Final Report

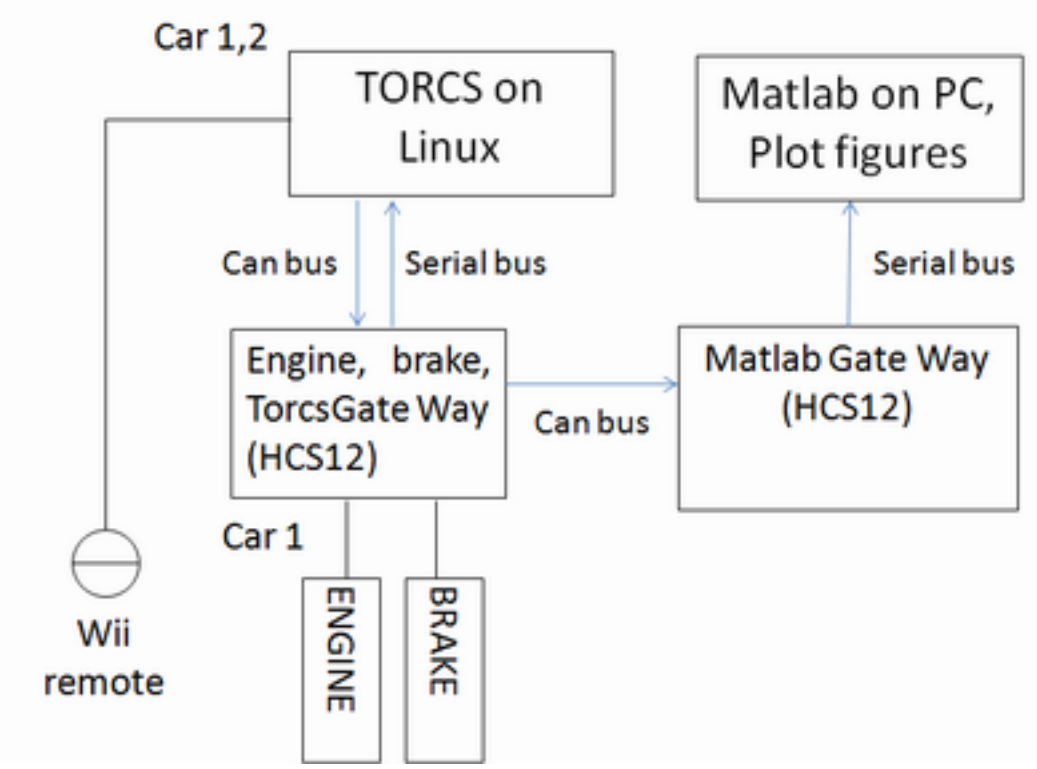
Part 1

Description of Autoplug 3.0:

This project is aimed to design an ACC system in HCS12 microcontrollers to help cars catching up with the front car and also prevent crushing. In this project, ACC will control the engine and brake of the cars. TORCS will send the current position, speed, acceleration and other parameters through Can bus to the ACC microcontroller. Then ACC microcontroller will calculate the speed and acceleration for the next moment and send them back to TORCS. Thus, if two cars stay too close, the HCS12 will make an order to decrease the speed of the back car. On the other hand, if two cars are far away, the HCS12 will increase the speed. We use PID function to insure that the acceleration and deceleration is smooth and comfortable. Moreover, our ACC system is able to detect multiple cars. If one car suddenly surpass our ACC controlled car and stay right in front of us, our car is able to detect this car immediately and make a emergent brake.

If you have any questions after reading this, refer to Autoplug 2.0. Their repo on github is https://github.com/mlab/AutoPlug-2.0

Architecture Diagram:



Evaluation:

We successfully accomplished the functions of ACC, which includes:

Tracking System

If tracking function is turned on, our car can track the nearest front car, there are a few limitations for tracking:

Distance between our car and front car <= 400m;

The front car must stay in the cone of 70 degree of our car;

The 2 cars must stay in the same straight line.

Now we are able to track the front car, and after tracking, there are 2 mode to follow up the front car, one is speed tracking, the other is distance tracking.

Speed Tracking

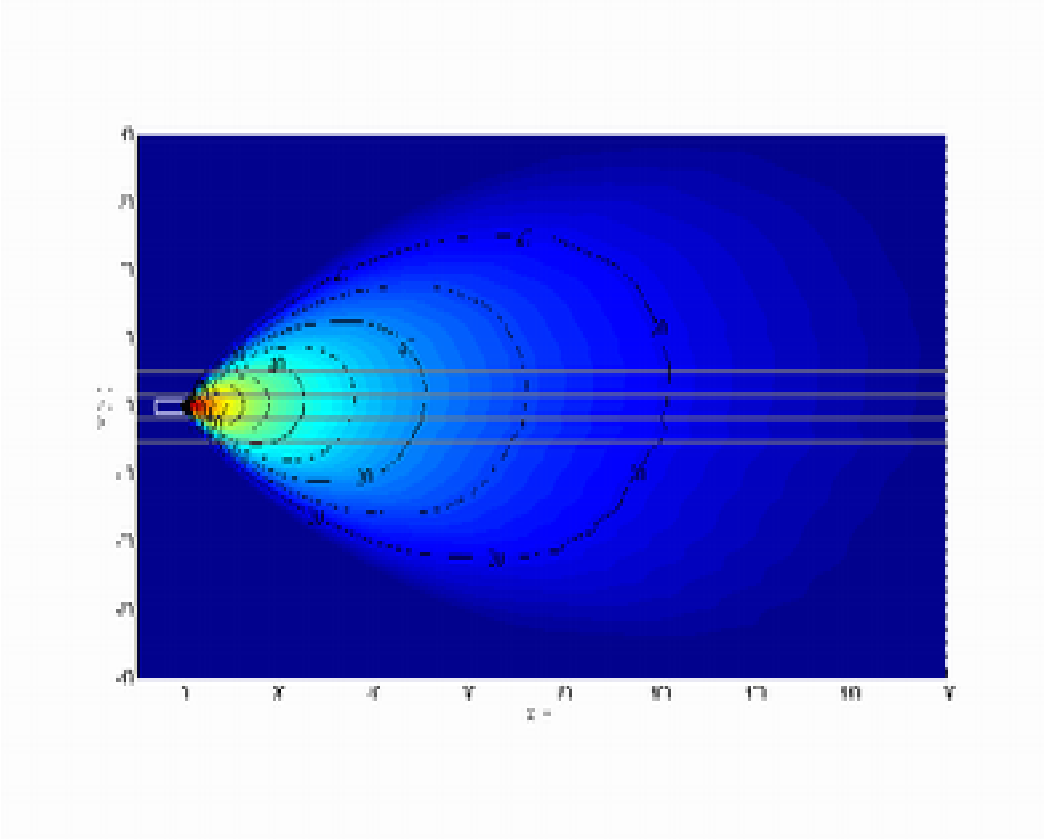
In this mode, regardless of any tracking conditions, system will always keep the same speed with the nearest car on the map. This is only a mode for testing, our car can track the front car even if it is at the other side of the map.

Distance Tracking

The other is distance tracking, in this mode, all the conditions of tracking must be satisfied for enacting the function, our car will stay at the fixed distance to the front car, and this set distance can be changed from 50m to 100m. If the distance is too near, our car will try to brake, if distance is too far, our car will speed up and always tries to keep at the set distance. When loses track, there will not be any auto cruise any more, you have to control the acceleration and brake all by yourself.

Real Radar Simulation System

In order to simulate real radar system, we introduced the radar FOV as stated above in the tracking conditions, the range is 400m in distance and 70 degree cone in front, beyond that range, we consider front car as “untracked”. There is error for real radar as well, and the error is larger if distance or angle is larger, the FOV is like the image below. The error is added to the speed and distance measurement, and will have a slight influence in the Distance tracking.



This image is from UMRR Automotive Type 30 Data Sheet, a Collision Warning Radar from Smartmicro, we took that as a reference.

Multiple Car Detection

Multiple cars can be added and our car is able to detect them, our car is always trying to check if the nearest car satisfies the tracking conditions. The nearest car ID is sent to matlab and torcs will transmit this car's information to the microcontroller. One example scenario is: our car is tracking car1, and car2 is catching up and plugs between our car and car1, our car now can track car2 and gives a sudden brake to avoid crashing.

Part 2

A. Torcs and Autonet:

Step1: Configuration

1. Before installing torcs, you have to install dependencies first. In Linux(We use Ubuntu), you have to install these libraries first if you build it from scratch. The libraries are "libx11-dev libxmu-dev xserver-xorg-dev libxxf86vm-dev freeglut3-dev libgl1-mesa-dev libglu1-mesa-dev libalut-dev libpng12-dev libxi-dev libxine-dev libvorbis-dev libxrender-dev libxrandr-dev". With apt-get, you just need to type "sudo apt-get" before these libraries and then it will install automatically.
2. Download plib from http://plib.sourceforge.net and ./configure && make && make install
3. Download and install TORCS (for Linux) from http://torcs.sourceforge.net/

Follow the installation steps given in the website. It's best to use torcs version 1.3.2. We tested 1.3.4 but it didn't fully work. We have 1.3.2 version in this repo.

Step2: Place the code
Place ESE519->TORCS->autonet in the folder where the other driver files are.(If you follow the tutorial on http://berniw.org, you should place autonet folder into $TORCS_BASE/src/drivers) Then make && make install. Every time you made changes to the autonet.cpp, you should recompile it(make) and reinstall it(make install)

Step 3:
Apply the torcs-individual-brakes patch.

If you have problems with this step, look into the patch file and manually copy/paste the sections of code to the car.h and brakes.cpp files (if you go through them, you will see where the patch adds code).

B. Firmware

Step 0:
Hardware connection. Refer to Autoplug 2.0 and our Architecture Diagram. You won't need microcontrollers for Engine and Brake. Other things are the same.

Step 1:

Download and install CodeWarrior for the HCS12(X) series from http://www.freescale.com/webapp/sps/site/overview.jsp?code=CW_DOWNLOADS

Step 2:

Go to the ESE519->Firmware folder.

CAN_serial is the project for the TORCS ECU.
Serial_CAN is the project for the MATLAB ECU.

Use codewarrior to flash the ECUs with their respective code.

C. Using the WiiMote steering input with autonet

Once TORCS has been setup and the ECUs have been flashed,

Step 1:

Download and install the PCAN-USB driver from http://www.peak-system.com/Produktdetails.49+M5c6d753fe9e.0.html?&L=1&tx_commerce_pi1[catUid]=6&tx_commerce_pi1[showUid]=16

Download and install the Wiiuse library and libbluetooth-dev (or any other bluetooth library).

Step 2:
(Make and) Run driver_input.c from WiiMote_PCAN folder

Press the 1 & 2 buttons on the WiiMote after executing driver_input, and the WiiMote should be connected.

Press the home button on the WiiMote and the WiiMote is ready to use.

Step 3:
When you get into the game, here are the button configuration.

Button	Function
Right	Turn on or off tracking function/Set to forward gear, controlling LED2
Left	Set to reverse gear
Up	Increase set distance by 10 meters
Down	Decrease set distance by 10 meters
Minus	Turn on/off distance tracking function, controlling LED3
Plus	Controlling LED4, should always be off
1	Acceleration
2	Brake
A	Turn on/off speed tracking function, controlling LED1
Home	Enable steer control
LED1(rightmost)	Speed tracking function on/off
LED2	Tracking function on/off
LED3	Distance tracking function on/off

LED4(leftmost)	Should be turned off

Attention: If you want to turn on the speed tracking mode, be sure LED2 and LED1 are on and LED3 is off, if you want to turn on the distance tracking mode, be sure LED2 and LED3 are on and LED1 is off, and LED4 should always be off. When in distance tracking mode the set distance range is from 50m to 100m, the FOV(field of view) to detect front car is 400m and in an angle of 70 degree.

D. Racing in TORCS

Run TORCS and goto the Practice mode. Select a track and select the autonet driver.
Start the race and you're ready to go.

E. MATLAB interface

On the machine running MATLAB, run ESE519->Matlab Gui->numberOne.m
(change the COM# if needed)