# Getting Started - Honda OBD to OBDII Bluetooth Compact usage.

The cable connects to a Honda ECU that uses Hondas' original ECU diagnostic protocol (1990's to the 2000's in some countries). It will not connect to OBDII ECU's (buy a $20 ELM327 OBDII BT dongle for that).

It physically connects to the cars three pin diagnostic connector (AKA  the DLC  port or plug).

The DLC plug is found on the S2000 AUDM or JDM cars below the LHS of the steering column .

The plug often has a green plastic cover and can also have the two pin diagnostic fault lamp activation connector next to it.

## Steps to using with a bluetooth enabled Android phone or tablet (or laptop via it's BT adapter is possible).

The dongle has a bluetooth module that must be paired to the Android device. Pairing bluetooth devices are well explained on the Android devices own user manual usually.
The PIN is set by the manufacturer  of the dongle (try 1234 or 0000), or might be changed by who made the dongle (To do that google for 'HC-05 AT set pin and baud rate' and follow the many online guides).

My compact dongle and cars extended version dongle use **1234** as the pairing PIN.
It should be obvious when you see the name of it on the Android device...
To pair the dongle it needs 12V power, for power it needs to be plugged into the car DLC plug (the ECU will not need to be running to pair). There will be a short beep from the device when the Arduino initialises.
There will be a green power LED on solid on the Arduino side of the dongle, and a slowly flashing LED on the bluetooth module side of the dongle.

The OBDII app you intend to use needs to be loaded on the Android device or PC/Laptop with BT.
I prefer the Torque or Torque Pro app on Android devices. There are many others too.

Start the app. Allow GPS use if the device has it. Open the 'Setup' button ( the cog icon lower LHS of main screen). Change profile to match your cars info if you want to, or leave it default.
Open 'Settings' and open 'OBD2 Adapter Settings'. Open 'Connection type' and select bluetooth.
Back out one menu level and select 'Choose Bluetooth Device', the device you paired should be listed there. Tick it.
Back out a few menu levels to the apps main menu. After a few seconds the blue top line bluetooth dongle status icon should lock steady (indicating good BT connection). The dongle should now have a red 'double flash' on the bluetooth module side of it indicating good BT connection.

Now turn on the ignition or start the car. After a few seconds the  Torque app will try to start communicating with the ECU. If it's successful the blue top status line icon goes steady AND on the dongle a yellow communication LED on the Arduino will start to flash at about 10 times per second.
On the Torque main menu, spin the menu to 'Adapter Status' and select. If the module is working well, there should be information about the dongle, and information listed about what sensors are supported by the dongles firmware  (normally sent by the actual ECU in an OBDII car).

If there is the expected information there, then you can now use the device.
Options here are use the default gauges on the Realtime Information dashboards, and or, setup data logging of some or all the sensors. Logging all sensors may slow down the logging rate so select important sensor items if so. Enable Automatic logging option when Torque starts (or risk missing events if you forget to manually start logging).

Options now are to set the app to use metric units if you use them. Normally after that an app user would setup dashboards with crucial sensors displayed as dials or charts and try it out.

Logs can be exported to spreadsheets, and can also be exported to Google Earth with GPS and sensor data displayed at each GPS waypoint. Graphing can be done also but I prefer to use MS Excel charts or similar.

It's not tested yet by me but it's possible the dongle could be used with the RaceChrono android app to feed it major engine data to log along with it's track data and video....

Currently fault code reading is not supported, that might get done eventually, but since that can be done by the cars dash flash code method it's not very important to me yet.

The buzzer does very little right now, I have not decided if or how to setup some warnings for RPM or coolant temp etc it's there for future use.

There is also data available for relay activity (i.e VTEC solenoid relay) plus A/C relay, O2 sensor heater and brake lights etc. That stuff requires custom PID codes to be programmed into the Torque App custom PID manager, and that is something the user can research how to do.
The custom PID's readable in this FW are:

| Name | "ShortName" | "Mode And PID" | "Equation" | "Min Value" | "Max Value" | "Units" |
|---|---|---|---|---|---|---|
| Aircon relay | A/C | 0x2008 | {B:1} | 0 | 1 | . |
| Brake | Brake | 0x2008 | {B:3} | 0 | 1 | . |
| Economy | Eco? | 0x200c | {B:7} | 0 | 1 | . |
| O2 Heater | O2 Heat | 0x200b | {B:2} | 0 | 1 | . |
| PGM-F1 Relay | PGMFI | 0x200b | {B:0} | 0 | 1 | . |
| VTEC E | VTEC E | 0x200c | {B:3} | 0 | 1 | . |
| VTEC valve | VTS | 0x200a | {B:2} | 0 | 1 | . |

It may be possible to import from the csv file but I suggest adding one at a time, you will need to create a 'Light' for each on the dashboard views.