Punto de Partida

El objetivo de este texto es tratar de analizar el trabajo que fue hecho en conjunto con el alcance propuesto del proyecto comenzado en el LAC, y a partir de allí definir un alcance y objetivos claros. En base a lo charlado la idea es definir una/s problemáticas claras que puedan ser definidas y resueltas en un tiempo prudencial. Con este fin pensé un análisis en 3 etapas:

- Análisis del proyecto anterior
- Simplificación de tareas que no aporten al proyecto
- Propuesta de alternativa/s

A partir de las propuestas que hago en este informe la idea es nuevamente charlarlo de manera de antes del 18 tener en limpio el alcance. Con eso listo puedo trabajar tranquilo durante el receso y después en febrero pulirlo.

Análisis del proyecto anterior

El proyecto propuesto en el LAC se centraba en el desarrollo e implementación de un sistema de control para un vehiculo autónomo A continuación presento un desglose de las problemáticas que eso abarca, hecho desde mi punto de vista:
En primer lugar tenemos cuatro grandes ramas:

- **Modelos de Simulación**. Para el armado de modelos que emulen el sistema real. Deben estar correctamente testeados para funcionar de manera genérica
- **Arquitectura de hardware**. Se debe comprender como esta compuesto el vehiculo, o en caso de estar diseñando, proponer una estructura determinada.
- **Parametrizacion**. El primer paso de la implementación Se debe relevar y dimensionar este hardware, de modo de instanciar los modelos de simulación e implementación
- **Modelos de Implementación**. Para las pruebas en campo y evaluación de la performance. Entra en juego la relación costo computacional / performance a la hora de elegir el grado de complejidad de los controladores.

Dentro de la **simulación** tenemos las siguientes problemáticas:

1. **Arquitectura de software**. Resuelve: la articulación de los distintos módulos dentro del sistema.
2. **Entorno de simulacion**. Se debe seleccionar una o varias herramientas, adecuadas para este tipo de trabajo
3. **Modulo de percepcion**. Resuelve: el posicionamiento del vehiculo. En una herramienta como Gazebo es posible simularlo, en Simulink no.
4. **Generacion de referencias**. Resuelve: a partir de una serie de puntos de paso entregar una ruta suave para que el vehiculo siga
5. **Modulo de navegacion**. Resuelve: la adaptación de las referencias de posición y el comportamiento que debe adoptar el vehiculo en cada caso.

6. **Modulo de control**. Resuelve: la necesidad de que el vehiculo siga de manera correcta y segura las referencias que provienen del modulo de navegación Se diseñan en función de los modelos de vehiculo.
7. **Modelos de vehiculo**. Resuelve: la necesidad de emular de manera digital al vehiculo para correr las simulaciones. Van desde un modelo cinemático simple hasta un modelo multicuerpo (gemelo digital) que copie al auto en cuestión.
8. Opcional: Modelos que emulen a los actuadores.

Dentro de la **implementación** tenemos las siguientes problemáticas:

1. **Arquitectura de software**
2. **Modulo de percepcion**.
3. **Generacion de referencias**.
4. **Modulo de navegacion**.
5. **Modulo de control**.
6. **Desarrollo de firmware**. Resuelve: la comunicación entre el alto nivel (Simulación) y el bajo nivel (actuadores). Involucra programar los microcontroladores y elegir los protocolos de comunicación

Simplificación del problema

En total contamos con 16 problemáticas (las 14 enumeradas + arq hardware + parametrizacion). De todos modos vale notar que los modelos de simulación e implementación comparten características y no son completamente independientes.

En primer lugar veamos cuales de estas están resueltas o pueden descartarse a simple vista:

Descartables

- **Modulo de percepcion**. Tal vez una de las cuestiones mas importantes para un problema de manejo autónomo Nos habían pedido una detección de obstáculos simple. Creo que eso no aportaría nada al PFI, mas que perder el tiempo.
- **Modulo de navegacion**. Sin una detección de obstáculos el comportamiento del vehículo puede reducirse a un simple estado de speed tracking. También pueden desarrollarse otros estados, pero de nuevo eso ya esta mas relacionado con la interacción con el ambiente. Como no es el principal interés del proyecto, propongo descartarlo.
- **Desarrollo de firmware**. Es una tarea 100 % de electrónica, y ademas es muy especifica de la implementación No aporta nada al PFI
- **Parametrizacion**. La idea creo sería hacer casos genéricos de manera que puedan ser usados por otra persona para realizar la parametrizacion e implementación

Resueltas

- **Arquitectura de software**. Esta idea la comprendimos bastante bien y elegimos un modelo claro para descomponer las partes y que permite realizar un escalado. El objetivo era que a partir de lo que hicimos nosotros pueda venir otra persona y mejorarlo, sin necesidad de tocar lo que ya esta hecho y aprovechando todo el desarrollo. Por ejemplo: si alguien quería agarrar y desarrollar una red neuronal para la detección de objetos podía usar todo nuestro bloque de control y modelo vehicular sin problemas.
- **Generacion de referencias**. Mediante un sistema relativamente simple de refinamiento y splines.
- **Entorno de simulacion**. Por sencillez, flexibilidad y conocimiento del posible usuario elegiría sin duda MATLAB / Simulink. Ademas tengo bastante trabajo hecho allí

Propuesta de alternativas

Análisis de la situación actual

El numero de problemas a resolver se redujo de 16 a 10. Si acotamos la cuestión al entorno de MATLAB, tenemos la estructura de software hecha:

- Escalable. Cosa que se puedan agregar mas casos y variantes para mejorar la herramienta.
- Con una interfaz de usuario relativamente sencilla. Para que sea fácil de usar.
- Documentada
- Con un sistema de carga de archivos y guardado de salidas también sencillo y claro.

Así pasamos de 10 problemas por resolver a solo 5. Creo que este es el primer punto de partida en el cual tendríamos que acordar (es decir, ver que te parece o si tendríamos que borrar todo esto de un plumazo). Sin embargo para irlo pensando voy ganando tiempo hablando sobre estos 5 problemas.

- Arquitectura de hardware.
- Modelos de vehiculo
- Modelos de actuadores
- Modelos de control (diseño e implementación)

Acá es difícil tratar cada uno de estos problemas como algo completamente distinto, porque están todos interconectados. La **arquitectura es en realidad el punto de unión de todos**. En función de ella se definen los otros 4. Involucra saber, sin parametrizar, como esta formado el sistema en cuestión sobre el que vamos a trabajar (en este caso el tipo de auto). Es decir:

- Mecanismo de suspensión (Normalmente modelado como barras y resortes)
- Tipo de transmisión Hay caja de cambio? Hay diferenciales? Como se produce el flujo de potencia del motor a la rueda?
- Tipo de neumático. Conocemos sus parámetros? Podemos usar un modelo complejo (como la formula mágica) o nos limitamos a algo simple.

- Tipos de actuadores. Como se le da la dirección al vehiculo, hay un volante o le mandamos un comando (como en autos a escala símiles al que estábamos usando). Que tipo de motor tiene? Cuales son sus limitaciones?

Posibles enfoques

Así como lo veo se me ocurren estas 3 alternativas para darle forma al PFI.

- La primera creo es la mas cercana a lo que estábamos haciendo. Involucraría considerar un vehiculo a escala y:
  - Modelarlo con distintos ordenes de complejidad
  - Armarle controladores adecuados
  - Probar el desempeno utilizando distintas trayectorias y establecer criterios dentro de los cuales un modelo es útil o no
  - Dejarlo armado a modo de biblioteca, de manera que pueda ser expandido o utilizado por un futuro PFI.
- La segunda involucra elegir un modelo de auto en particular (con una determinada arquitectura) y:
  - Modelarlo de manera compleja. Tratando de copiar sus características de suspensión y transmisión lo mejor posible
  - Armar un sistema de entrada de señales que emulen los comandos del volante, acelerador y freno. Acá no hay controladores sino que el controlador es un humano.
  - No se me ocurre bien como ensayaría esto. Se puede probar la respuesta ante las diferentes entradas pero no veo claro el objetivo. (En cambio en el caso anterior el objetivo es que siga una trayectoria dentro de las limitaciones físicas existentes)
  - Dejarlo armado a modo de biblioteca, de manera que pueda ser expandido o utilizado por un futuro PFI.
- La tercera es mas genérica e involucra hacer una biblioteca modular que involucre distintas formas de suspensión, transmisión y neumáticos De manera que un usuario pueda relevar un auto y elegir los módulos que mas se parezcan a lo que tiene presente. Para eso deberían estudiarse los tipos de transmisión, suspensión, etc. mas comunes, y proceder en consecuencia. Nuevamente no me quedaría claro como ensayar estos modelos para probar que están bien "Programados"

En lineas generales, la primera trata de seguir la lógica de un problema de manejo autónomo pero acotando bastante el alcance de este PFI en particular. Por ello se limita a trabajar con un modelo de auto a escala completo (transmisión y suspensión relativamente simples).

La segunda complejiza el modelado vehicular pero se independiza de la idea de autonomía (es un cambio grande pero permite reutilizar una parte del trabajo). Los interrogantes que me surgen son: que tipo de arquitectura elegir y como probar el modelo.

La tercera involucra prácticamente re-definir todo y arrancar "de cero". Soluciona en parte el problema del tipo de arquitectura ya que contempla las mas comunes y no solo una, pero permanece la problemática de como testear que este todo correctamente armado.