica en mayor medida la implementación de efectos visuales, usar canvas en lugar de labels permite poder introducir imágenes y letras sin tener que preocuparse por aspectos tales como la transparencia.
* Darle proriedad a la lógica de la interfaz en lugar de la estética. Es mejor comprobar que la lógica de especificación del proyecto funciona, en lugar de usar tiempo que quizás luego se ocupe, en la estética del producto final.
* Trabajar con flags en lugar del evento keyrelease. Para las distintas acciones del carro desde la interfaz del drive mediante teclas los flags acortan el código y lo vuelven más sencillo de entender que la implementación del keyrelease.
* Ya que cada comando que es enviado al NodeMCU para ser ejecutado por el mismo se asocia a teclas específicas, es mejor no hacer un código de comprobación del comando enviado, en lugar de eso, es mejor asegurarse que cada comando se adapte a lo que pueda leer e interpretar el NodeMCU.
* Conocer la longitud de cada comando enviado al NodeMCU permite saber si se trata o no de la respuesta del comando sense para la lógica del nivel de batería, estado del carro y tiempo (día o noche) del mismo; de lo contrario es más dificil distinguir entre la cantidad de comandos que se le envían al carro, cuál pertenece al comando sense.
* Recorrer la respuesta del comando sense permite tener el valor exacto del battery level y el sensor de luz, de otra forma, la lectura del mismo será inestable y podrá causar excepciones en la ejecución del programa.
* Cambiar los rangos de los niveles de la batería en el archivo .ino permite tener valores más estables en la interfaz mientras se ejecuta el Drive Car. Sin esto, es posible que el usuario vea cambios en el porcentaje de la batería muy inestables y pensará que el carro está dañado.
* Definir un valor inicial para los distintos flags para el manejo de las luces y el movimiento del carro tanto de forma automática como manual afecta menos el manejo del mismo tanto físicamente como en la interfaz.
* Cuando por motivos de mantenimiento se requiera volver abrir el carro, se recomienda cambiar el valor de las resistencia en el circuito del sensor de luz ya que por el momento es muy sensible.
* Comprar baterías recargables o de buena calidad perimiten tener un mayor rendimiento en el carro, de lo contrario el carro trabajará de manera menos eficiente.
* Para que el código que contiene el NodeMCU no se resetee es mejor poner el sensor de luz del carro en cualquier pin menos el pin D8 del NodeMCU puesto que este al encederse el carro si lee un valor de lectura en alto hace reset del código cargado.
* Para reducir la lógica del encendido y apagado de las luces del carro, el método char permite reducir de sobre manera dicho código, de otra forma abría que crea una función independiente para cada evento relacionado con las luces del carro.
* Si las baterías del carro son de mala calidad es mejor configurar un valor inicial del pwm de 700 o -700 según sea el caso para que carro funciona de manera eficiente. Si las baterías que usa el carro son recargables o de buena calidad, se recomienda poner el pwm del mismo en 400 o -400 según corresponda.
* Para toda ventana que implique modificación de algún elemento de la interfaz es mejor siempre actualizar dicha ventana, de lo contrario habrán cambios que el usuario no podrá visualizar inmediatamente.
* Para acciones repetitivas se recomienda usar un tiempo adecuado entre una acción y la que le sigue, así la comunicación con el carro no se ve afectada.
* Si los cambios realizados a un txt por el usuario son mínimos usar el método de espaciamiento hará más sencillo el código, de otra forma habrá que implementar una función adicional que valide el txt.

**Análisis de resultados**

Desde el inicio del proyecto se contó con tres protoboards, lo cual permitió mantener el circuito principal en una, y trabajar los distintos circuitos de prueba que se fueron requiriendo a lo largo del proyecto, en las otras dos, para así evitar tener que quitar y rearmar cada vez que se requiriera montar un circuito nuevo. Además, debido a que se necesita poder manipular de manera independiente la dirección, se buscó un carro que contase con dos motores que no dependieran el uno del otro, esto fue una enorme facilitación al momento de cumplir con la especificación solicitada.

La experiencia previa con Arduino era mínima, por esta razón fue de gran alivio el que dicha plataforma contara con una amplia cantidad de controladores y además que aunque no contara con el NodeMCU de manera preinstalada, su instalación no requiriera de mayor conocimiento del propio Arduino sino que fuese bastante sencillo de instalar, a lo largo de todo el proyecto no ocurrió ningún problema que se le pudiese adjudicar a la versión de Arduino utilizada, la 1.8.9.

Nunca antes se había introducido el manejo de operadores lógicos dentro de algún código, y luego de estudiar más a fondo su comportamiento y aplicarlo al registro de corrimiento, se entendió que era la mejor forma para asegurar que únicamente se modificara un bit exclusivamente a pesar de que el corrimiento afectase a todos los ocho bits. Al utilizar los operadores lógicos se usó en conjunto la función preconstruida *shiftOut*, que luego de verificar su comportamiento (siendo este el desplazar los bits uno a uno según indicara el programador), completaría de manera satisfactoria el código que controla el comportamiento de las LEDs del carro.

Al principio el comportamiento de las keywords switch y case se mostraba algo complejo, luego de una ligera investigación se encuentra que ambas son excelentes herramientas que evitan sobre cargar el código utilizando una mayor cantidad de ifs y else, que en el fondo cumplen la misma función, salvo que switch y case simplifican el código.

A lo largo de las diversas pruebas realizadas en el funcionamiento del NodeMCU y muchos errores de conexión que ocurrían a lo largo de estas pruebas se verifica que el método que garantice su funcionamiento posterior a la carga del código es si se tiene el NodeMCU conectado tanto a la computadora como a la fuente de energía del carro y luego retirar el cable USB, de otra manera podría dar pie a que ocurran problemas.

Al momento de construir el circuito encargado de controlar uno de los motores y conectarse a este, a pesar de recibir el voltaje adecuado no se producía ninguna reacción, luego de muchas pruebas y reconexiones se determina que el problema se encuentra en la ausencia de la rectificación debido a que no estaba conectado a una tierra general, y de esta manera no se completaba el circuito por lo tanto nunca iba a funcionar. Dicha rectificación está realmente conectada al otro motor, sin embargo, al estar los dos interconectados la misma rectificación actúa de igual manera para ambos motores, asegurando así su operamiento adecuado.

Debido a que se debía probar distintas funciones para el carro, como equipo se optó por fragmentar las distintas funciones en códigos y circuitos particulares, de esta manera era posible analizar con mayor detalle el funcionamiento de las distintas tareas que se debían realizar, y luego unir todo en el código y circuitos definitivos.

El trabajar con programas compilados resulta en un trabajo que requiere mayor cuidado pues únicamente muestra errores una vez que ya se recorrió todo el código, a diferencia de los intérpretes que al primer error detienen su ejecución, por esta razón es más difícil para el desarrollador saber dónde se equivocó.

El transistor sugerido fue un pnp 3906, mas debido a su estructura y forma de trabajar con el voltaje se decidió que era mejor trabajar con un npn, específicamente el 2N2222A, pues se adecuaba mejor a las necesidades del proyecto, al emplearse se comportó de la manera esperada.

En uno de los comentarios dejados por el asistente del curso hacía una comparación entre el *Str.format* de Python y el *sprintf* de Arduino, al investigar sobre la función de *Str.format* generó cierta desconfianza debido a que elimina el string que se le asigne, y no se busca eso, al investigar sobre *sprintf* se nota que su función es reescribir un string sobre otro, lo cual parece una opción viable, por esta razón se deja y actuó de manera esperada.

Al investigar sobre posibles soluciones que permitieran determinar el voltaje de la batería y así sacar un porcentaje se encontró un método que sugería utilizar un promedio de 10 detecciones de voltaje que permitieran así trabajar con un valor más certero.

Al iniciar las pruebas de la fotoresistencia ocurrían problemas de funcionamiento, y se revisó el voltaje, estaba recibiendo un total de 3.3V aproximadamente, se decidió cambiar a 7.5V y tomando las previsiones necesarias, como el cambio de resistencias, operó de manera correcta según lo esperado.

Al ser un lenguaje compilado se tuvieron que idear métodos que comprobaran el funcionamiento, activación del código y modificación correspondiente de variables, esto se realizó por medio de diversos prints en el monitor serial, estos permitían un mayor control sobre la operación real del código en momentos determinados.

Al estar iniciado con la creación de códigos en Arduino y ver distintos ejemplos que utilizaban las funciones *analogRead, analogWrite, digitalRead* y *digitalWrite* se decidió replicar estas funciones según se fueran requiriendo en la generación del código controlador, al final se determina que son en extremo útiles para manejar los diversos pines que tienen asignada una función.

El lograr una conexión satisfactoria fue trabajo de varias horas, se logró luego de distintas pruebas, donde muchas culminaron en distintos errores, se logró al tener todos los documentos de código abiertos, además de concluir que una red creada con el celular no da problemas como sí los da una red WiFi local, de igual manera se detectaron errores de red perdida o desconectada que se debían a la lejanía del celular, o que este tuviera la pantalla apagada, por lo que deben ser consideraciones que se deben tomar siempre en cuenta para un uso adecuado.

A la hora de asignar una modificación en el byte controlador de las luces al pin ldr ocurren comportamientos no esperados que resultan ser debidos a un problema de dominancia donde el ldr siempre se superpone antes los comandos, por esta razón las luces se comportan de manera atípica. Dicho pin ldr está asignado a la fotoresistencia, cuyo comportamiento de sensibilidad varía según el voltaje que esté recibiendo. Además, una vez  conectada la fotoresistencia al pin del NodeMCU comenzará a afectar su funcionamiento de una manera que puede resultar poco ventajosa ya que al estar operando bloqueará la funcionalidad total del NodeMCU.

Debido a que el integrante Alejandro Vásquez cuenta con experiencia previa en el manejo de PhotoShop CS6 y al no tratarse de un trabajo muy elaborado se decide utilizar este programa en la creación de las placas del carro.

Una vez se estuvo más avanzado en el desarrollo del proyecto se llega a la conclusión de que resultaría conveniente tener un fácil acceso al NodeMCU en caso de requerirse modificaciones en el código, por lo cual, usando cuchillas de precisión, se hizo una abertura en la ventana trasera y que esta se comportara como compuerta para tener un sencillo acceso al NodeMCU.

Al trabajar con el estaño y el cautín, siendo la única forma de manipular dicho elemento en su estado líquido, se torna difícil su control, esto obliga a colocar la placa preperforada en una posición donde la gravedad no mueva el estaño a lugares indeseados, aun así suelen ocurren errores donde el estaño cubre espacios que no debería, gracias a que se cuenta con un removedor de soldadura se facilitó el retirar el estaño indeseado, sin embargo esto se logró con cierto nivel de dificultad pues lograr que el estaño regresase a su estado líquido no fue tan sencillo, sin embargo, fue posible.

Debido a la inestabilidad y poco control que se puede tener del estaño se volvió necesario verificar que únicamente hubiera conexión entre los puntos deseados, para asegurar que esto se hubiera realizado de manera esperada se aprovechó la herramiento del multímetro que se encarga de indicar continuidad, fueron varios los casos donde había continuidad no deseada y se procedió a eliminarlos por medio de el retirador de soldadura y las cuchillas de precisión cuando la cantidad de estaño era muy pequeña.

Los diodos poseen una ligera marca que ayuda a determinar su polaridad lo cual delimita la forma en la que se deben conectar, sin embargo, los diodos usados tenían dicha marca muy borrosa, por esta razón se utilizó la función del multímetro que indica la polaridad de los diodos para asegurar su correcta colocación, esto se realizó colocando las patas del multímetro a ambos lados de cada diodo y si coincide positivo con positivo y negativo con negativo muestra un número distinto de cero, caso contrario muestra cero, y así se asegura qué extremo del diodo es negativa o positiva.

La protoboard con el circuito completo se veía demasiado saturada, entonces, se siguió la recomendación del compañero Emanuel Marín de utilizar únicamente la longitud necesaria del cable en la placapreperforada y de esta manera no saturarla.

A la hora de unir los cables de las LEDs con el circuito se produjeron errores debido a que los cables de las LEDs se soltaban de manera inmediata, por esto se tuvieron que cambiar las conexiones de tres LEDs que fueron las que se dañaron, se crearon nuevas conexiones con cables más fuerte y una mayor unión de estaño. de igual manera era necesario que estuvieran adheridas a la carrocería del carro pero sin tocarse entre ellos (el negativo y positivo de cada LED) por eso se decidió usar silicón caliente para pegarlas al carro y aislarlas entre ellas, aunque en ciertas ocasiones el silicón se despegó no hubo mayor problema.

Como se recomendó el uso de cuchillas de precisión se usaron para realizar cortes en la carrocería, a pesar de tomaron mucho tiempo los distintos cortes realizados los resultados fueron bastante adecuados tanto en funcionalidad como en estética, a pesar del gran trabajo que requirió.

Se probaron los distintos comandos del carro, y al primer intento no funcionó, lo cual dejó un enorme desconcierto, se revisaron las conexiones y se notó que estaban flojas en el L298, este problema ocurrió en dos ocasiones, la primera solución dada fue asegurar con cinta aislante, no funcionó, posteriormente se adhirió pegamento para asegurar las conexiones.

Al pensar el comando especial se planteó programas hacer un cuadrado, sin embargo el carro no puede crear ángulos de 90 grados, eliminando así gran cantidad de posibilidades de figuras para programar.

El NodeMCU es muy sensible si recibe un voltaje mayor a 3.3V ya que se quema en este caso, por tanto se mantuvo mucho cuidado acerca del voltaje que iba a recibir, con este cuidado nunca hubo problemas de cortocircuito o sobrecarga en el mismo.

La placa preperforada se corta perfectamente con una cegueta además que no es conductora, lo cual garantiza que no se afecte por calor ninguno de los componentes ya soldados a esta.

Se hicieron diversas pruebas para determinar el mínimo requerido de potencia para el funcionamiento de los motores, de partió de 850 pwm, como aún funcionaba se comenzó a disminuir gradualmente hasta alcanzar un pwm de 500 en el cual ya no se producían acciones, solo el sonido del motor.

**Análisis de resultados III Proyecto:**

Al intentar pasar las imágenes de .gif a .png para poder utilizarlas sin fondo se produjo una reducción de la calidad que las dejó inutilizables lo que forzó a buscarlas nuevamente para así poder convertir directamente a .png sin perder calidad y trabajar de una manera más estética.

La versión de Python 3.7.3 resultó ser eficiente, aunque ocurrieron varios problemas cuya causa fue únicamente un error del propio Python 3.7.3 ya que al reiniciar no volvió a ocurrir el error, y esto consumió tiempo que pudo ser empleado en otras tereas pendientes.

En lugar de crear toda una ventana distinta que mostrara los datos de los autos se decidió crear una nueva pestaña dentro de la misma ventana luego de investigar distintos widgets de Tkinter, lo cuál simplificó en gran medida el código al montar todos los datos en una sola ventana, además de permitir una navegación sencilla entre las dos tablas de datos.

La creación de labels requiere una mayor cantidad de líneas de código, además de cargar el texto con un fondo lo cuál arruina cualquier intento de estética que se intente, resultan ser en extremo imprácticos cuando se busca una interfaz agradable a la vista, además del mayor recurso computaciones y espacio en el código que ocupa a diferencias de sus contrapartes create\_image y create\_text.

El método de ordenamiento por inserción resulta ser altamente eficaz en ordenamiento de los elementos de una lista, incluso si los datos con los cuales se quiere realizar el acomodo se encuentran en el medio de algún elemento de la lista, además de que es un código legible y de fácil adaptación y uso.

Aunque al inicio fue difícil idear la lógica que permitiera cargar las imágenes de cada piloto según sea el caso, el tener los nombres de las imágenes dentro del mismo archivo txt del que se toman todos los demás datos resultó ser de gran utilidad y permitió facilitar el ordenamiento de los demás datos. Por otra parte, colocar el nombre de la imagen al inicio de cada fila dentro del archivo permite y manejo más sencillo a la hora de indicar qué debe tomar como imagen a cargar y qué datos debe mostrar como texto. Ya que el usuario no tiene acceso se puede trabajar con una cantidad de datos específica y esto simplifica a gran medida la complejidad del código.

Las herramientas create\_text y creat\_image, a pesar de necesitar una mayor cantidad de argumentos requieren únicamente una línea de código para poder introducir imágenes o texto en pantalla, que a pesar de no ser legible para cualquier usuario es de alta eficiencia computacional y estética.

A pesar de que se modifique el orden de los datos en el archivo txt no se muestra el cambio en la ventana de manera directa ya que la ventana se cargó con los datos previos a su reacomodo y al ya haber pasado por la línea que cargaba la información el programa de manera instantánea no va a actualizar la información, por esto se buscaron distintos métodos que mostraran la información modificada, se probó con distintos métodos que no dieron un resultado esperado, en la mayoría de los casos no funcionaron del todo, por tanto se decidió cerrar la ventana y volver a cargar de manera automática.

Al crear los botones para editar la información de los autos y los pilotos se buscaba crear una función para todos ya que en caso de hacer una para cada uno serían demasiadas funciones y al intentar enviar la función con argumentos desde el *command* del botón de esta manera la función se ejecutaba de manera automática, usando lambda se logró que solo se ejecutara al presionar el botón y enviando los argumentos respectivos. De esta manera se construyó una función capaz de leer cada fila y colocarla dentro de un Entry.

Al intentar guardar el cambio generado se daba el problema de que a la hora de modificar un dato este se modificaba en la lista generada a partir del documento, pero no en el documento mismo, por tanto. se debió crear una función aparte en la cual se reabriera el archivo en un formato “w” para poder escribir únicamente y en esta modalidad se escribía toda la nueva lista que poseía el cambio efectuado por el usuario. De esta manera se consiguió guardar el cambio, más no mostrarlo debido a que la ventana seguía desactualizada por lo que se repitió el método de errar y abrir nuevamente la ventana con la información de las posiciones.

Ya que no se quería depender de los espacios debido que el usuario tiene acceso ellos y puede modificarlos se buscó el método que permitiera siempre encontrar los diferentes elementos, se hizo por medio de funiones que compararan cada elemento del string con los números y de esta manera se delimitaba hasta donde llegaba un dato y empezaba el otro aprovechando que se podían colocar los datos en el orden que fuera más conveniente y se colocaron de manera intercalada entre datos de palabras o de números, las funciones resultaron ser altamente útiles en estos casos pues permiten prescindir de la cantidad de espacios. A pesar de que debieron crearse funciones diferentes para los autos y para los pilotos debido a la diferencia en la cantidad de datos que poseen.

Una vez que se logró determinar con precisión los distintos tipos de datos se empezó a trabajar en mostrarlos en distintos entrys para una mejor comprensión de los datos para el usuario. Tomó más tiempo del esperado, esto debido a que los entrys se configuraban de manera errónea, e intentar solucionar este error tomó mucho más tiempo del esperado, al final se solucionó reescribiendo las líneas de código que creaban los entrys, adjudicando así este error a un problema propiamente de Python 3.7.3.

Al descubrir un bug que perjudicaba por completo el funcionamiento de la edición de datos, que era causado debido a que los botones de la tabla de posiciones seguían habilitados y aun era posible crear nuevas ventanas de edición afectando así el programa, se decide optar por deshabilitar los botones de la ventana de posiciones mientras la ventana de edición esté activa, y esto soluciona posibles errores al poder interactuar con ambas ventanas a la vez.

Al comprobar la funcionalidad del código se decidió revisar qué partes podían ser optimizadas, gracias al compañero David Solís quien dio la idea de utilizar *for* para cargar los datos de la pantalla, se creó un código que recorría todos los elementos de la lista del documento y cargaba las diferentes partes de la fila usando iteración, lo cual simplificó en gran medida el código.

El módulo ttk de Tkinter resultó ser una herramienta muy útil para el desarrollo de pestañas y simplificación de lógica de selección mediante el uso de radio button. Al principio se quería desarrollar un botón de selección asociado a cada uno de los logos para la Escudería y también para la selección de uno de los dos pilotos de la presente Temporada para poder realizar el Test Drive, sin embargo, con radio button se logró obtener la selección del usuario mediante el click del mouse en un área específica de la ventana para cada opción seleccionable.

Como el comando sense retorna si es de día o de noche, así como el nivel de la batería; mediante un Thread con un time.sleep de dos segundos, para no sobrecargar tanto la comunicación con el carro, se quería estar comprobando estos dos parámetros y a su vez poderlos visualizar en la ventana de Test Drive, sin embargo, solo se podía ejecutar, o bien la función que muestra en pantalla el nivel de batería con su respectivo porcentaje, o la función que cambia el fondo principal del Test Drive dependiendo del valor del sensor (1 o 0). Luego de varios intentos para lograr que ambas funciones trabajaran simultáneamente dentro de la función que las contenía, se llegó a la conclusión de que esto se debía al principio de ejecución secuencial de instrucciones, es decir, cuando dentro de la función que contiene a las dos funciones se llega a la primera función esta se ejecuta, pero como en ningún momento se retorna ningún valor al punto desde donde fue llamada, la segunda función nunca se podrá ejecutar. Para solucionar esto se creó un Thread que ejecuta una de las dos funciones mencionadas anteriormente, siendo este Thread dependiente de la otra función.

Modularizar un bloque de código implica que este mismo es indispensable para la lógica general de la interfaz, en este caso el haber modularizado las validaciones de limitación del txt de los pilotos y autos fue necesario considerando que el usuario puede modificar la información de cada uno de ellos y es por eso que se ocupa siempre verificar que no se haya alterado de sobre manera tal información de cada uno.

El efecto de animación inicial de la interfaz al estar condicionado a una duración de 2.7 segundos es posible que su ejecución se interrumpa al cumplirse el tiempo establecido o bien que el efecto de animación ya haya terminado, en ambos casos puede o no que se genere una excepción de ejecución. Implementar try-except fue una medida de seguridad para que independientemente de las dos situaciones antes mencionadas, el programa sigua su ejecución sin ningún inconveniente.

Al desarrollarse la interfaz de la ventana Test Drive, en un principio usando Labels, se presentaron inconvenientes con aspectos como la transparencia, pues si bien las imágenes estaban en formato png, el label ignoraba ese atributo y colocaba como fondo predeterminado de la imagen, el color blanco; esto realmente fue un inconveniente pues volvía a la interfaz menos atractiva para el usuario, es por eso que toda la lógica se pasó a textos e imágenes montados directamente sobre el canvas, con ello las imágenes con fondo transparente mantenían el formato png en la interfaz y los textos dejaban de tener forzadamente un fondo.

El evento keyrelease a menudo suele ser una herramienta indispensable para toda interfaz que interactúa directamente con el usuario, sin embargo, a medida que se avanzaba con el desarrollo de la interfaz de la ventana de manejo del carro (Drive Car) su uso se notó innecesario, pues con flags se lograba obtener el mismo resultado, inclusive algunos mejores y más sencillos de entender en la lógica del programa en general.

Una vez desarrollada toda la lógica del manejo de las luces, movimientos y demás comandos especiales, se comprende que cada uno de ellos son manejados internamente por el programa y no por el usuario, por tal hecho, la función creada para comprobar el comando que se le envia al NodeMCU, si bien era indispensable para la primera parte del proyecto, para este era innecesario, por tal motivo se eliminó del mismo.

El comando Sense devuelve un string que contiene el nivel de la batería y el estado del sensor, acceder mediante índices a tales dato parece ser la mejor opción en primera instancia, sin embargo, al considerar que la cantidad de comandos que el usuario manda al NodeMCU para manejar al mismo, el índice previamente establecido suele correrse y por lo tanto devolver datos incorrectos afectando la interfaz de la ventana del Drive Car, es por esto que como medida de seguridad, implementar una función que recorre el string y almacena solamente los datos requeridos para mostrar el porcentaje de batería en pantalla y el estado del sensor, disminuye el margen de error en la ejecución del programa.

Los motores, son los elementos físicos que más consumo de energía requieren, por lo tanto es común ver que el porcentaje de batería en la interfaz disminuye cada vez que este se usa para mover el carro hacia adelante o hacia atrás o en cualquiera de las direcciones (derecha o izquierda). Para asegurarse que esto no pase, o bien que no sea muy notorio, aumentar los rangos entre un porcentaje de batería y otro permite que el cambio del porcentaje en la interfaz del Drive Car sea menos notorio o frecuente.

El carro tiene funcionalidades automáticas; encendido de las luces traseras cada vez que el carro se mueve hacia atrás, cambio del fondo en la interfaz dependiendo si es de día o de noche, actualización del nivel de batería, entre otras. Por su característica de automatización pueden que haya conflictos con los comandos manuales, por eso usar flags es una buena opción ya que reduce las excepciones que se puedan producir.

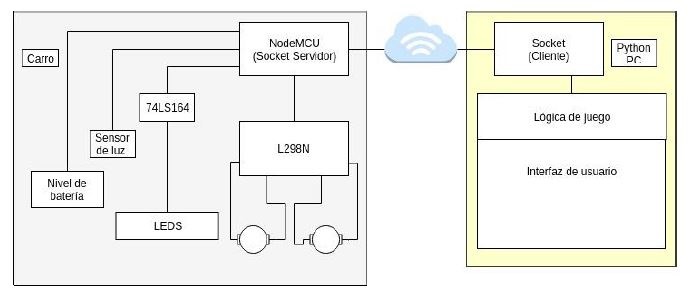
Al hacer pruebas del manejo del carro desde la interfaz de Tkinter, por la sensibilidad del sensor de luz, este detecta, incluso estando de día, que es de noche y por eso cambia el fondo de la interfaz del Drive Car en modo nocturno. Haciendo un análisis de tal motivo se llega a la conclusión de que el el circuito del sensor de luz, es muy sensible por los valores y la alta tolerancia de las resistencias que este contiene. Para un futuro en que por equis motivo se tenga que abrir el carro, cambiar las resistencias del circuito detector de luz, volverían a este menos sensible.

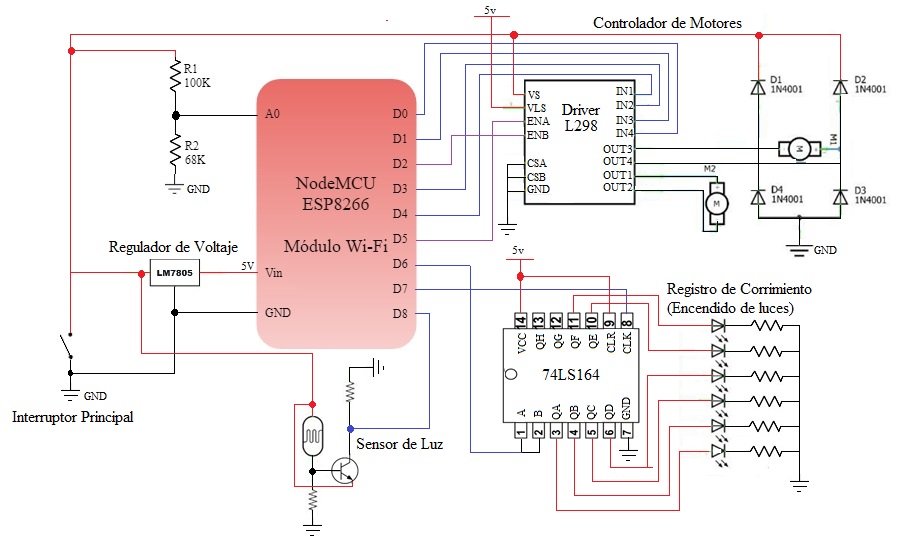
Cada vez que el carro se pone a funcionar con batería nuevas, recargables o de buena calidad, la eficiencia de este aumenta considerablemente, sin embargo, cuando las baterías ya están usadas o son de una marca mala, paso lo contrario. Como usuario de la interfaz, uno siempre desea que el carro funcione de la mejor manera posible, que responda a cada comando casí instantáneamente, que encienda las luces con alta intensidad y así; es por ello que para que ese cambio de fuente de energía no sea notoria en la eficiencia del carro, configurar parámetros como el pwm son una buena opción, pues de está forma si las baterías del carro son buenas el pwm para el movimiento hacia adelante y hacia atrás del carro perfectamente puede iniciar en 400 o -400 según corresponda, por otra parte, establecer el pwm con valores iniciales de 700 0 -700 compensarán el hecho de que las baterías del carro sean de mala calidad.

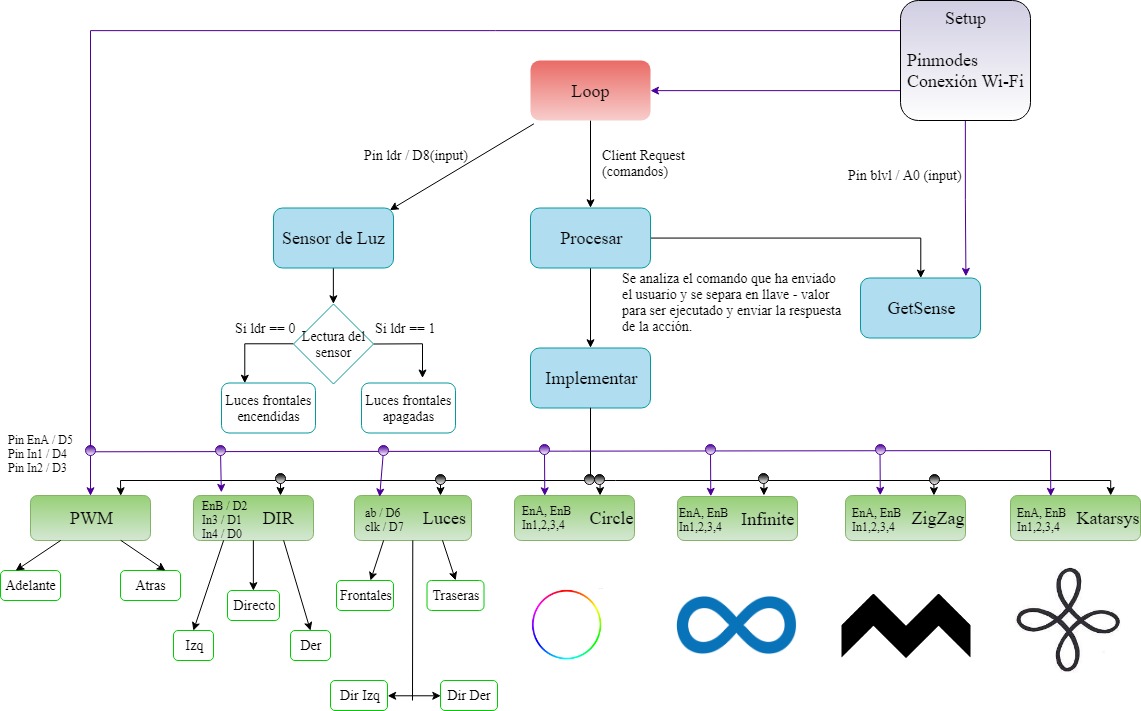
Se presentaba el problema de que habían demasiadas funciones en el programa, sobre todo en la ventana del Drive Car, ante tal situación se quería simplificar de una u otra forma líneas del código y para eso se investigó un poco hasta que se encontró el método char con lo cual se logró hacer de cuatro funciones, una sola que cumplía con lo mismo que hacían las cuatro funciones anteriores, mediante la captación de eventos específicos definidos por el programador, este método fue usado para el manejo total de todas las luces del carro.

Diagramas de Módulos

**Diagrama Arquitectura Hardware - Software**







Plan de Pruebas

En aspectos generales cabe decir que se hicieron gran cantidad de pruebas de todo tipo a lo largo del desarrollo del proyecto. Siento un poco más puntuales podemos decir que dicho a manera de plan las pruebas se realizaban bajo el siguiente método:

Luego de realizar en la protoboard una nueva conexión o se añadiera una nueva resistencia que tendría que conectarse directamente con alguna de las LEDs o los motores usados, o se conectara al NodeMCU se revisaba con anterioridad el voltaje que iban a recibir estos componentes, y así asegurar que no iban a recibir más voltaje del necesario y evitar que no llegaran a quemarse, de igual forma que el voltaje recibido fuera más del mínimo para operar, esta prueba se realizaba con cada modificación en la protoboard dirigida al hardware del carro. Además se realizaban distintas pruebas de voltaje una vez iniciado el montaje y soldadura de los distintos elementos en la placa preperforada, esta prueba se realizaba previa a la soldadura y así mantener la certeza de un correcto ensamblaje del circuito.

Para verificar la correcta conexión entre la interfaz de Tkinter y el carro, se realizan distintas pruebas a las cuales primeramente se obtienen errores, hasta que se prueba nuevamente con todos los archivos necesarios abiertos y funciona sin mayor problema salvo ciertas excepciones.

El código era revisado en tres ocasiones, la primera de estas al compilar y revisar los posibles errores indicados por el mismo programa de Arduino, los cuales constituyen errores de sintaxis o son errores semánticos, una vez superada esta prueba se procedía a verificar su funcionamiento por medio de los circuitos creados de manera independiente para cada función específica (si se trabata de los motores, luces por control de comandos, y luces por control de la fotoresistencia), esto se comprobaba conectando el NodeMCU tanto a la protoboard como a la computadora utilizada para cargar el código y así poder revisar su comportamiento de manera más detallada por medio de los prints en el monitor serial y comandos ya previamente asignados, una vez superada esta prueba se realizó una definitiva que tomaba lugar luego del montaje del circuito ya completo y ya unidas las distintas partes del código que fue creado de manera separada según cada función.

Otra de las pruebas realizadas fue del mínimo número de potencia que se le puede asignar a los motores de modo que estos tuvieran algún movimiento, luego de las pruebas en el motor de tracción, esto ya que el de dirección se dejó siempre con un pwm de 1023, se demuestra que el número mínimo sería 500.

Una vez iniciado el proceso de soldadura en placa se realizaban pruebas de continuidad del estaño con mucha frecuencia, utilizando el multímetro, esto debido a que en caso de detectarse una continuidad donde no debía haberla era necesario proceder a eliminarla de manera inmediata para mantener el funcionamiento correcto del circuito y evitar así problemas futuros, en caso de encontrarse error de continuidad se usaba el retirador de soldadura y cuchillas de precisión hasta que la continuidad diera negativa y se procedía con el resto de la soldadura, repitiendo de manera periódica las pruebas de continuidad a lo largo de todo el circuito, incluso una vez finalizado.

A la hora de montar la rectificación se volvió necesario hacer pruebas sobre la polaridad de los diodos, esto debido a que era prácticamente imposible delimitarla únicamente con la vista y en caso de colocarse de manera errónea produciría un error en el circuito dejándolo inutilizable, por ello, debido al peligro que representaba se hacían pruebas constantes al momento de colocar cada diodo en su lugar, repasando además que tuviera una continuidad con los elementos correspondientes.

Una vez montado el circuito en su totalidad se realizaron nuevamente pruebas de voltaje, continuidad del estaño, colocación de diodos, tres veces cada una para dar certeza de su correcto funcionamiento, se completó con estas pruebas de manera exitosa por lo que se pasaría a la siguiente.

Una vez probado el circuito, y el código, se decide probar ahora el funcionamiento completo de este, por esta razón se vuelve a conectar el NodeMCU al carro previamente encendido y se hacen dos pruebas completas de comandos, al operar estas con normalidad, se retiró el NodeMCU de la computadora y se vuelve a comprobar con dos rondas completas de comandos el funcionamiento óptimo del carro.

Realizadas estas pruebas se procede con la creación del código que controlará los comandos especiales, los cuáles necesitan definirse por medio de delays en el código, para este propósito se carga dicho código y se prueba hasta alcanzar un comportamiento esperado, así con los cuatro tipos de códigos especiales solicitados.

PLAN DE PRUEBAS TERCER PROYECTO

|  |  |  |
| --- | --- | --- |
| Prueba | Sí | No |
| Hay comunicación entre las ventanas principales de la interfaz (About y Position Table y Test Drive). |  |  |
| La computadora envía los comandos de encendido y apagado de las luces del carro cada vez que el usuario presiona un tecla específica para cada luz. |  |  |
| El carro se mueve en todas las direcciones que el usuario decida desde la interfaz mediante las flechas del Teclado. |  |  |
| El encendido y apagado de las luces se refleja en la interfaz del Drive Car. |  |  |
| Cada movimiento del carro (izquierda, derecha, adelante y atrás) se presenta mediante flechas en la interfaz del Drive Car. |  |  |
| El carro acelera mientras se presiona la tecla para moverse hacia adelante y atrás. |  |  |
| Las luces traseras del carro se encienden cada vez que el carro se mueve hacia atrás. |  |  |
| La luz direccional izquierda enciende y se apaga tres veces cuando el carro gira a la izquierda. |  |  |
| La luz direccional derecha enciende y se apaga tres veces cuando el carro gira a la derecha. |  |  |
| Se muestra la aceleración o no del carro en la interfaz del Drive Car. |  |  |
| El fondo de la interfaz del Drive Car cambia de acuerdo a la lectura del sensor de luz que integra el carro. |  |  |
| El porcentaje de batería se refleja en la interfaz y se actualiza conforme el carro se vaya descargando. |  |  |
| El sensor de luz y el nivel de batería funcionan simultáneamente. |  |  |
| La animación inicial de la interfaz dura el tiempo establecido. |  |  |
| Al concluir la animación inicial de la interfaz se muestran los botones para acceder a las demás ventanas. |  |  |
| La música de Fondo de la interfaz suena indefinidamente. |  |  |
| La música de fondo puede ser silenciada y se puede volver a reproducir. |  |  |
| El logo de la Escudería se presenta en la ventana principal de la interfaz. |  |  |
| El estado actual de la carro se presenta en la ventana principal de la interfaz. |  |  |
| La ventana de información de la Escudería se comunica con la ventana de información. |  |  |
| La ventana de información de la Escudería muestra todos los datos solicitados en las especificaciones del proyecto. |  |  |
| Se puede cambiar el logo de la Escudería y respectivamente su información (nombre y ubicación). |  |  |
| Los patrocinadores de la Escudería pueden ser editados. |  |  |
| Hay acceso desde la ventana de información de la Escudería a la tabla de posiciones de los pilotos y los autos y viceversa. |  |  |
| La ventana de información de la Escudería se actualiza cada vez que el usuario hace un cambio en ella. |  |  |
| El estado del carro en la ventana principal cambia de acuerdo al nivel de las baterías del carro. |  |  |
| El estado actual del carro se muestra en la ventana del Test Drive. |  |  |
| El usuario puede seleccionar uno de los pilotos de la temporada actual para poder realizar el Drive Car. |  |  |
| Se puede realizar el manejo del carro en la ventana Drive Car siempre y cuando el usuario haya seleccionado uno de los pilotos de la Temporada actual y el estado del carro sea disponible. |  |  |
| En la interfaz de la ventana del Drive Car se muestra el nombre  y la nacionalidad del piloto que el usuario seleccionó para realizar el Test del carro. |  |  |
| Se muestra el nombre de la Escudería en la interfaz del Drive Car |  |  |
| Motor de tracción se mueven para adelante y atrás según señal |  |  |
| El piloto seleccionado tiene una celebración específica. |  |  |
| El usuario puede seleccionar cualquiera de los comandos especiales del carro (ZigZag, Circle, Infinite y Katarsys) para que este inmediatamente los ejecute. |  |  |
| La aceleración del carro se muestra en la interfaz del Drive Car. |  |  |
| Los carros y los pilotos de la Escudería tienen todos los datos acordes a las especificaciones del proyecto. |  |  |
| Desde la ventana de Position Table se pueden ver en pestañas aparte los pilotos y los carros de la Escudería. |  |  |
| Los pilotos se orden de forma ascenderte de acuerdo al rendimiento global del piloto o el rendimiento específico del piloto. |  |  |
| Los pilotos se orden de forma descendente de acuerdo al rendimiento global del piloto o el rendimiento específico del piloto. |  |  |
| Los carros se orden de forma ascendente con respecto a la eficiencia. |  |  |
| Los carros se orden de forma descendente con respecto a la eficiencia. |  |  |
| Todas las imágenes de los pilotos están asociadas a la información de cada uno de ellos. |  |  |
| Todas las imágenes de los carros están asociados a la información de cada uno de ellos. |  |  |
| El usuario puede editar la información de cada piloto de la Escudería |  |  |
| El usuario puede editar la información de cada carro de la Escudería |  |  |
| Al final de cada Temporada se le permite al usuario agregar un nuevo auto a la Escudería. |  |  |
| En la ventana principal aparece la Temporada actual. |  |  |
| La lectura del porcentaje de la batería es más estable |  |  |

**Bitácora**

Viernes 12 de abril

Entrada de Emanuel Marín: Se compró el kit organizado por el asistente.

Domingo 14 de abril

Entrada de Emanuel Marín: Se inicia la construcción del circuito proporcionado por el asistente en una protoboard.

Lunes 15 de abril

Entrada grupal: Se compraron jumpers macho-hembra, 16 baterías AA, 1 batería de 9V, se compra el carro a usarse para el proyecto.

Se comprueba que el carro funciona.

Se desarma el carro para poder disponer de las conexiones que se necesitarán luego para realizar el proyecto, se nota que hay cables pegados con cinta adhesiva y un exceso de silicón, y se verifica que tenga dos motores.

Se termina de montar el circuito brindado en la protoboard sin conectarlos aún al carro.

Se inicia el análisis del archivo .ino dado por el asistente del curso, por lo que se procede a investigar sobre las funciones y usos de pinMode, de millis, y se investiga acerca del funcionamiento de los operadores lógicos siguiendo la recomendación del asistente para el registro de corrimiento que va a controlar las LEDs, así mismo se investiga el comportamiento de shiftOut que funciona con el mismo propósito de manipular las luces del carro.

Se investiga la estructura del código de swithcase, su uso y su comportamiento, para una mejor comprensión del código.

Martes 16 de abril

Entrada grupal: Se quiere comprobar la funcionalidad del NodeMCU por lo que se dispone un código sencillo con LEDs y al estar conectado por USB a la laptop opera de la manera esperada, al querer asegurarse de que funcione por medio de WiFi sin necesidad de estar conectada se retira el USB y el módulo así deja de funcionar. Se consultó al asistente acerca de esta conducta y nos indicó un método que garantice la eliminación de este error, ya que, según nos indicó, debe cargarse el código al módulo, conectar una fuente de poder suficiente al módulo y posterior a eso retirar el cable USB, se prueba y funciona.

Se inició la unión de los motores del carro con el circuito de la protoboard, se empezó conectando únicamente el motor de tracción al circuito, no ocurre nada. Se verifica el voltaje en las salidas del driver (L298) y aunque existe una diferencia de tensión el motor no enciende. Se probó aumentando el voltaje de las salidas, sin éxito. Se aumenta ahora la corriente, sigue sin encender el motor. Se prueba conectar en paralelo una batería adicional (9V) en las salidas, y el motor enciende, pero al quitar esta batería el motor dejaba de funcionar. Se debe solucionar.

Miércoles 17 de abril

Entrada grupal: Se compran jumper hembra-hembra, dos placas preperforadas, y dos baterías de 9V adicionales.

Se sigue trabajando en conseguir el arranque, de manera ideal, del motor de tracción. Se prueba tomar una salida y conectarla a uno de los cables del motor y el otro conectarlo a tierra, el motor funciona. Considerando el resultado obtenido ayer donde el motor solo operaba con una batería adicional y este último se logra deducir que es necesario que el circuito del motor posea rectificación (según lo indicado en el programa), además, analizando con mayor profundidad el diagrama del circuito se llega a comprender que ambos motores comparten sistema de rectificación y que al no tenerla no se genera una tierra común lo que no da paso a completar el circuito, por tanto sin esto, el motor nunca llegaría a funcionar. Debido a la necesidad de la rectificación se procede a montar el circuito del segundo motor con la rectificación debida según el diagrama dispuesto en las especificaciones del proyecto, se conecta a una fuente de poder y enciende. Posterior a eso se desean probar los distintos casos posibles que puedan realizar los motores por lo que se genera un código .ino que compruebe estos comportamientos, se crea, se carga al ModeMCU y se verifica de manera satisfactoria que el circuito está colocado de manera correcta.

Entrada de Emanuel Marín: Se comienza a trabajar en el sensor de luz (con fotoresistencia), se monta el circuito indicado en el diagrama, luego de varios intentos sin éxito, se investigan otras formas de implementar este mecanismo y se concluye que en lugar de trabajar con un transistor pnp es mejor implementar el sensor con un transistor npn, específicamente el 2N2222A, se implementó el circuito y el sensor actúa de manera esperada pero no detectaba el pin de entrada ldr de manera correcta por lo que se analiza.

Entrada de Alejandro Vásquez: Además, se inicia la creación del código necesario para ejecutar los comandos, para esto se analiza nuevamente el código con mayor detalle y se vuelve necesario investigar acerca de la función STR.FORMAT en Python y su aparente similar en C llamada sprintf. Al investigar queda claro que no son ni parecidas. Se procede a modificar los distintos códigos de pruebas en el código dado por el asistente de modo que, según se espera, logre controlar las distintas funciones del carro, aun no es posible probarlos de manera directa por lo que funcionan únicamente a nivel teórico, se construyen las condiciones para el movimiento del carro, su dirección, las distintas luces, así como también la función para medir ldr y el voltaje de la batería, este último siendo mucho más complejo que el otro requirió de una investigación más profunda donde se encontró una página cuya lógica y código permitieron la formación del código implementado por nosotros para obtener el porcentaje de la batería, utilizando distintas fórmulas que permiten obtener el voltaje a partir de resistencias, así como la idea de utilizar un promedio de 10 valores para hacer más preciso el voltaje calculado.

Jueves 18 de abril

Entrada grupal: Continuando con el problema del sensor de luz, se realiza otra prueba en relación a la detección del pin ldr y se llega a la deducción de que alimentar el sensor con 3V desde el NodeMCU no es lo más idóneo, ante ello, se cambia la alimentación por una directa de 7.5V (la batería propia del carro), y por tanto se cambian los valores de las resistencias. Se cambia la posición del pin ldr con respecto al sensor en la protoboard, con estos cambios se demuestra que sí funciona. De misma manera que con los motores se desarrolló un ligero código para comprobar que el comportamiento fuese adecuado.

Se realizan pruebas individuales del funcionamiento de los motores y luces para comprobar nuevamente que el código creado y que las conexiones del circuito operen. Se comprueba su funcionamiento por lo que se procede a ensamblar las distintas partes en una sola protoboard, así como unir los fragmentos de código dentro de uno solo. La función que determina el porcentaje de la batería se modifica para que dependiendo del voltaje no haga prints en el serial del porcentaje, sino que retorne el valor entero según el voltaje ya que la función getSense se encarga de realizar dicho print.

Entrada grupal: Se modifica la función getSense para que en lugar de imprimir -1 para la batería y el sensor, imprima el resultado de la función que calcula el voltaje y arroja un porcentaje en relación a este, así como también en light haga un digitalRead del pin D8 (ldr) y así indicar por medio de 1 o 0 su estado.

Entrada grupal: Se termina de montar todo el circuito y la programación en uno solo. Se pasa a intentar enviar comandos a partir de la ventana de Tkinter dada por el asistente, a su vez se le pregunta si el código es totalmente funcional. Al inicio ocurrían errores de tipo Timed out, pero ambos se conectaban a la red de manera satisfactoria. Se abrió el IDLE del archivo WiFi Client para revisar su código y se verifica que las IP en este archivo y en el de Arduino sea la misma. A su vez se consulta nuevamente y con mayor detalle el funcionamiento de los códigos en Python ya que aún no se logra conectar.

Viernes 19 de abril

Entrada grupal: Se continúa intentando conectar la interfaz de Tkinter con el archivo de Arduino y a su vez con el NodeMCU. Sin mayor éxito. Se vuelven a abrir ambos archivos .py y se intenta nuevamente, sin éxito. Se intenta una vez más y se muestra el mensaje “ok;” en la ventana de Tkinter, por tanto se logró conectar. Se envían los distintos comandos hasta hacer una prueba completa de todos, y operan según lo esperado. Se reinicia para seguir haciendo pruebas, el primer comando siempre es un Timed out, funciona correctamente a partir del segundo. Se realizan pruebas de todos los comandos, pero con el NodeMCU desconectado de la laptop, de igual forma actúan de manera esperada. Cada ciertos comandos ocurre un error de conexión que se arregla inmediatamente al ingresar nuevamente el comando. Aún no se sabe el porqué.

Se realizan más pruebas con la fotoresistencia debido a que afecta las luces frontales de una manera inadecuada, ya que se quiere que la fotoresistencia encienda las luces según detecte o no luz, se verifican muchos casos de en la programación, se descubre que es por problemas de dominación, se buscan solucionar, sin pronto éxito. De forma empírica, y luego de gran cantidad de intentos se logra llegar a un comportamiento lógico de manipulación de dichas luces.

Se nota que la fotoresistencia tiene una sensibilidad dependiente del voltaje por lo que decide regularse con un potenciómetro para mantener la sensibilidad de una forma más estable.

Sábado 20 de abril

Entrada de Alejandro Vásquez: Se crean las etiquetas con el nombre “Katarsys A7” que se usarán en el carro usando PhotoShop CS6.

Entrada de Emanuel Marín: Se realizan pruebas al circuito para verificar el funcionamiento del NodeMCU al apagar y encender el carro. Al hacer esto, el NodeMCU no recibe comandos. Se hacen pruebas para verificar donde está el error. Se comprueba que el NodeMCU enciende. También se conecta a la red. Pero no recibe comandos, por tanto se concluye que es un error en el código.

Entrada de Alejandro Vásquez: Se comienza a idear el rediseño del carro, se calcula el espacio que requerirá el circuito y se le adapta un espacio dentro del carro. Se decide que se necesitará tener acceso al NodeMCU una vez esté dentro del carro para poder realizar actualizaciones en el código, se adapta el carro de manera que cumpla con estos requisitos retirando los asientos traseros, para poder tener acceso al NodeMCU, que se colocará en la parte trasera del carro se decide retirar la ventana de la cajuela y que esta se comporte ahora como una compuerta. Además se calcula el orificio que se requiere para colocar la fotoresistencia y se dibuja en el espacio elegido. Para evitar que se vea el circuito arruinando la estética del carro se decide pintar de negro las ventanas y así mantener su diseño elegante.

Se pegan las etiquetas con el nombre de la escudería en el lugar de las placas delantera y trasera.

Entrada grupal: Se ve necesario pintar las ventanas superiores para ocultar, de manera satisfactoria, el circuito de la vista. Se decide usar un color plateado.

Se cortó la carrocería que impedía que se pudieran ver las luces direccionales.

Se inicia con la soldadura del circuito en la placa preperforada.

Se hizo el orificio por el cual se introduce la fotoresistencia, se hizo con el cautín.

Una vez soldada gran parte del circuito se hicieron pruebas de continuidad.

Domingo 21 de abril

Entrada grupal: Se inicia el montaje del carro. Previamente se hacen pruebas que verifiquen el funcionamiento de las LEDs, todas operan con normalidad, se despega el cable positivo de las luces adheridas desde fábrica a la parte delantera del carro, por lo que se procede a remover dicha parte y así poder conectar otro cable y que continúe funcionando.

En el orificio realizado para las direccionales se pegan con silicón las LEDs elegidas para este propósito. Anteriormente se adhirieron cables a las patas de las LEDs para así poder conectarlas con el integrado, esto se hizo por medio de conexiones con estaño, se midió la continuidad de estas con el voltímetro, al funcionar en el circuito se verifica su  correcto funcionamiento y ahora sí se proceden a pegar en su lugar correspondiente en la carrocería del carro.

Se comienzan a montar las distintas partes del carro de juguete (parte inferior, asientos y carrocería). Se introducen los distintos cables de los motores, las luces y la fuente de poder por debajo de los asientos para conectarlos al circuito, el cable de la fotoresistencia se pega al techo con cinta aislante para que no arruine la estética del carro. Debido a que los cables de las luces están sujetos a la carrocería se vuelve cada vez más difícil manipular las distintas partes del carro. Se hace presión en el circuito para que calce de manera casi perfecta en el espacio asignado por lo que queda sujeto a este.

Se hacen pruebas nuevamente para verificar que ninguna conexión se haya alterado y ocurren errores que luego se descubre que corresponden a los cables conectados al L298, ya que se sueltan con gran facilidad por lo que se deciden fijar con cinca aislante y se pega el L298 con silicón directamente al carro.

Se decide aprovechar los espacios para tornillos del NodeMCU y se crean con el cautin dos orificios más en el carro para poder colocar los tornillos. Se termina de montar todo el carro, al inicio con dificultades debido a algunos cables que quedaban afuera, luego de asegurarse de que estuvieran dentro de la carrocería se reanuda el intento de ensamblar por completo el carro, se logra ahora sin mayor dificultad.

Entrada de Emanuel Marín: El NodeMCU queda demasiado abajo para poder conectar el cable USB en caso de ser necesaria alguna modificación por lo que se le quitan los tornillos

Entrada grupal: Se hacen pruebas nuevamente y todo opera con normalidad.

Jueves 25 de abril

Entrada grupal: Según se había acordado se realiza la llamada por Skype a las 6pm, cuyo fin era plantear y escribir los códigos para los distintos comandos especiales.

Se crea el código del comando circle, para que este sea hacia la izquierda o hacia la derecha según lo quiera el usuario. Pero la velocidad sí es decidida por los desarrolladores.

De igual forma se agarra la base de este código para poder realizar el de zigzag y el que crea el símbolo de infinito, esto debido a que ambos involucran girar, utilizando los delay*s* y cambio en la configuración de direcciones apropiados se consiguen los códigos de dichos comandos especiales sin mayor dificultad. Posteriormente se piensa en el comando especial y personalizado y se decide crear el código para uno que consiste en 4 círculos interconectados, este comando recibirá el mismo nombre que la marca del carro, es decir, “Katarsys”, se decidió que fueran círculos debido a que crear ángulos de 90 grados es imposible con este carro en particular.

El integrante Emanuel, prueba los comandos de los tres primeros comandos especiales y funcionan a medias debido a que las llantas no giran hacia la derecha, únicamente a la izquierda, por lo demás, se comportan de manera esperada cuando deben girar a la izquierda.

Viernes 26 de abril

Entrada grupal: Se decide hacer una reunión de emergencia a las 9:20 am para encontrar y solucionar el problema encontrado el día anterior. Se cree que es un problema de los motores, por lo que se desarma el carro nuevamente y se revisa de manera superficial los motores, se intentan desarmar pero resulta imposible desatornillarlo para poder retirarlo y examinarlo de manera más meticulosa, al levantar el carro las llantas giran a la derecha por lo que se hacen distintas pruebas sobre qué podría estar causando ese comportamiento.

Se llega a la conclusión de que el problema está en el L298 y los cables que actúan como pines, por tanto se deciden ajustar con más cinta de manera temporal, sin montar el carro nuevamente hasta asegurarlo correctamente

Domingo 28 de abril

Entrada de Emanuel Marín: Se decide usar un pegamento especial, más potente para asegurar los cables y evitar ese tipo de problemas nuevamente.

Martes 30 de abril

Entrada grupal: Se termina de montar el código para el comando “Katarsys”, y se prueba nuevamente el funcionamiento del carro.

**Inicio de la Bitácora III Proyecto**

Viernes 17 de mayo

Entrada grupal: Se designa a Emanuel Marín como líder del grupo así como que el trabajo se dividirá por ventanas; al líder le corresponde trabajar en la ventana Test Drive y al integrante Alejandro Vásquez le corresponde trabajar en la ventana de Posiciones.

Entrada de Emanuel Marín: Se leen las especificaciones del tercer proyecto y se le pregunta al asistente del curso ciertas dudas respecto al mismo.

Sábado 18 de mayo

Entrada de Emanuel Marín: Se crea un repositorio en GitHub, ya que se llegue a la conclusión de que es más fácil de usar que GitLab; en el repositorio, se suben todos los archivos trabajados en el segundo proyecto para luego editarlos en ramas distintas de acuerdo a las especificaciones del proyecto.

Se hace una videollamada con el compañero Alejandro Vázquez para enseñarle a usar GitHub y de paso crear las ramas en los que ambos estaremos trabajando independientemente. Se acuerda dividir el trabajo por ventanas, es decir, él trabajará la ventana de tabla de posiciones y yo las ventanas restantes.

Se empieza a editar el código TelemetryLog, sobretodo se trabaja sobre la ventana *Test Drive.* Se logra asociar teclas a los comandos del carro y se comprueba con éxito sobre el mismo.

Domingo 19 de mayo

Entrada de Emanuel Marín: Se crea una función para el manejo independiente de todas las luces del carro. Se logra aumentar o disminuir la velocidad en forma progresiva, así como manejar las direccionales; se hacen pruebas al carro y este funciona de manera esperada.

Lunes 20 de mayo

Entrada de Emanuel Marín: Se crean unos cuadros que representan el color de las luces del carro y se logran adaptar a la interfaz de manera que si el usuario enciende las luces delanteras en la interfaz aparezca el cuadro de color blanco y así con las demás luces. También se desarrolla la lógica para que el usuario asocie que un click a la tecla específica representa el encendido de la luz y otra click a la la misma tecla representa el apagado de la luz y viceversa.

Entrada de Alejandro Vásquez: Se crean las carpetas en las cuáles se almacenarán las imágenes necesarias para el proyecto, así como también se crean los archivos .txt que guardarán los datos solicitados de los autos y los pilotos. Se descargan las 10 imágenes de los carros, así como las fotografías de los pilotos. Se inicia a montar la ventana de posiciones creando los botones “back”, “Ascendente”, “Descendente”, “Tabla de Autos”. Se decide que la primera tabla que se mostrará será la de los pilotos, por esta razón se crea una ventana adicional que contará con la tabla de posiciones para autos. Se crea la imagen que se utilizará en la ventana About. Y se decide la implementación de efectos de sonido y música en la interfaz.

Martes 21 de mayo

Entrada de Emanuel Marín: Con respecto al movimiento del carro hacia adelante y hacia atrás, se crea un canvas el cual muestra en pantalla el pwm actual del carro, de esta forma cuando el carro acelere inmediatamente ese cambio de velocidad se refleje en la interfaz.

Entrada de Alejandro Vásquez: Se recorta una imagen con varios pilotos para obtener las 10 imágenes que se utilzarán en el proyecto. Se investigan los diferentes widgets de Tkinter y se encuentra la funcionalidad de “pestañas”. Se implementan en la ventana de posiciones creando una ventana con la información de los autos y otra con la de los pilotos, se comienzan a realizar pruebas de pestañas para entender mejor su funcionamiento, nombre, creación, orden, creación en una específica y demás.

Se montan los textos de los pilotos en labels para poder iniciar con el ordenamiento, las imágenes aun no debido a que se desconoce cómo.

Se comienza a pensar la lógica de ordenamiento por lo que se repasan los diferentes métodos de ordenamientos vistos en clase y se decide utilizar el de inserción, usando el código base brindado por el profesor Milton Villegas Lemus. Se prueba primero en el botón ascendente de REP, funciona por lo que se implementa a los demás botones. Se descubre que para cambiar el orden de la función basta con cambiar el signo “<” o “>” según se desee. El ordenamiento de los carros depende únicamente de un parámetro mientras que el de que pilotos puede ser realizado con dos se construyen funciones específicas para cada uno de los 6 tipos de ordenamiento.

Miércoles 22 de mayo

Entrada de Emanuel Marín: Se asocian las teclas del movimiento del carro con flechas (imágenes), de esta forma si el carro gira hacia la derecha que aparezca una flecha en la interfaz indicando tal giro y lo mismo con todas las demás acciones de movimiento del carro. También se configura que si el carro se mueve hacia adelante o atrás y gire a la izquierda o la derecha, ambas acciones simultáneamente se reflejen en la interfaz.

Jueves 23 de mayo

Entrada de Emanuel Marín: Se monta un Thread para la acción cuando el carro gira  a la derecha o a la izquierda, así apenas se realiza la acción, se inicia un Thread el cual lo que hace es encender y apagar tres veces las luz direccional específica con un tiempo entre en cada encendido estratégico para que parezca como que el usuario ha decidido girar. También se asocia el encendido permanente de las luces traseras para cuando el carro se mueve hacia atrás.

Entrada de Alejandro Vásquez Oviedo: Se comienza a plantear la solución al problema de cargar datos y sus imágenes de manera que ambos siempre estuvieran en la misma fila, para ello se deciden escribir los nombres de las imágenes en la misma fila de los nombres dentro del archivo txt. Y ya que los datos deben ser editables se colocan al inicio de cada fila del archivo para así impedir que eso se muestre al usuario. Y debido a esta misma razón se comienza a pensar en el método de no depender del .seek para recorrer el documento y no depender de la cantidad de espacios que pueden ser modificados por el usuario, además de arruinar el orden en la interfaz. Se decide trabajar con los datos de forma que únicamente se dependa de los índices, investigar cómo.

Viernes 24 de mayo

Entrada de Emanuel Marín: Se optimiza el código para el manejo y las luces del carro tanto la acción física como la interfaz, se trabaja en tiempos específicos entre el envío de un comando y el de otro con el objetivo de sobrecargar la comunicación entre la computadora y el carro.

Entrada de Alejandro Vásquez: Se le pregunta al compañero David Solís por el método enseñado en las tutorías de Ignacio Mora por el método que permite cargar las imágenes en formato png sin fondo y así tener una interfaz más estilizada. Además, se conoce, gracias a Davis Solís ahora el método de create\_text que permite la creación de texto en el Canvas sin fondo. Y con base en esto se comienzan a montar los textos y las imágenes por fila de cada piloto usando índices y subíndices del txt. Y prueban las funciones de ordenamiento, no funciona de manera automático, solo de manera manual al salir y entrar de nuevo en la ventana. Se investigan los métodos de refresh, update, con manejo de hilos para su actualización en pantalla, nada funciona, se usa el método de destruir y recargar la ventana de forma automática

Sábado 25 de mayo

Entrada de Emanuel Marín Gutiérrez: Se trabaja en el efecto de introducción de la interfaz, con el objetivo de que una vez iniciado el programa, se logre llamar la atención del usuario por la interfaz; la introducción consiste en mover el logo de Fórmula E a la vez que suena el efecto de un carro acelerando. Se establece que el tiempo de duración de la introducción no supere los 2.7 segundos, con el fin de no perder el interés del usuario. Cuando se termina el efecto de introducción se muestra la ventana principal y se corre simultáneamente una canción de fondo.

Se configura que la  música de fondo suene indefinidamente para que el usuario no tenga que volver a reproducirla, a su vez, se crean botones para silenciar y reiniciar la misma.

Entrada de Alejandro Vásquez Oviedo: Se crean los botones que permitirán editar la información de los pilotos, se piensa primero en una función para cada botón lo que haría el código demasiado grande por tanto se piensa en una función que sea capaz de hacer todo y se crea una que reciba la posición del botón y con esto delimite la fila que debe modificar, al agregar argumentos a la función se ejecuta directamente, se busca la manera de solucionarlo, se encuentra el método de usar lambda para evitar que esto suceda. Se crea una función que dependiendo del botón recibe una “y” diferente y un 1 o un 0 dependiendo si es un auto o un piloto, así como un argumento que inicia en 0 para delimitar la columna. Y cuando la encuentra muestra el Entry con los datos de la fila respectiva. Se intenta hacer que se modifique directamente sin depender del seek. Entonces se crea una variable que guarde la lista con el dato modificado y luego se escribe esta lista en el txt correspondiente.

Domingo 26

Entrada de Emanuel Marín Gutiérrez: Se trabaja en la información de la Escudería, para ello se crea un txt con los datos solicitados en la especificación del proyecto (nombre, ubicación, índice ganador, logo, pilotos, autos y patrocinadores). Debido a que el usuario puede cambiar el logo de la Escudería, y por ende, la información de esta, se investigó sobre las Escuderías más importantes que participan en Fórmula E, así como algunas marcas patrocinadoras para ponerlas como información en el txt.

Se trabaja en la representación gráfica del estado del carro con respecto al nivel de batería y se establece que si el nivel de batería es menor igual a 50, significará  que el carro está descargado y por lo tanto no estará disponible para realizar el Test Drive; por lo contrario, si el nivel de batería es mayor que 50, significará que el carro está cargado y por lo tanto, sí se podrá realizar el Test Drive.

Entrada de Alejandro Vásquez Oviedo: Se termina de realizar pruebas que verifiquen la correcta apertura y cierre de archivos, así como su modificación, y se realiza la edición de los fondos de las tablas de pilotos y autos. Se realiza commit de las ediciones realizadas hasta la fecha. Se plantea cómo solucionar que los datos se muestren de forma desordenada según las columnas.

Lunes 27 de mayo

Entrada de Emanuel Marín Gutiérrez: Se sigue trabajando en la información de la Escudería y el estado del carro. Con respecto a lo segundo, se adaptan imágenes para cada caso (disponible y descargado), y se establece un medio de comprobación para el nivel de batería que más adelante se usará para poner el nivel y porcentaje de la misma, en tiempo real, en la ventana del Test Drive.

Una vez concluida la investigación de las diferentes Escuderías que participan en Fórmula E, se establece que desde la ventana principal, se pueda acceder a la información de la Escudería; para ello se crea una nueva ventana, la cual tendrá dicha información, acceso directo para ver todos los pilotos y autos, así como la posibilidad de cambiar el logo, y toda su información a la vez, y agregar o quitar los patrocinadores.

Para la selección del Logo se utiliza el método *radio button* de la Biblioteca ttk de Tkinter el cual permite obtener la selección que ha hecho el usuario al dar click sobre un área específica asociada a la posición del logo, cabe destacar que con el fin de que el usuario pueda escoger uno entre muchos de los logos, se presentan en pantalla los 8 logos de las Escuderías más conocidas en Fórmula E, y dependiendo del logo seleccionado, la información de ubicación, nombre, entre otros, también cambie.

Para la edición de los patrocinadores de la Escudería, se crea un Entry el cual le permite al usuario agregar o quitar  patrocinadores. Hay incovenientes a la hora de actualizar los cambios realizados por el usuario.

Entrada de Alejandro Vásquez Oviedo: Se decide crear diferentes Entrys para los diferentes datos de los autos y los pilotos por tanto se busca un método que permita delimitar dónde termina un dato y donde comienza el otro, para no depender del seek, se crean funciones para esto que resultan ser funcionales y se aplican separando así los diferentes fragmentos de cada línea de texto y permite un mayor orden. Solo se aplica en los pilotos por el momento.

Martes 28 de mayo

Entrada de Emanuel Marín Gutiérrez: Se logra resolver el problema de actualización de cambios realizados por el usuario, se comenta tal hecho al compañero Vasquez, puesto que él también tiene problemas con lo mismo.

Previo a iniciar el Test Drive se crea una ventana con el fin de que el usuario seleccione, antes de iniciar el Test, uno de los dos pilotos de la Temporada actual de la Escudería. Se configura que si el usuario no selecciona ningún piloto o que si el estado del carro es descargado, no se podrá realizar el Test, indicandole al usuario tales razones; si lo anterior no se cumple, el usuario podrá iniciar el Test sin ningún problema.

Entrada de Alejandro Vásquez Oviedo: Se terminan de ordenar los datos de los pilotos utilizando las funciones creadas y se intenta ordenar los datos de los autos, no funciona por lo que se crean otras funciones específicas para este ordenamiento, funcionan y se logran dividir los datos de los autos por columnas. Se crean los entrys que muestren los distintos fragmentos usando las funciones delimitantes. Ocurre un problema ya que Python toma dos entrys como si fueran uno solo ingresando ambos datos en dos entrys distintos. Esto fue un problema debido a que no parecía tener mayor lógica ya que los distintos entrys estaban asociados a diferentes variables y colocados en diferentes posiciones, a todas luces era distintos. Sin embargo se comprobó que los datos que se buscaban ingresar eran correctos, aunque se mostraran en entrys erróneos.

Miércoles 29 de mayo

Entrada de Emanuel Marín: Se hacen mejoras significativas al código de la ventana de la Escuderia y estado del carro.

Jueves 30 de mayo

Entrada de Alejandro Vásquez Oviedo: Luego de diversas pruebas se reescribe la creación de los entrys y ahora funciona sin ningún problema. Además, se editan los fondos de manera que las columnas calcen de mejor manera con los datos mostrados en pantalla. Se realizan pruebas de edición de datos para comprobar el correcto funcionamiento de las diferentes funciones que debe realizar la ventana, como la edición y ordenamiento. Funciona de manera deseada.

Entrada de Emanuel Marín: Se hacen mejoras al código de selección del piloto, se implementa una mejor lógica para la restricción por la cual el usuario pueda o no realizar el Test Drive.

Viernes 31 de mayo

Entrada de Emanuel Marín: Se retoma el código para mostrar en pantalla el nivel de batería (imagen) con su respectivo porcentaje y que se detecte si es de día o de noche. Se hace retroalimentación con el compañero de curso Alejandro Quesada y comentando la lógica general del código para tal efecto, se llega a la conclusión que la instrucción return y pass son equivalentes o realizan la misma función. Además se prevé que será un problema hacerlos funcionar simultáneamente ya que ambas funciones están ligadas a un mismo comando y dentro de una misma función.

Se trabaja en el código y estos funcionan solo si se ejecutan de manera independiente, como se tenía proveído, se piensa en cómo hacer funcionar ambos códigos al mismo tiempo sin que se generen excepciones en la ejecución del programa.

Se optimiza la ventana de Test Drive para que sea más amigable con el usuario.

Entrada de Alejandro Vásquez: Se trabaja para afinar detalles que vuelvan el código más fuerte y menos dependiente de los cambios en las entradas de los usuarios.

Sábado 1 de junio

Entrada de Alejandro Vásquez Oviedo: Se modifica el fondo de la ventana de edición de modo que ahora se muestre encima de los entry los datos que deben ser ingresados, así como la adición de un fondo para hacerlo más estético. Se descubre un problema de manera inesperada al abrir rápidamente dos ventanas de edición de una misma fila, creando dos ventanas de edición en las cuales una aparece con los entrys vacíos y la otra mostrando los datos, sin querer probar qué problema podía ocasionar el reescribir los datos con estos entrys mal creados se prefirió eliminar y buscar una solución donde no se le permita al usuario abrir más de una ventana de edición pues la causa es que los botones de la ventana de posiciones sigan habilitados. Por tanto, se deciden desactivar todos los botones de la tabla de posiciones cuando la ventana de edición se esté mostrando, esto se hizo utilizando un método que deshabilita todos los botones(atributos) de una ventana en específico.

Entrada de Emanuel Marín: Se comenta el código de la ventana de selección del piloto y se logra hacer funcionar simultáneamente tanto el código del nivel de bateria y la detección del tiempo (día y noche) en la interfaz del Test Drive

Se hace merge de la rama del compañero Alejandro Vasquez hacia mi rama para trabajar en una sola rama y ver el producto final del proyecto. Además, se edita el código para que el usuario pueda editar los patrocinadores de la Escudería, sin éxito debido a que, si el usuario desea quitar patrocinadores, estos no se eliminan en el txt, el código solo funciona correctamente si el usuario agrega patrocinadores, esto posiblemente puede ser por problemas en la posición del seek a la hora de guardar el cambio que ha hecho el usuario respecto a los patrocinadores de la Escudería.

Domingo 2 de junio:

Entrada de Emanuel Marín Gutiérrez:

Entrada de Alejandro Vásquez: Se comienzan a editar las imágenes que se usarán de manera definitiva en la ventana Test Drive, incluyendo el modelo del carro, las imágenes de las luces, así como un control de flechas tomado de un control de videojuegos que indicará hacia qué dirección se está moviendo el carro. Se solucionan además dos errores.

**Fuentes Consultadas**

Millis(). (s.f.). Recuperado el 14 de abril, 2019, de <https://www.arduino.cc/reference/en/language/functions/time/millis/>

Baldengineer. (2018, April 30). Arduino pinMode(): When to use and why. Recuperado el 15 de abril, 2019, de <https://www.baldengineer.com/when-to-use-arduinos-pinmode-and-why.html>

Tasos, A. (2011, September 13). Hardware To Software. Recuperado el 15 de abril, 2019, de <http://www.hw2sw.com/2011/09/13/arduino-bitwise-operators-and-advanced-tricks/>

Switch...case. (s.f.). Recuperado el 15 de abril, 2019, de <https://www.arduino.cc/reference/en/language/structure/control-structure/switchcase/>

ShiftOut. (s.f.). Recuperado el 15 de abril, 2019, de <https://www.arduino.cc/en/tutorial/ShiftOut>

ShiftOut(). (s.f.). Recuperado el 15 de abril, 2019, de  <http://manueldelgadocrespo.blogspot.com/p/shiftout_1.html>

(s.f.). Recuperado el 17 de abril, 2019, de <https://www.google.com/search?hl=en&ei=_723XNf6Nu7U5gKg4pzYBg&q=str.format+python+3&oq=str.for&gs_l=psy-ab.1.2.0l10.18848928.18854068..18857232...1.0..0.269.1757.2-8......0....1..gws-wiz.......0i71j0i131j0i67j0i10i30j0i30.sKETh6v0vQo>

Tutorialspoint.com. (s.f.). C library function sprintf(). Recuperado el 17 de abril, 2019, de <https://www.tutorialspoint.com/c_standard_library/c_function_sprintf.htm>

Sprintf. (s.f.).Recuperado el 17 de abril, 2019, de  <http://www.cplusplus.com/reference/cstdio/sprintf/>

Iker.landajuela, D. (2016, December 20). Arduino: Sintaxis básica del lenguaje de programación. Recuperado el 18 de abril, 2019, de  <https://ikerlandajuela.wordpress.com/2016/12/20/arduino-sintaxis-basica-del-lenguaje-de-programacion/>

**III Proyecto**

# Fuentes consultadas

*Devcode*. (24 de Mayo de 2019). Obtenido de Devcode Diccionarios en Python: https://devcode.la/tutoriales/diccionarios-en-python/

*Effbot*. (Junio de 2019). Obtenido de Effbot: http://effbot.org/tkinterbook/entry.htm

*likegeeks*. (18 de Mayo de 2019). Obtenido de likegeeks : https://likegeeks.com/es/ejemplos-de-la-gui-de-python/#Adicionar-un-dialogo-para-archivos-elegir-archivo-y-directorio

*Programiz*. (26 de Mayo de 2019). Obtenido de Programiz Python programming: https://www.programiz.com/python-programming/methods/list/insert

*Stackoverflow*. (21 de Mayo de 2019). Obtenido de Stackoverflow: https://stackoverflow.com/questions/45467143/disable-exit-or-x-in-tkinter-window

*Stackoverflow*. (23 de Mayo de 2019). Obtenido de Stackoverflow: https://stackoverflow.com/questions/6920302/how-to-pass-arguments-to-a-button-command-in-tkinter

*Stackoverflow*. (24 de Mayo de 2019). Obtenido de Stackoverflow disable exit or x in tkinter window: https://stackoverflow.com/questions/45467143/disable-exit-or-x-in-tkinter-window

*YouTube*. (22 de mayo de 2019). Obtenido de YouTube: https://www.youtube.com/watch?v=Y5EMGBo39Kk

The Tkinter Radiobutton Widget. (s.f.). Recuperado el 19 de mayo, 2019, de <http://effbot.org/tkinterbook/radiobutton.htm>

Tutorialspoint.com. (s.f.). Python Tkinter Entry. Recuperado el 20 de mayo, 2019, de https://www.tutorialspoint.com/python/tk\_entry.htm