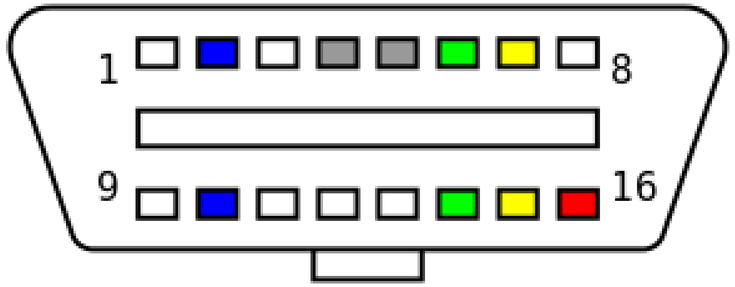
ELM327 OBD II

An overview of the ELM327 microcontroller

Marco Cuoci

OBD II / EOBD Bus

- OBD stands for On Board Diagnostics
- The OBD bus interconnects sensors and control systems inside the car
- By law, the OBD must cover all emission relevant components
- According to the SAE J1962 standard all OBD compliant vehicles must have a standard connector near the driver’s seat
- Usually, queries on the bus can just fetch information (no modify)



ELM327 Microcontroller

- Provides a level of abstraction from the underlying OBD II protocol
  - Hexadecimal queries and responses are mapped to strings
- Connects directly to the car’s OBD connector
- Presents information via a UART interface
  - Usually sent to a Bluetooth or Wi-Fi antenna
- Supports a plethora of protocols for the underlying OBD system

ELM327 OBD Interface

- Queries to the OBD bus are hexadecimal values structured in the following way
- One byte to identify the MODE
  - Ex 01 is current, 03 is to request trouble codes…
- One to three bytes to identify the PID (Parameter ID)
  - As defined in this table

Supported OBD protocols

- SAE J1850 PWM (41.6 kbit/s)
- SAE J1850 VPW (10.4 kbit/s)
- ISO 9141-2 (5 baud init, 10.4 kbit/s)
- ISO 14230-4 KWP (5 baud init, 10.4 kbit/s)
- ISO 14230-4 KWP (fast init, 10.4 kbit/s)
- ISO 15765-4 CAN (11 bit ID, 500 kbit/s)
- ISO 15765-4 CAN (29 bit ID, 500 kbit/s)
- ISO 15765-4 CAN (11 bit ID, 250 kbit/s)
- ISO 15765-4 CAN (29 bit ID, 250 kbit/s)
- SAE J1939 (250kbit/s)
- SAE J1939 (500kbit/s)

ELM327 UART Interface

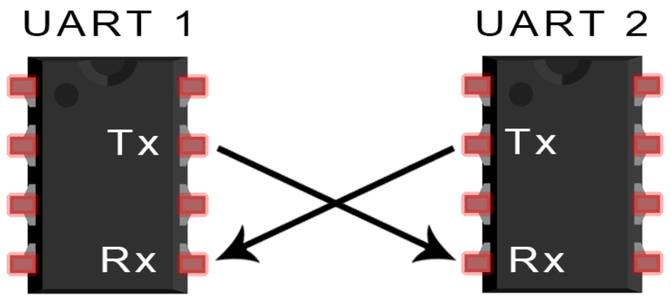
- The ELM 327 can be accessed by UART at baud rates 9600 or 38400

  Fortunately for us, usually the UART interface is wired to an antenna

- The microcontroller acts as an interpreter: strings sent via the UART interface are converted automatically into hexadecimal data packets

- Being a microcontroller, it can be programmed to do various tasks by issuing an “AT” command

  Any query starting with “AT” is directed to the microcontroller, otherwise it is assumed to be a query for the OBD bus

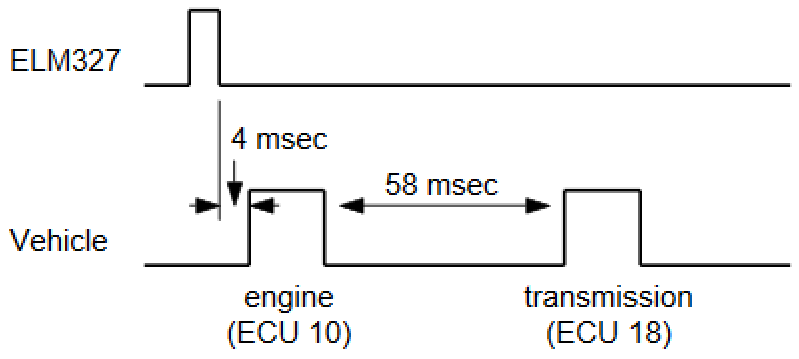


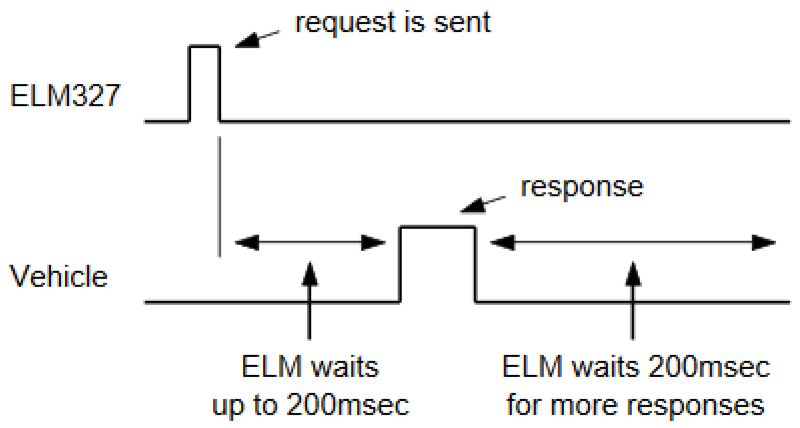
Querying the ELM327

- In order to query the ELM327, we need to know which instruction set we need to use
  - There is an AT instruction set for commands directed at the ELM
  - As well as an OBD instruction set for commands meant for the OBD bus
- Commands are to be sent as strings, terminated by a carriage return
  - Spaces are ignored
- The command is interpreted and if meant for the OBD bus, it is encapsulated according to the negotiated protocol's frame and sent on the bus

Querying the ELM327

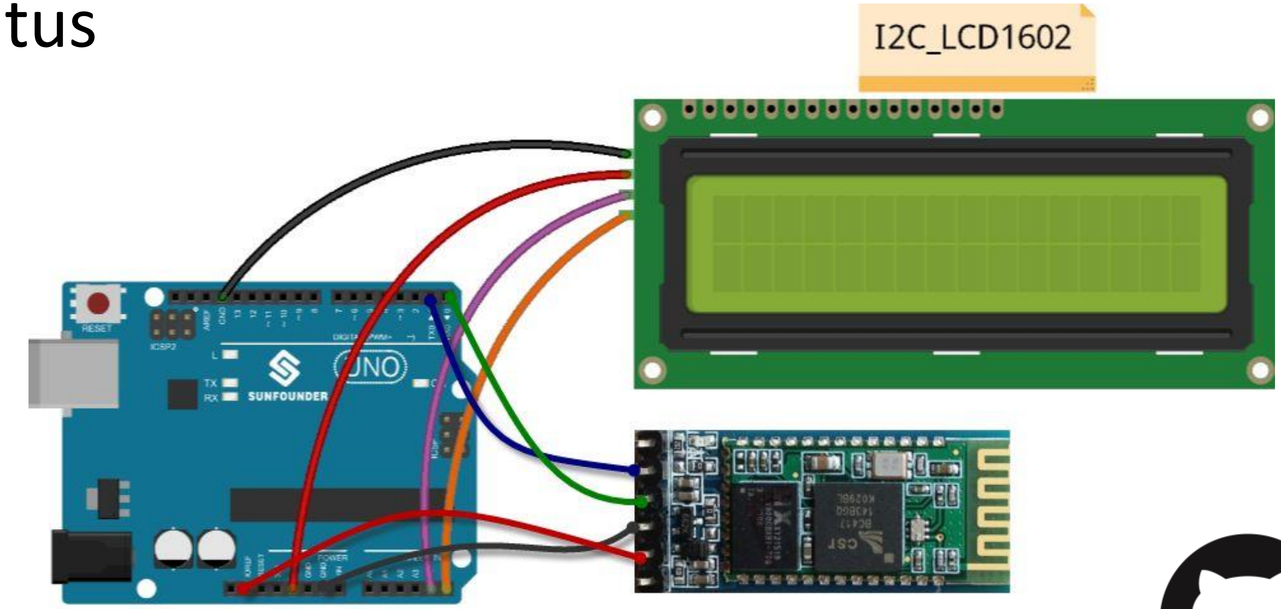
- When the microcontroller receives a response from the OBD bus, it will wait approx. 200ms before returning
  - The delay is meant for diagnostic (mode 03) queries which involve multiple components
  - The time can be cut to near-zero by issuing the “AT AT2” command
- The value is then returned as a string via the UART interface





A practical example: Overview

- Use an Arduino to communicate over Bluetooth to the ELM327 device and project real time data to a transparent glass piece using a LCD
- The Arduino should make contextual decisions on what to display according to the vehicle's status



A practical example: Hardware Issues

- First problem: How do mirrors work

  Mirrors reverse the image, if we were to project directly from the LCD to the glass piece, we would have an inverted view of the values

  Solution: Use two mirrors to reverse the image twice

- Second problem: Apparently having loose cables on the dashboard of a car is not a good idea

  During testing, poor road condition was enough for jumper wires to come off the breadboard breaking the circuit and crashing the software

  Solution: 50% duct tape, 50% good *capable* friends

A practical example: Documentation

- Three sets of documentations were used for the project
  - OBD Instruction Set from the Wikipedia page
  - ELM Instruction Set from Sparkfun site
  - ELM Datasheet from ELM electronics site

- Two header files containing definitions for the needed commands were then produced following the instructions from these documents
  - ELM327_defines.h
  - OBD_defines.h

A practical example: Encoding the data

- We have said that the ELM accepts strings terminated by a carriage return
- Arduino compilers support C++
- We can make use of the Arduino String library
  - Automatically handled
  - Concatenation is an easy supported operation (+)
- Strings are then sent over the Bluetooth interface via the Wire library
  - Use Wire.println() function for an automatic carriage return at the end

A practical example: Decoding the data

- Data is sent from the ELM327 to the Arduino in string format
- The original command is sent back too as an acknowledgment
- Unnecessary data such as the original command and spaces need to be cleaned from the response
  - Use String.replace() to clean responses
- The string is a hexadecimal number
  - Use strtol() to convert to a long from hexadecimal to decimal format
- Beware! Different commands have different response lengths!