CST 472

# Memo

**To:** Kevin Pintong

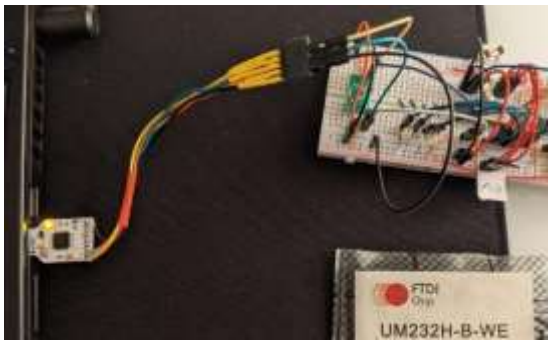
**From:** Zak Rowland

**Date:** February 6, 2021

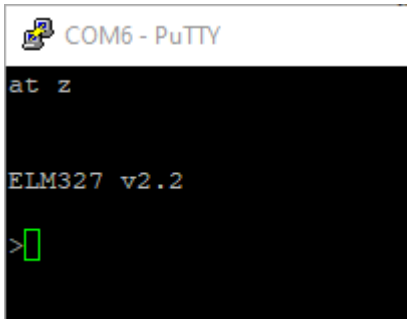
**Re:** Memo 2

---

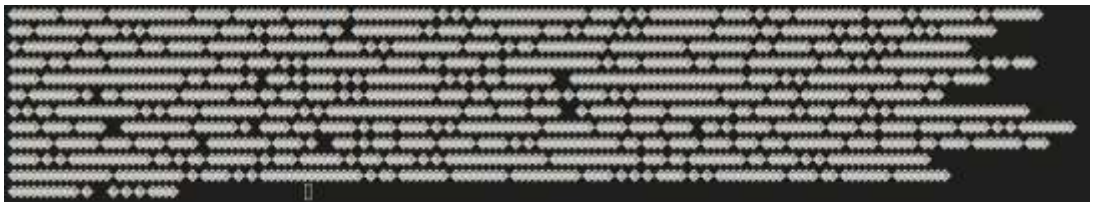
Since I would be doing some debugging on my circuit, I found a 15V wall power brick I could use in place of my car's battery. The 7805 regulator I'm using can tolerate up to about 30V so this worked out just fine. I disconnected the ELM327 chip from power and measured the 5V and 3.3V sources again to confirm nothing broke in the power circuit. Both sources were fine so I plugged the ELM back in. I used a serial to USB device from a previous course to communicate with the ELM through a terminal.



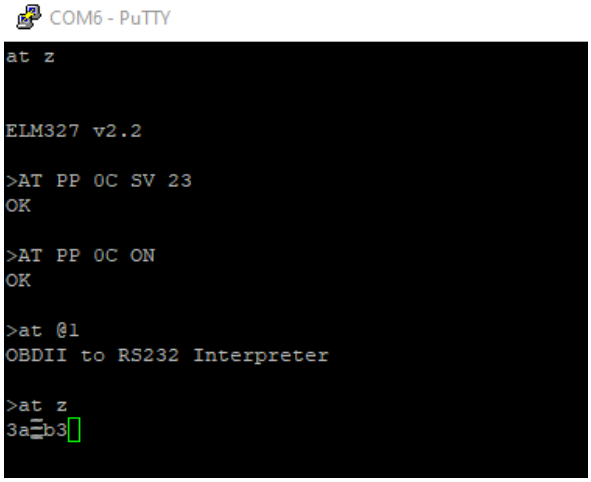

After installing PuTTY and setting up the serial settings, I was able to issue commands and view responses from the ELM327.



This confirms the rest of the circuit is working is expected, but the issue lies in the ESP32's UART or the logic level conversion chip. The logic level conversion chip is the one I may have overheated while soldering. I set up a simple UART configuration in the Arduino IDE and also tried an example UART code in the Visual Studio Code ESP extension, both ended in the same result of garbage data in the serial monitor.



One weird thing I noticed was when I commented out all reads and writes to the UART, the serial monitor still filled with gibberish – indicating something else may be going on. I also configured the ELM327 to use 115.2K baud instead of the default 38,400 as seen below. The AT PP 0C X commands are setting the new baud rate, and once AT Z (reset) is issued, the configuration takes effect.




After updating the UART tests to use this new baud rate, the results were still the same. At this point I determined I would probably need an oscilloscope to debug the ESP side of the circuit and moved on to testing the functionality with my car. With the serial to USB device, I connected the circuit to my car's OBD-II port and was able to successfully interface with it. I was able to read coolant temps., engine RPM, and the VIN number and others. I also issued the commands to clear diagnostic codes and read diagnostic codes. Most of this data comes back as hex, so I had to interpret the different pieces of data I requested as seen in the screenshots below.

```
>at I
ELM327 v2.2

>at SP 0
OK

>01 00
SEARCHING...
41 00 BE 1F A8 13

>01 05
?

>01 0C
41 0C 00 00

>01 05
41 05 3A

>01 0C
41 0C 00 00

>01 0C
NO DATA

>01 0C
?

>01 0C
?

>01 0C
?

>01 0C
NO DATA

>01 0C
41 0C 17 0C

>01 0C
NO DATA
```

at I - returns the version number, confirms the chip is on and functioning

at SP 0 - tells the ELM327 to search for a protocol automatically based on the connected vehicle

01 00 - initiates the protocol search on the OBD-II port, 41 00 means it is a response to the 01 00 cmd., the last 4 bytes is the requested data (supported PIDs)

01 05 - requests the current coolant temperature, this attempt didn't work

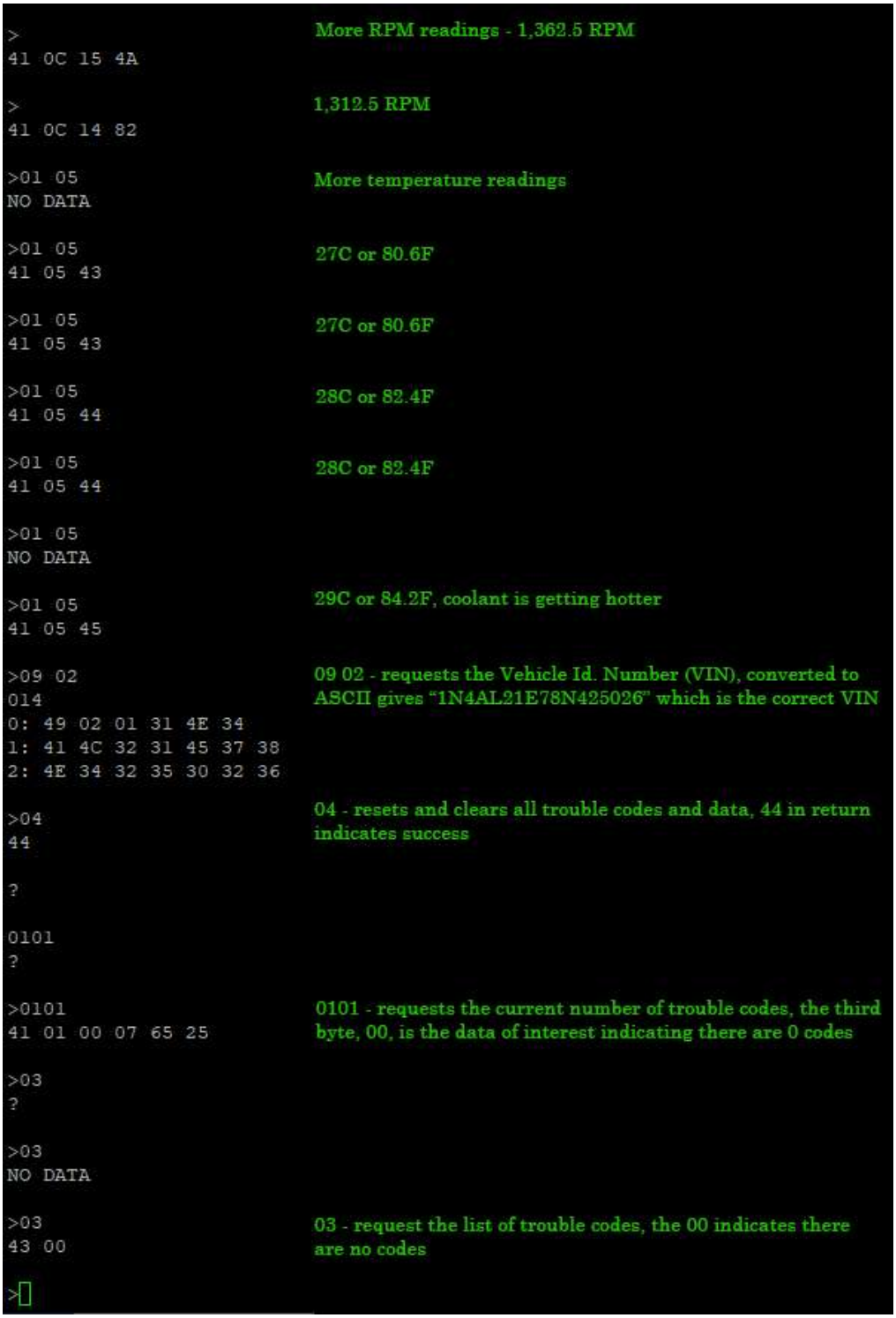
01 0C - requests the current engine RPM, the last two bytes, 00 00, is the data indicating 0 RPM (car was off)

01 05 - requests the coolant temperature, the byte 3A is the data in Celcius (58 in decimal), there is an offset of 40 to allow subzero temps. so it is actually 18C/64F

01 0C - confirmed the engine RPM as still 0 then started the car

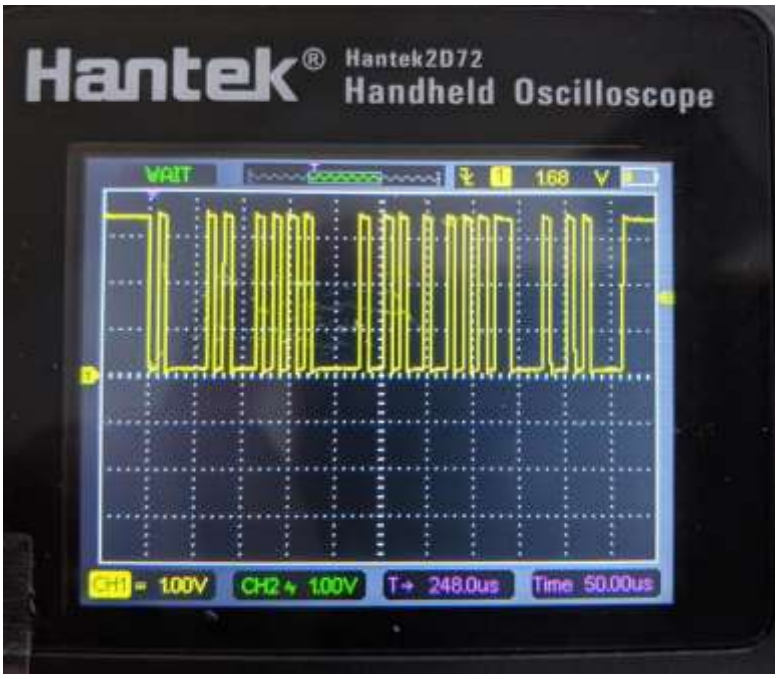
Constantly requesting RPM until car is initialized after starting ...

Successfully retreived RPM once car was idling, 17 0C, which is 5,900 but must be divided by 4 since the RPM is read in 1/4 increments, so the RPM was 1,475



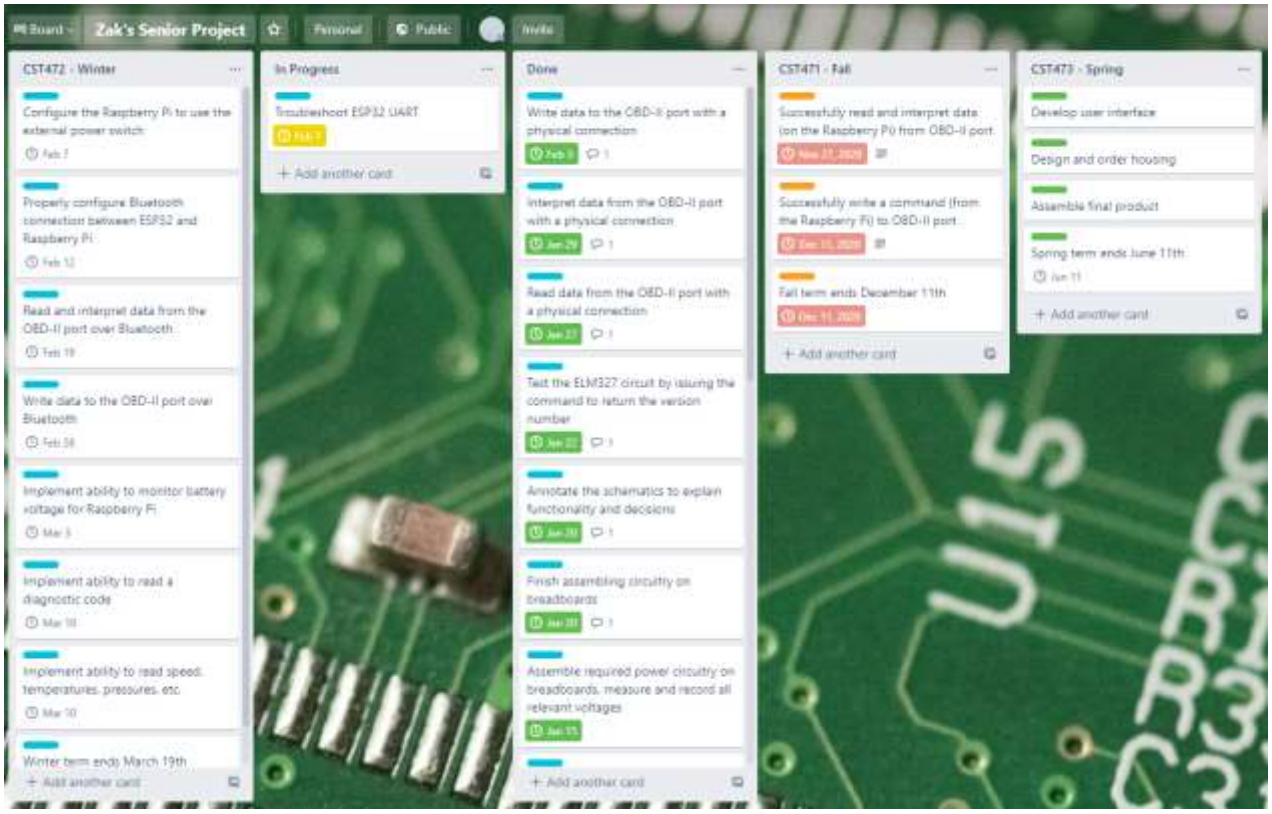

I recently picked up an oscilloscope from Oregon Tech to debug the circuit. It's a handheld unit that is a little confusing to use but I think it will work for the project. I connected it the ESP's UART1 TX line to see what was happening. I noticed in the serial monitor that the ESP gets stuck in a short boot loop with SPI errors if the scope is connected while it boots. After looking through the Espressif documentation, it mentioned UART1 pins can have unexpected behavior since they are used for SPI by default. Instead of changing the config., I just changed

pins to UART0 TX. I was able to get a screenshot of the signal and it appears to be 8N1 serial data as it should, but when I tried decoding the data it didn't make sense and start/stop bits didn't line up.




My next steps are continuing to debug the serial connection and begin working on the Bluetooth connection with the Pi.

Hours worked since Memo 1: 9.5




https://trello.com/b/57kp2fr0/zaks-senior-project