**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 trabajo 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.

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

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.

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

**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.