**Instituto Tecnológico de Costa Rica**

**Ingeniería en Computadores**

**Taller de Programación**

Project II (Part 2):

Formula E CE TEC

Energy Saving and Telemetry Part I

**Estudiantes:**

**José Fernando Morales Vargas**

**Carné: 2019024270**

**Alejandro José Quesada Calderón**

**Carné: 2019150208**

**Profesor: Milton Villegas Lemus**

**Grupo 2**

**I Semestre**

**2019**

**Bitácora del Tercer Proyecto, Alejandro Quesada.**

**Viernes, 17/05/2019.**

**Administrativa:** El día de hoy, los integrantes del equipo nos sentamos a definir los roles que tendríamos a lo largo del desarrollo de esta fase del proyecto, el plan al hacer esto es que podamos tener un plan de trabajo bien definido desde el inicio y tengamos la posibilidad de rectificar la calidad de nuestro trabajo con el profesor encargado del proyecto.

**Técnica:** Al definir los roles, acordamos que me corresponderían las tareas relacionadas al *Front-End developer*: Documentación del código, verificación de calidad (en general) y ser *Back-End support.* Mientras se asignaron estos roles, entre mi compañero y yo también acordamos que su rol sería el *Back-End developer*: correspondiente a la coordinación del trabajo y dar soporte en la documentación. Cabe destacar que la definición de estos roles es el paso inicial del trabajo y pueden llegar a cambiar según avancemos en el desarrollo del trabajo.

**Sábado, 18/05/2019.**

**Administrativa:** El compañero José se encargó de la organización inicial del trabajo antes de comenzar el desarrollo.

**Técnica:** Hoy se creó el repositorio en GitHub en el cual se contendrá todo lo necesario para el funcionamiento del proyecto a medida que avancemos en este. El repositorio consiste de tres ramas, la principal, y una rama para cada desarrollador, inicialmente, estas subramas son una copia de la principal en la que cada uno tendrá su espacio para trabajar y guardar las modificaciones del repositorio; más adelante (cuando sea necesario) utilizaremos la función *merge* de git para actualizar el trabajo de cada uno respecto a la rama del otro integrante del equipo. Para esto y durante el resto del trabajo, estamos utilizando la versión de escritorio de Git, *GitHub Desktop*, la consideramos sumamente útil ya que agiliza el trabajo y simplifica el entendimiento de cómo funciona Git a comparación de la consola base de esta herramienta. Igualmente, los conocimientos adquiridos en el Taller de Git dado en tiempo lectivo fue de ayuda para comprender mejor el funcionamiento de esta herramienta.

Luego de que José creara el repositorio, me dediqué a crear la base de la interfaz para este proyecto. Para esta tarea, utilicé la biblioteca Tkinter de Python, el entorno de desarrollo integrado de Python y la versión 3.7.2 del lenguaje. Gracias a que realizamos un proyecto con estas mismas herramientas anteriormente, pensé que sería fácil adaptar los conocimientos previos a las necesidades de este trabajo.

Hasta el punto de desarrollo actual, logré que existan las ventanas solicitadas en el pdf del proyecto, y que pueda haber una comunicación entre ellas, básicamente basándome en comandos de retorno que destruyen una ventana y muestran la ventana “destino”.

**Extensión de la Administrativa:** Hoy logré completar esta tarea de generar el código base para la interfaz y lo subí al repositorio de GitHub; creí que iba a ser una tarea bastante sencilla, pero noté que (por el tiempo que ha pasado desde que realizamos el primer proyecto, y lo poco que logré completar este mismo) tuve que investigar un poco sobre la sintaxis de Tkinter (sus funciones y comandos). Para esto fue útil volver a consultar el sitio effbot.org/tkinterbook.

**Lunes, 20/05/2019**

**Administrativa:** El día de hoy no hubo avances **significativos** en el proyecto de mi parte. Esta semana estaré ocupado por quices, tareas y otras evaluaciones, así que probablemente no pueda avanzar en el proyecto hasta el fin de semana.

**Técnica:** Modifiqué el archivo del código que se tenía para las ventanas, a este le añadí una versión muy preliminar de los contenidos de la ventana de about o créditos, para la configuración de esta utilicé una función de creación de imágenes que nos había cedido el asistente Santiago para el primer proyecto, y al haber utilizado su TelemetryLog en el proyecto pasado, encontré una herramienta interesante para el display de texto, que es el ScrolledText; esta permite insertar un texto amplio en un espacio pequeño, lo cual será útil para aprovechar más el espacio disponible en las ventanas. Para la ventana about, también se creó una imagen de prueba que funge simplemente como *place-holder* para cuando incluyamos las imágenes de los autores del trabajo. Sobre el trabajo en general, también se modificaron las dimensiones de las ventanas de la interfaz. Para esto fue útil una herramienta de la cual desconocía en el primer proyecto, la window.geometry, esta permite insertar un string con las dimensiones de la ventana de la forma (“1280x720”) lo que la hace más fácil de configurar, y también vuelve más legible este parámetro, dado que es más notable que corresponde a una relación de aspecto.

**Sábado, 25/05/2019**

**General:** Hoy no trabajé en profundidad en mis tareas del trabajo. Los compañeros de clase David Solís y Miguel Mesén (quienes conforman otro equipo) van a reunirse hoy para avanzar en su propio trabajo; me comuniqué con ellos para aprovechar dicha reunión y estar presente en caso de que necesitara ayuda con el desarrollo de mi propio trabajo, ya que hasta la fecha, David ha avanzado bastante justo en lo que me corresponde desarrollar, por lo cual tenerlo a la mano en caso de necesitar sus fuentes de investigación o un poco de orientación con la lógica operacional del Test Drive.

**Técnica:** Avancé la parte de la interfaz que corresponde a la ventana Test Drive, pero para lograrlo, necesitaba tener los archivos Python dados por Santi en el segundo proyecto, el TelemetryLog y el WiFiClient, ya que estos son cruciales para comunicar la parte programada con el hardware en el proyecto. Al copiar estos archivos y ponerlos dentro del mismo dominio del código para la interfaz, fui capaz de importar algunos métodos de estos, como la función send.

Investigué sobre cómo utilizar la misma función para devolverse de una ventana a la principal, utilizando esta ventana como argumentos, ya que para simplificar el código había intentado simplemente enviar el comando de la ventana como un argumento; pero al intentarlo noté que había un error de código. Luego de investigar, aprendí que esto se puede realizar por medio de expresión lambda, al consultar en YouTube sobre cómo cambia la funcionalidad de un comando o de un proceso (en comparación con situaciones que no requieran utilizar esta expresión).

Luego de realizar este cambio en la utilización de un botón de retorno, me dediqué a buscar imágenes para llevar a cabo el desarrollo de la ventana Test Drive justo como quería realizarla. Para esto, era necesario utilizar imágenes en formato png o Portable Network Graphics, dado que, si se quiere utilizar una imagen con fondo transparente, deben crearse justo encima del canvas (o sea, sin utilizar un label para colocar la imagen sobre este) y también que la imagen por sí misma no posea un color de fondo.

**Domingo, 26/05/2019**

**Administrativa:** Aún me encuentro en el domicilio del compañero David, cada uno está enfocado en sus propias tareas y sólo nos consultamos en caso de necesitar despejar alguna duda sobre el proyecto.

**Técnica:** Aún me encuentro trabajando en dos enfoques principales de la ventana Test Drive: programar los aspectos funcionales para los eventos en el teclado, y también ir adaptando la interfaz gráfica a lo que se desea obtener al final de esta etapa. Respecto a este último punto, estuve buscando imágenes que coincidan más con lo que buscamos para el trabajo; es por esto por lo que algunas imágenes hoy fueron eliminadas y otras fueron añadidas a la carpeta en el repositorio.

También, estuve trabajando en la parte lógica específicamente para los comandos de dirección en el vehículo. Para esto, modifiqué la función *send*() de TelemetryLog, y esta versión modificada la incluí en el código de nuestro trabajo. Esto porque es más conveniente utilizar simplemente esta función para hacer llegar un comando al NodeMCU, que llamar a la función del archivo cada vez que fuera necesario.

Para el manejo de eventos en este proceso, se está utilizando un *keybinding* en la ventana Test Drive, estos binds se asignan a funciones que se activan cuando existe un evento en el teclado; para esta tarea en específico se está trabajando con KeyPress y KeyRelease, por lo que las funciones sólo se activan con eventos de presión de una tecla, y la liberación de esta misma.

Para controlar los eventos con más facilidad, se añadieron variables globales para manejar estos eventos; hasta ahora con los que se ha trabajado es con los eventos de movimiento. Para estos eventos, no basta con llevar un control de las teclas, ya que puede haber varias generando eventos al mismo tiempo; por lo cual recurrí a la implementación de hilos y así ser capaz de ejecutar subprocesos paralelamente sin depender de la ejecución secuencial del código. Para ayudarme con estos eventos, consulté de nuevo la página effbot.org esta vez en el directorio de “Events and Bindings” https://effbot.org/tkinterbook/tkinter-events-and-bindings.htm

También se añadió texto al canvas con un formato que no había utilizado anteriormente, consiste en generar el código sobre el canvas sin la necesidad de utilizar un label como contenedor para el texto. En el canvas de la ventana principal, se añadieron textos que serán place-holders para la información de los pilotos, cuando eventualmente se incluya ese módulo.

Consulté con el asistente del curso, Santiago Gamboa, sobre algunos aspectos de las especificaciones del proyecto que no me quedaban del todo claro, por el momento lo que logré aclarar es sobre las luces direccionales, creí que estas debían encenderse automáticamente cuando el carro toma una dirección; Santiago me explicó que estas no debían complicarse tanto, simplemente debía asignársele un botón a las luces para que estas pudieran encenderse independientemente. Este cambio se está implementando actualmente en la lógica del trabajo.

**Lunes, 27/05/2019**

**Administrativa:**  Hoy fui capaz de trabajar en el proyecto hasta horas de la tarde porque tenía clases en la mañana.

**Técnica:** Estuve teniendo problemas para validar casos especiales en la utilización de las teclas para el control del carro: estas estaban funcionando parcialmente, debido a que si se ejecutan una vez se comportan de la forma esperada, pero mantener la tecla presionada estaba generando múltiples eventos y alterando el funcionamiento del código, para solucionar este problema tuve que dedicarle bastante tiempo a replantear lo ya programado antes de seguir con los próximos pasos; tuve que añadir nuevas variables para forzar al sistema a trabajar con una única presión de la tecla: ahora se comportan de modo que para repetir un proceso, se debe liberar la tecla primero. Luego de corregir estos errores, procedí a centrarme en los eventos faltantes, ya que por el momento la funcionalidad del Test Drive (respecto al control del hardware) está cerca de completarse; sin embargo, aún no he implementado la operación con imágenes para dar la retroalimentación al usuario y que este sea capaz de notar los elementos del auto que están en funcionamiento a medida que lo vaya manipulando.

Hoy el compañero José y yo actualizamos nuestras ramas para que estén las 3 sincronizadas, hicimos un merge a la rama principal y cada uno actualizó su rama correspondientemente. Estar trabajando cada uno con su rama ha sido útil para llevar los archivos organizados y evitar conflictos.

Continué trabajando en la lógica del programa (respecto al Test Drive), en la noche logré que las direccionales funcionen completamente en términos operacionales, sin embargo, aún está pendiente la parte estética/gráfica de estas funciones.

Estuve editando la imagen que cumple como *layout* del auto (o sky view) pues esta tenía un aspecto extraño de tener una sombra a un lado del carro, para todas estas ediciones estuve utilizando una versión online no oficial de Photoshop y me ha sido de gran ayuda para preparar las imágenes para esta ventana.

**Martes, 28/05/2019**

**Administrativa:** Hoy y los próximos días se me dificultará trabajar en el proyecto debido a las clases y asignaciones que les corresponden. Específicamente hoy, logré dedicarle tiempo hasta en la tarde.

**Técnica:** Hoy requerí detener el avance del proyecto para corregir una implementación errónea en la lógica del código respecto a la aceleración y desaceleración del auto: como en los días pasados había tenido conflictos con las luces direccionales y su comportamiento, supuse que la aceleración del auto debía acercarse del mismo modo a la de un auto de verdad; tal que, si se libera el acelerador, el auto debe ir disminuyendo la velocidad poco a poco. Consulté sobre la situación con Santiago, quien me orientó y me explicó el comportamiento esperado para este aspecto (cito textualmente): “Cada vez que yo presiono el acelerador, él manda pwm:algo; eso lo ejecuta una vez, cuando yo suelto la tecla, el carro deja de recibir pwm:algo y ahí se queda”.

Debido a la explicación anterior, tuve que hacer un cambio mayor en la lógica de movimiento, tenía un hilo que se activaba cuando ni la W ni la S (Que son parte de nuestras 4 teclas de movimiento, W-A-S-D) se están presionando, y este aproximaba la velocidad hasta que fuera 0. La corrección dejó un nuevo comportamiento, que cumple con las especificaciones dadas por Santi.

Luego de corregir este problema, trabajé en las imágenes para las direccionales y las luces frontales con el objetivo de que estas aparezcan en pantalla, y al dejarlas preparadas, se guardaron en el repositorio.

Santi más tarde me recomendó que adaptara mi lógica para que no mande valores de pwm pequeños (aproximados a 0 entre 400 y -400) para evitar que el auto se quedara estático emitiendo pitidos, y adapté los hilos, de forma que dentro del while principal del hilo, existe un while secundario que controla este rango de valores, y no es hasta que se supera la condición de operación de este loop que se empiezan a enviar los valores al auto. Como llevo poco tiempo practicando con estos loops, las condiciones parecen no estar comportándose explícitamente como se escriben, ya que si le determino al while que se mantenga activo mientras la potencia sea menor que 1000, el hilo termina superando este while y enviando un comando de más. Estos errores en las condicionales de los loops deben corregirse más adelante, por ahora la prioridad es llegar todo a un estado funcional.

**Jueves, 30/05/2019**

**Administrativa:** Estamos en los últimos días de clases, aproveché la tarde y me fui a trabajar al LAIMI 1 para recrear la ventana Test Drive.

**Técnica:** Hoy me dispuse a verificar si el código estaba siendo 100% funcional, para esto tomé el código ya existente para la ventana Test Drive, lo copié en un nuevo archivo de Python, y lo trabajé por fuera del repositorio; esto con el objetivo de depurar el código obtenido hasta el momento, y mejorar la funcionalidad del archivo. Este nuevo archivo se llamó *NewGUI*, básicamente una versión mejorada de la interfaz gráfica. Se avanzó en funciones de movimiento, encendido y apagado de luces para el auto, y estas son completamente operacionales. Respecto a la estética sólo faltan dos aspectos: la aparición de las luces traseras y de las ruedas delanteras al virar. Aún no existe un hilo o función que se encargue de operar las luces traseras, pero este se trabajará más adelante.

También se corrigió un problema con la implementación del parpadeo de las luces direccionales: estas se habían programado para funcionar exclusivamente, por lo que, si el usuario presionaba la tecla que enciende una luz e inicia el loop iterativo de su parpadeo, y posteriormente encendía la otra luz, la más reciente cancelaba la primera en activarse (con tal de que solo existiera una parpadeando a la vez). Por sugerencia y orientación de Santiago, esto se corrigió para que las luces sean independientes, y no sólo puedan encenderse sin estar moviendo el carro, si no que también puedan activarse simultáneamente. Para esto, decidí que las luces mantendrían el mismo binding en el teclado, la Z para encender la izquierda, la C para encender la derecha, y la X para apagar cualquiera de ambas que esté activa al momento de presionar dicha tecla.

Todos los cambios que se hicieron el día de hoy se estuvieron trabajando en el archivo NewGUI.py, sin embargo, por la tarde se tomó el código de este archivo, y se creó un nuevo archivo en el que se trabajó este código, nuevamente para mejorarlo, todos los cambios que se le hicieron al código se guardaron en este archivo con el nombre de DebuggedGUIv2. Hasta ahora, ha sido bastante útil trabajar los hilos con iteración, si se compara a la recursividad implementada en el primer proyecto, es bastante más simple.

Además, se revisó un problema menor que existía con el código NodeMCU, dado que este no se había utilizado hasta la fecha, por lo cual no se encontraba dentro de una carpeta con su mismo nombre (lo cual fue un problema debido que el Arduino IDE requiere que se encuentren los archivos en este formato). Se añadió el archivo a una carpeta con el mismo nombre y se solucionó el problema.

**Viernes, 31/05/2019**

**Administrativa:** Hoy fue el último día de clases, pude dedicarle más tiempo al trabajo por esta razón. Mi compañero (y los estándares de entrega que definimos anteriormente) requerían que la funcionalidad del Test Drive estuviera preparada para hace un par de días, por motivos de atraso aún falta que funcionen completamente.

Me quedé trabajando con el código, en conjunto con el auto, hasta las 5 de la mañana del sábado.

**Técnica:** La ventana de Test Drive aún se está trabajando en un archivo ajeno al principal, el DebuggedGUIv2, las tareas faltantes para la funcionalidad mínima del Test Drive son las lecturas de los sensores del auto: la fotorresistencia y la batería. Los datos de estos sensores debían ser incluidos en esta ventana, pero aún no había trabajado en la forma de leer estos. Hoy me dediqué a estudiar los códigos brindados por Santiago, gracias a una recomendación del compañero Kevin Masís, busqué una función en el WiFiClient que se encarga de guardar los comandos (junto con la respuesta que reciben del NodeMCU) en una lista, esto facilita bastante la manipulación de la respuesta del comando *sense;* y llevar esto a cabo es necesario para obtener los datos de los sensores. Para esto, validé con el largo de la respuesta obtenida, que si superaba una cantidad de caracteres (15) sería la respuesta del comando sense; sin embargo, por las modificaciones realizadas al código base en la etapa anterior del proyecto, varias respuestas podían llegar a tener esta dimensión, por lo cual me di la tarea de modificar el archivo NodeMCU.ino y darle mensajes de retorno más cortos al resto de comandos, con el fin de que solamente el comando sense fuera tomado en cuenta para esta tarea. También, en este archivo, reemplacé la lógica de sensor de la batería, ya que esta no era la más precisa. Para esto, consulté la página Arduino Reference sobre maneras de convertir un valor de entrada (en este caso PWM) en otro margen de salida (como el deseado, de 0 a 100). Encontré el comando map, y lo implementé en el código de Arduino para el NodeMCU (este cambio fue documentado por consulta con Santiago, quien recomendó que toda modificación a este debía documentarse).

Una vez que el código en el NodeMCU estaba listo y cumplía con lo que necesitaba para manipular el comando *sense;* dentro del código de Python generé una variable en la que se guardaría la respuesta obtenida por el comando; modifiqué la función get\_log para que retornara la respuesta sin la palabra result, así, esta respuesta se enviaría como argumento para dos hilos: uno que controla lo recibido por el LDR; y el otro del sensor de baterías. Estos hilos funcionan con los índices del string obtenido en la respuesta, mientras que el índice del valor retornado por la fotorresistencia tiene una posición fija ([-4]), los dígitos del sensor de batería podían variar dependiendo de su cantidad (1 si es de 0-9, 2 de 10-99, y 3 si lee 100) por lo cual esta requirió más validación. Estas funciones van en conjunto con otro hilo que se ejecuta desde que la ventana se abre, este lo que hace es enviar el comando sense cada cierta cantidad de segundos, utilizando un delay (time.sleep()) para poder realizarlo.

Los aspectos faltantes del test drive corresponden a la obtención de los datos de los pilotos, del test drive se podrá migrar el sensor de batería para usarlo como el estado actual del carro.

**Sábado, 01/06/2019**

**Administrativa:** Hoy me levanté tarde porque me quedé hasta la mañana trabajando en el Test Drive, por lo cual no realicé tareas del proyecto hasta la tarde.

**Técnica:** La mayoría de los aspectos que me corresponden sobre el código que me corresponden están listos, por lo que hoy me dispuse a trabajar en la documentación. Me faltaba añadir detalles a mi bitácora, ya que en esta la parte técnica estaba poco desarrollada, para esto me ha sido muy útil la posibilidad de añadir descripciones a cada commit realizado al repositorio, ya que en este me basé para ir trazando cada avance y verificar que no dejara ningún detalle de lado. Mi compañero José se está encargando de integrar sus métodos para el manejo de pilotos con la interfaz que había creado hasta el momento, hay algunas tareas pendientes sobre la parte programada, pero iré avanzando la documentación mientras él se encarga de integrar ambas partes.

Luego de un rato, José me estuvo corrigiendo sobre algunas malas prácticas que tenía, me hizo saber que no era necesario tener todas las variables declaradas como globales, si no utilizarlas dentro de cada función como no-locales (*nonlocal variables*). Otro error bastante importante es que yo utilizo mainloop() en cada ventana para que los aspectos de estas ventanas funcionen, mi compañero me informó que la forma correcta de cargar elementos a una ventana es crearlos como propiedad del canvas de esa ventana, lo que quita la necesidad de generar un loop para cada ventana. Me dijo que lo malo de esto es que el código puede generar bugs y se pone en riesgo el funcionamiento del trabajo. Estas fallas están siendo corregidas, se está mejorando el código mientras avanzamos en áreas diferentes.

**Domingo, 02/06/2019**

**Administrativa:** Me fui en la mañana al apartamento de un compañero (Brian W) por un motivo de que trabajo mejor en compañía que estando sólo en mi departamento.

**Técnica:** Aún me encuentro trabajando en la documentación de mi parte del trabajo. Más adelante debo trabajar en algunas partes faltantes de la parte programada, y cuando terminemos el proyecto, trabajaremos en conjunto para terminar el documento administrativo de este.

# Introducción:

El trabajo realizado en esta fase consiste en la Parte II (final) del proyecto. En el trabajo pasado, se requería que el equipo de trabajo realizara modificaciones a un carro a control remoto, para que este se asemejara más a un carro de verdad (incluyendo en sus funcionalidades) y que además pudiera ser controlado por internet. Para esa etapa, se utilizaron diversos componentes electrónicos (todos mencionados en la documentación de la parte I), entre los que destaca el NodeMCU v3 de Lolin, el cual posibilita esta comunicación inalámbrica.

Fue necesario mencionar el producto de la fase anterior a la presente, para explicar dicho trabajo, puesto que, al ser dos partes del mismo proyecto, el producto final de la primera se necesita para el desarrollo de la etapa final. Esta etapa, consiste en crear una versión mejorada del control remoto “TelemetryLog” de Python; incluir las mismas funciones de este, y, además, generar una interfaz gráfica en la que el usuario del auto pueda ver el estado del carro. En sí, la mejora del control del carro sólo es parte de lo que debía realizarse, ya que también debe añadirse a toda esta interfaz una ventana de posiciones en las que se guarda el rendimiento de los pilotos, una para llevar un registro de los autos de la escudería, y la información sobre los patrocinadores, junto con el logo de la empresa.

Para estas tareas, se está usando como herramientas la biblioteca Tkinter de Python, con dicho lenguaje en su versión 3.7.2, y GitHub (en su versión para escritorio) como repositorio remoto/ herramienta para control de versiones. De la biblioteca Tkinter se utilizan los métodos para generar ventanas, y en estas se trabaja con lienzos en los que se insertan objetos, en conjunto con hilos y funciones iterativas para el manejo de estos hilos. En profundidad sobre la ventana de test drive, se están controlando los eventos del carro con binds en las teclas, que consisten en los eventos *KeyPress y KeyRelease*, cada tecla de las utilizadas se controla en una función y dependiendo de su aplicación, invocan un hilo, o simplemente cambian el valor de banderas que mantienen activos otros hilos. Algunos ejemplos de esto son la tecla W (aceleración) y la X (apagado de las direccionales); el primer ejemplo invoca un hilo al presionarse, y el segundo funciona para cambiar las banderas mencionadas. Estas banderas se trabajan con variables no-locales y globales.

Para comunicar el módulo de software con el hardware aún se está utilizando el archivo WiFiClient.py dado por Santiago, sin embargo, se ha descartado el uso de TelemetryLog como parte del módulo, simplemente se utilizó como fuente de información y para modificar sus funciones (e implementar estas al código desarrollado por los integrantes del grupo).

Para la parte gráfica del Test Drive, se decidió simular el Heads-Up Display de un videojuego, por esto existe tanto una visión en primera persona, como una perspectiva aérea del carro en la que se puede ver el estado de este (en términos de luces y direcciones). Como decisión de desarrollo, el valor de la potencia se está dando de forma porcentual para hacerlo más legible.

***\*Se debe añadir la sección del análisis en la que se discute lo logrado del proyecto. Es mejor realizar esta parte hasta haber completado el trabajo\****.

# Conclusiones:

* Definir los roles y tareas para los integrantes del proyecto en las fechas iniciales del periodo de trabajo fue útil para la organización de este.
* Organizar el repositorio en GitHub antes de empezar el trabajo digital ayudó con la modularización del proyecto.
* Los conocimientos adquiridos en el taller de Git aumentaron el uso de todas las herramientas prestadas por Git al usuario.
* La utilización de Tkinter como herramienta para interfaz gráfica fue buena opción considerando su manipulación en trabajos anteriores.
* Generar las ventanas necesarias y que se pueda comunicar entre ellas antes de trabajarlas individualmente ayuda a organizar el trabajo con la interfaz gráfica.
* Estudiar los códigos prestados por el asistente del curso fue necesario para entender su funcionamiento.
* Del estudio de estos códigos se pueden descubrir
* La herramienta ScrolledText de Tkinter es muy útil si se desea incluir un texto espacioso en una ventana compacta.
* Colocar imágenes de prueba para que tomen el espacio del objeto final en la ventana About fue útil para avanzar sin dejar la ventana completamente sin manipular.
* La función geometry() fue útil para manejar las dimensiones de la ventanas con más facilidad, igualmente para que este parámetro sea más legible.
* Importar el WiFiClient.py del asistente del curso es mandatorio para el funcionamiento de este proyecto.
* Definir un solo comando de retroceso, y asignarle este a todos los botones de retorno, simplificó el código y lo optimizó.
* Aprender sobre la expresión lambda y su utilidad para “funciones anónimas” fue necesario para que se pueda utilizar un botón con argumentos.
* Avanzar paralelamente la parte estética y la lógica de la ventana Test Drive fomentó la creatividad y mantuvo una buena estructuración de dicha ventana.
* El uso de imágenes en formato PNG fue necesario para poderlas manejar con fondos transparentes y mantener una buena calidad de imagen.
* Revisar bien las funciones que se toman del código TelemetryLog o WiFiClient es conveniente, pues algunas tienen implementaciones lógicas que ya no son necesarias y se pueden modificar.
* Utilizar eventos de “keybinding” para el control del carro fue una práctica bastante útil, aún más el haberla implementado con funciones separadas según el tipo de evento (Press/Release).
* Manejar los eventos en conjunto con variables globales (o también, con no-locales) facilita el control de casos especiales/específicos.
* Consultar en internet para verificar la sintaxis o aplicación de ciertos métodos, fue útil debido al tiempo que pasó desde la última vez que se utilizó Tkinter en el curso.
* La función canvas.create\_text() fue de gran utilidad para mantener un canvas limpio, dado que este elimina la necesidad de un label para contener el texto.
* Consultar con el asistente del curso sobre divergencias entre los desarrolladores y las especificaciones del proyecto fue necesario a lo largo del desarrollo, para asegurar un cumplimiento de dichas especificaciones.
* La validación de casos especiales en los eventos fue más complicada de lo que parecía en un inicio al planearse.
* Mejorar los conocimientos necesarios para aplicar la lógica correcta en Tkinter fue necesario debido al poco manejo de esta herramienta que se tenía.
* Tener una rama en Git para cada integrante del equipo ayudó a la hora de evitar conflictos, ya que así se manejan archivos y espacios virtualmente distintos.
* Trabajar con mayor prioridad en la parte funcional de Test Drive antes de la gráfica (sin dejar esta última de lado) fue importante para un decente manejo del tiempo de trabajo.
* Utilizar una versión online de Photoshop fue necesario para adaptar las imágenes al formato que se necesitó en el desarrollo, sin tener que descargar dicha aplicación.
* Detener el avance de las tareas cuando la lógica de las direccionales se implementó sin errores (pero diferente del comportamiento solicitado), fue bueno para evitar que las equivocaciones permanecieran.
* La utilización de iteración dentro de las funciones invocadas por los hilos facilitó la validación de casos con las variables nonlocal/global.
* Tomar el módulo de test drive y mejorarlo en un archivo ajeno al repositorio en el archivo NewGUI (y posteriormente DebuggedGUIv2) fue crucial para mejorar la funcionalidad de este.
* Revisar la funcionalidad del código de Arduino para el NodeMCU fue necesario para avanzar en las tareas asignadas.
* Manejar las respuestas del NodeMCU en Python con la lista “Log” de WiFiClient fue útil, ya que validar estas con el largo de sus elementos facilitó la implementación de los sensores del auto.
* Documentar los cambios realizados al NodeMCU se considera una buena práctica para evitar problemas con los encargados del curso.
* Se aprendió (por parte de uno de los integrantes del grupo) que el uso de mainloop en todas las ventanas, así como de globales se corrige con uso de *nonlocal* y al implementar las imágenes y botones como propiedades de la ventana a trabajar.

# Recomendaciones:

* Empezar el trabajo por la declaración de tareas y roles, así se ahorra tiempo y evita desorden, de lo contrario podría sufrir conflictos y otras complicaciones entre los integrantes.
* No empezar a trabajar los módulos de software sin haber organizado un repositorio. Hacer caso omiso puede llevar a pérdida de proceso y a una mala modularización.
* Utilizar los conocimientos del Taller de Git (si se le da uno en clase) para aprovechar las herramientas que brinda GitHub al usuario.
* Si anteriormente utilizó Tkinter y se familiariza con la biblioteca, apéguese a esta en vez de buscar alternativas, se ahorra tiempo en investigación y se tiene mejor manejados los conocimientos.
* Al trabajar en la interfaz, generar todas las ventanas especificadas y poder comunicarlas antes de dedicarles tiempo individualmente le ayudará a llevar el trabajo mejor organizado. No hacerlo podría hacer que omita alguna especificación del trabajo.
* Si el asistente de curso presta archivos para el manejo de la comunicación, es altamente útil estudiar estos, así puede decidir qué funciones se importan a su código o qué utilizar de estos, igual si decidiera modificarlos. Omitir esto complica mucho las tareas del proyecto.
* Si desea introducir un texto amplio en alguna ventana, la herramienta ScrolledText puede serle útil. No utilizar esta o alguna similar, implica que deberá cambiar el tamaño o geometría de dicha ventana.
* Si no va a trabajar profundamente una ventana, pero esta contiene imágenes que requiere colocar estratégicamente (como el caso de la de créditos), utilice imágenes de prueba para calcular estos espacios, así evita dejar la ventana sin trabajar del todo mientras avanza alguna otra tarea.
* Si desea declarar alguna dimensión definida para una ventana, utilizar la función geometry(“XxY”) para especificar estas dimensiones es una mejor opción que declararlas por separado, así le ayudará a comprender más qué se está definiendo.
* Debe importar el WiFiClient.py o utilizar este en el mismo módulo de su programa para comunicarse correctamente con el hardware, de lo contrario, deberá generar su propio método de conexión y le dificultará mucho el desarrollo del trabajo.
* Si va a colocar un botón de retroceso en todas o varias de sus ventanas, utilizar una misma función como comando para todas estas es la mejor opción, de lo contrario necesitará uno por ventana, esto le tomará más tiempo y recursos.
* Para manejar comandos con argumentos se debe utilizar la expresión *lambda*, de lo contrario, no podrá reutilizar segmentos de su código como comandos con argumentos.
* No avanzar demasiado la parte lógica de la interfaz de manejo dejando de lado la estética (o viceversa), este modus operandi le llevará a una mala estructuración de la ventana, o hasta olvidar funciones que debe cumplir.
* Si desea generar una ventana limpia y atractiva al usuario, utilice el formato PNG para sus imágenes, así mantendrá una buena calidad y podrá implementarlas con fondos transparentes. Para esto quizá necesite investigar sobre editores gráficos (depende de su conocimiento sobre el tema)
* Si toma funciones de los códigos prestados por el asistente, evalúe si toda la lógica implementada es útil para sus propósitos, así puede descartar lo que no sea necesario. No hacer esto puede llevar a incluir código que no es necesario, por ende, gastar más recursos computacionales.
* Si desea controlar el hardware por medio del teclado, utilizar los bindings KeyPress/Release es una excelente opción, y también es bueno trabajar cada evento/mapeo con una función por separado.
* Para tener un mejor control de la inicialización y detención de un proceso, se facilita con el uso de variables globales o no locales. Esto le ahorrará tiempo, y volverá el código menos propenso a fallar por casos especiales.
* Por más que confíe en sus conocimientos sobre la herramienta de interfaz que esté utilizando, siempre es bueno verificar la implementación de algunos métodos dudosos en páginas o libros de referencia.
* Para tener un canvas bien ordenado y atractivo, si busca implementar texto en este, puede evitar la necesidad de un label con la función créate\_text(), eso genera el texto directamente sobre el canvas, tomando menos espacio.
* Si tiene alguna duda sobre las especificaciones del proyecto, o cree que está implementando algo de manera errónea, siempre es buena idea consultar con el asistente del curso, así puede evitar llegar a discusiones con los encargados, y eventualmente perder puntos en la rúbrica del proyecto.
* Sentarse a investigar más sobre la herramienta en uso (por ejemplo, Tkinter) si no manejaba bien esta antes de iniciar sus tareas.
* Si está trabajando con otra persona o varias, tener una rama en el repositorio para cada integrante le ayudará a evitar conflictos entre cargas de cambios al repositorio remoto, y a la larga le ayudará a llevar un mejor control del avance.
* Dar prioridad al trabajo en la lógica/funcionalidad del código antes de la parte gráfica es un buen camino que seguir si desea tener un buen manejo del tiempo, sin embargo, no es bueno dejar de lado la parte gráfica completamente.
* Para adaptar las imágenes que desee usar al formato recomendado, puede utilizar editores en línea, tales como Online Photoshop (ver fuentes consultadas), así evita tener que descargar programas de este tipo.
* Si se encuentra con que una de sus aplicaciones no falla, sin embargo, esta no se comporta de la manera esperada, es mejor detener el avance en las demás tareas y reparar esta, ya que, si continúa con el trabajo, puede que olvide solucionar ese problema y termine arriesgando puntos en las evaluaciones de su producto.
* Ya que se permite el uso de la iteración, es bueno dar uso de estos métodos en el manejo de hilos que deben ejecutarse siempre que algún evento se esté dando; facilita la validación de casos al igual que detener estos mismos hilos con las condiciones implantadas en los loops.
* Si ve necesario cambiar algo muy significativo (o varias líneas de código) en alguna función o ventana específica, es una buena idea tomar esta función y manipularla en un archivo por aparte, luego cuando se sepa que los cambios realizados funcionan, implementarlos al repositorio en Git; esto puede evitarle pérdidas de código o progreso y tener que revertir los cambios con un commit anterior.
* Revisar la capacidad de ser operacional y el estado del código para el NodeMCU si va a hacer utilidad de este, ya que puede que este no cumpla con sus necesidades y le lleve a atrasarse con otras tareas.
* Para manejar las respuestas que Python obtiene de los comandos enviados al Node, es óptimo utilizar la función get\_log() del WiFiClient, pues esta retorna la respuesta que se requiere manipular en una lista, esto implica que se puede tratar con índices y posiciones, y le ahorrará problemas el apegarse a este método.
* Si realiza algún cambio que vea necesario al código de Arduino o al hardware en sí, es mejor documentar este, para así evitar tener problemas con los encargados a evaluarle. No seguir este paso puede llevarle a perder puntos en la calificación.
* Si desconoce de algún tema o de las implementaciones generales de una herramienta como Tkinter y otro integrante de su equipo es más diestro en ese campo, solicítele asesoramiento para asegurarse que todos están al tanto de qué prácticas conforman el conjunto de *buenas* prácticas, y cuáles deben evitarse.

**Fuentes Consultadas:**

*Arduino Reference. (2019, Mayo, 31). Retrieved from* [*https://www.arduino.cc/reference/en/*](https://www.arduino.cc/reference/en/)

*Free Phototool, Retrieved May 28th, 2019 from* [*https://www.freephototool.com/*](https://www.freephototool.com/)

*Tkinter Events and Bindings, Retrieved May 26th, 2019 from* [*https://effbot.org/tkinterbook/tkinter-events-and-bindings.htm*](https://effbot.org/tkinterbook/tkinter-events-and-bindings.htm)

*How to pass arguments to a tkinter command, retrieved May 27th , 2019, from* [*https://www.delftstack.com/howto/python-tkinter/how-to-pass-arguments-to-tkinter-button-command/*](https://www.delftstack.com/howto/python-tkinter/how-to-pass-arguments-to-tkinter-button-command/)

*Lambda Expressions, consulted May 27th , 2019 at Socratica,* [*https://www.youtube.com/watch?v=25ovCm9jKfA&t*](https://www.youtube.com/watch?v=25ovCm9jKfA&t)

*Global, local, nonlocal variables, Programiz, retrieved June 1st, 2019 from* <https://www.programiz.com/python-programming/global-local-nonlocal-variables>