24th/June/2020

CANBUS DIAGNOSTIC PROTOCOL CONNECTION WITH ELM327 AND MIT AI2 APP

--- FOREWORD ---

This piece of code is not a complete APP. It's just a way to show how to connect a car ECU (Electronic Control Unit) by means of a CAN interface (the diagnostic interface). It assumes a little knowledge on Bluetooth and CAN protocols (not really so much!). There are many App's capable of a complete diagnose of a car status, but this code is just to verify if we are capable to do the same. Anyone can go further improving this code which is just an exercise. Hoping it can help someone.

--- HW REQUIRED ---

ELM327 CAN to Bluetooth interface: from 5$ on Amazon .

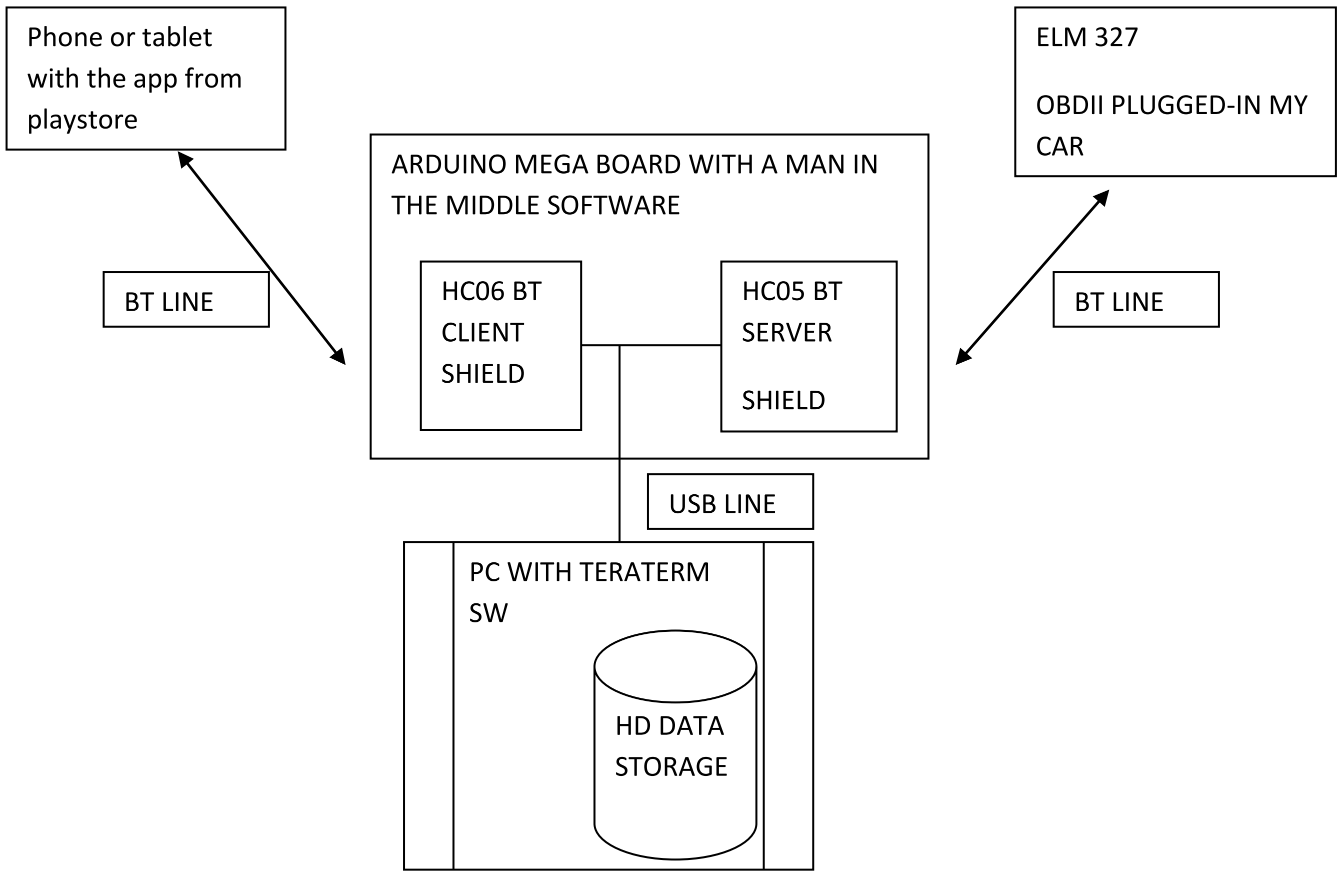
Any 8" display tablet (if you want to use my Screen1 layout, see below, otherwise you can re-arrange it according to your available device)

--- MAN IN THE MIDDLE ---

I have been able to discover the commands sequence used to wake up my car's ECU and to put it diagnostic mode by putting a sniffer (a "man in the middle") between an already done (free) app, available on the Playstore, and the ELM327 connected on the OBDII socket of my car.

In this way I have recorded and analyzed the commands, so to replicate them with my app.

The scheme is here below:



The Arduino SW (written by me) is simply a “sniffer”: any character received by the phone is sent both to the ELM327 and to the PC, where it is stored on the HD. Any character received by the ELM is sent both to the phone and to the PC where it is stored. In this way I have catched all the commands sent by the phone to the ELM 327 (to my car) and all the responses sent by my car to the phone. The Mega board is necessary because we need 3 serial lines (2 for the BT’s and one for the USB toward the PC)

Once stored the commands and their responses on the HD, in a file, I have had the possibility to read them. They are ASCII characters, so easily readable. You don’t need to do the same. I did it just to understand which could be the commands sequence between the app and the car. Once understood the commands and their responses I have written my app.

--- WHY SO MANY CLOCKS ---

Clock1= performs the data receiving from ELM327; it calls the “show data procedure(s)”. It detects if an AT command or CAN command has been received. In the first case it shows the modem commands (AT commands) and their responses. In the second case it calls the procedure that shows CAN data. It is enabled each time a command is sent by the app to the ELM327, and it is stopped whenever a good answer is received.

Clock2 = a scheduler that performs a timely sending to the ELM327 of the CAN commands. It is started at the and of Clock4 and repeats forever until the STOP button is pressed (Note: the START button toggles its label to STOP, once started. It toggles back to START if pressed while it shows STOP).

Clock3 = to ask the ELM327 for Data Trouble Codes retrival from car ECU. Necessary to allow the ECU to read DTC’s and to have tham ready for display. Tested only in a simulated environment, not on the real car.

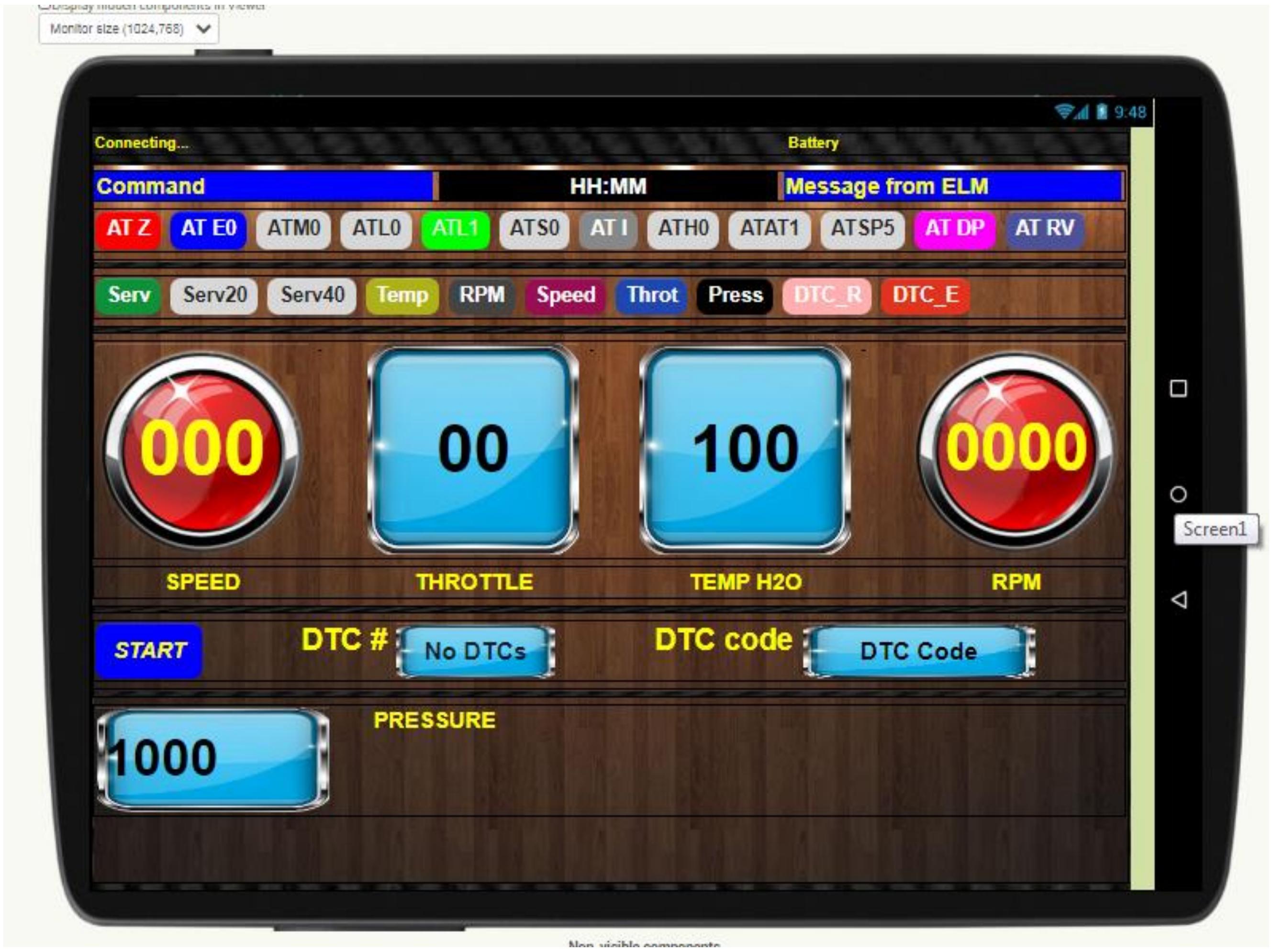
Clock4 = a scheduler to send AT commands in sequence to to initialize the ELM327 and the CAN communication with the car ECU. Started by the START button. It fires many times until the last command in the sequence is called. Then it disables itself.

Clock5 = simply shows hours and min’s onto display. It fires at any second.

--- BT RECEIVING PHYLOSOPHY ---

Once activated the BT client, the app sends a request (via BT) to the ELM and waits for the answer. It waits until a complete response has been collected. Any response is terminated with a “>” character, so a -1 is set as a receiving buffer lenght and it waits until the “>” is received. To avoid to leave characters in the buffer, the receiving steps are repeated until no new characters are received, in the meanwhile (see Clock1). An empty message is also discarded. A CAN response is checked to be the one related to the command: in other words if the command was 01 05 the only accepted answer is 41 05 (any other answer id rejected until a true 4105 is received: no timeouts: “just a quick and dirty software”) .

--- HOW THE SCREEN LOOKS LIKE ---



The coloured buttons send the respective command once (by hand ☺).

They are put in a sequence (from left to right) as they were sent by the ELM327 automatically.

The button START sends the complete sequence automatically in order to initialize the coomunication toward the car ECU

The CAN protocol currently selected is KWP @250 Kbps (#5 in ELM327 list) because that's the one of my car. Most probably yours is another one, just check it with an automatic search of the ELM327.

The BT address is fixed (my ELM327 one): simply change it with the address of your ELM327 interface box.

Take care of the interframe time (some 250 ms typically). It can be suitably tailored with the transmission speed of your car (i.e a CAN @500 Kbps). Don't be too much fast (too short interframe time) otherwise it does not work. Once found the best compromise between updating rate of the screen and the communication capability of your car you can, leave it.

A complete document on ELM327 use and commands is embedded here.

ELM327DS.pdf

--- THANKS ---

Thanks to **KEN**: Don’t care if the throttle, temperature and pressure digits seem to be left aligned, the alignment becomes correct once the app is running because the seven segment font that I use in “your font” extension (KEN’s extension) aligns the characters a bit on the right, so I need to offset them on the left in order to have them in the middle.

Thanks to **TAIFUN**: for his “keep screen on” extension.