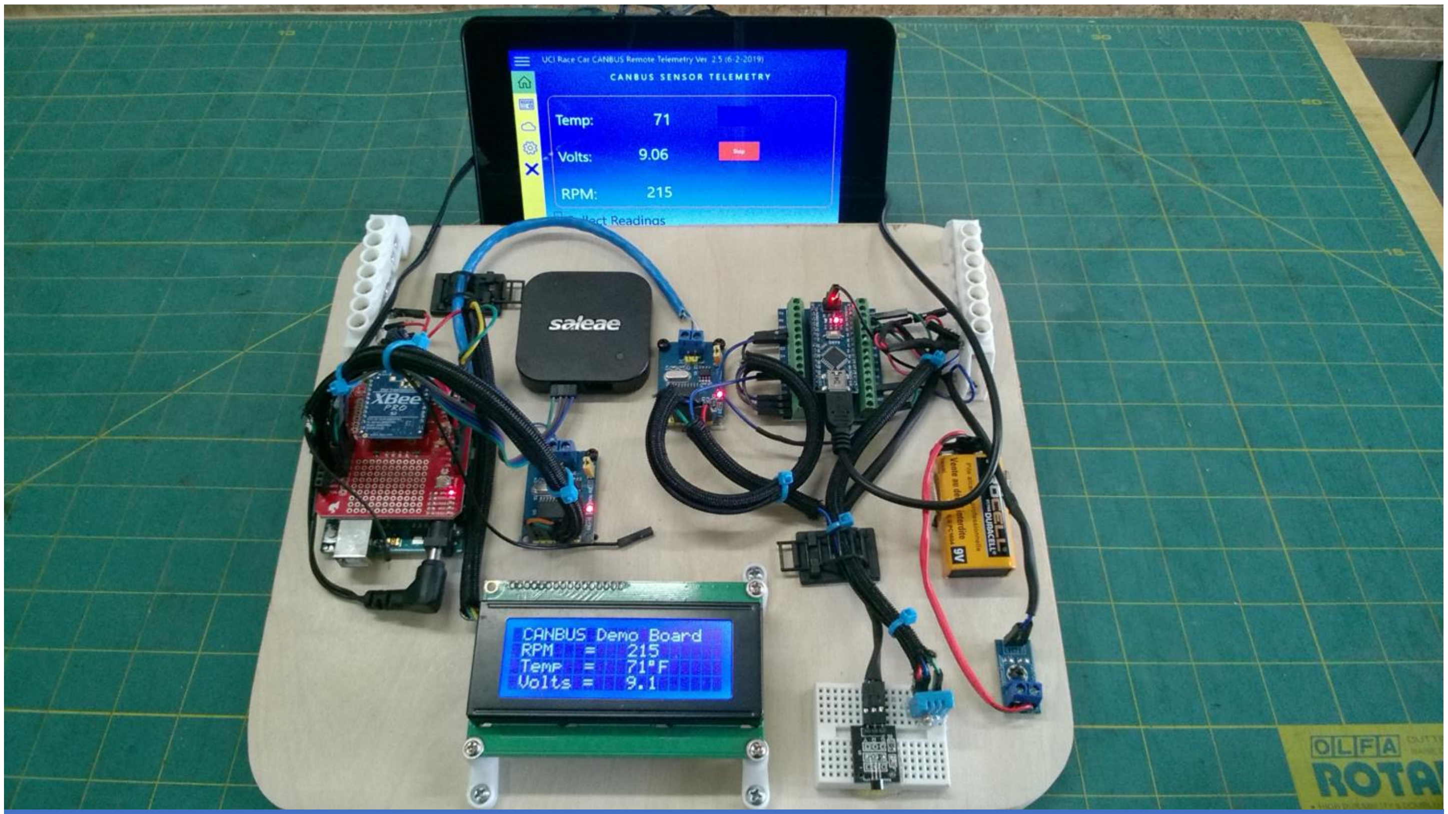


Working with CANBUS Serial Communication Networks Part 3

Decoding SPI Sensor data packets with the Saleae Logic 8 analyzer

Using the Logic Analyzer to decode data from Three Sensors

In this section I have included the actual screenshots for the results of the SPI decoder on three types of data. I transmitted RPM data from my Hall sensor, Temperature from the DHT-11 sensor, and Voltage from my volt sensor. Since the RPM range of my drill (with attached magnet) is around 500, I had to use 16 bits to represent that number.

RPM was converted into two bytes and were the first two elements in my 8-byte array. The temperature was byte 3. Since I sent it in Celsius there was no need to convert it. The max temp could be 255!

Finally, the voltage sensor data was multiplied by 10 to remove the decimal. This made it easy to package up. Then it was converted into two bytes just like RPM data was.

The remaining bytes = 0x0 and are not shown here.

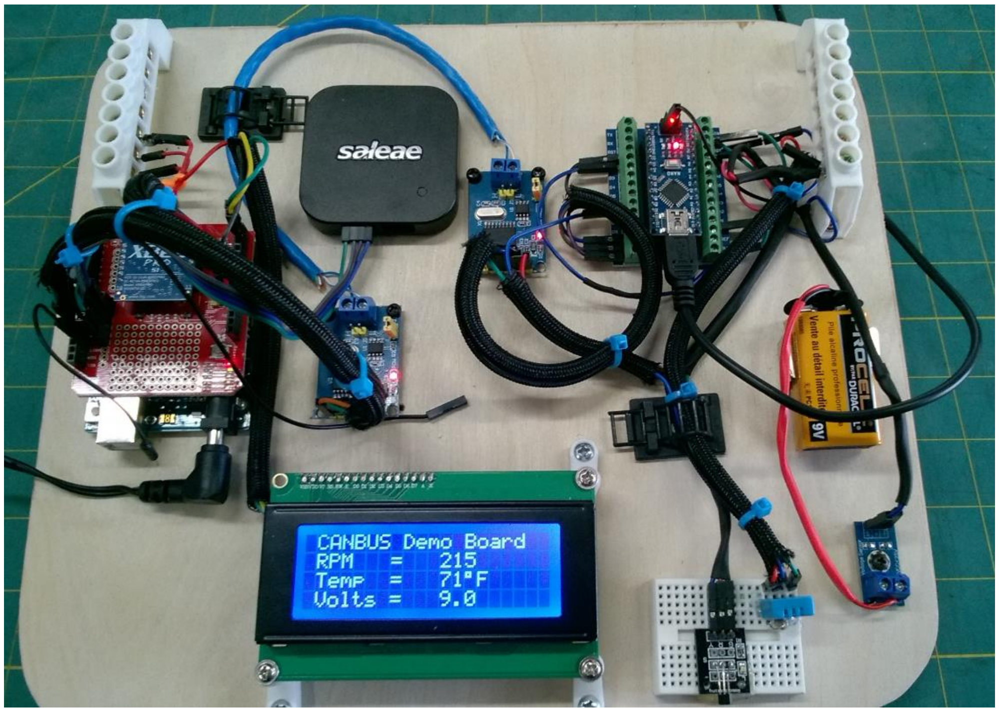
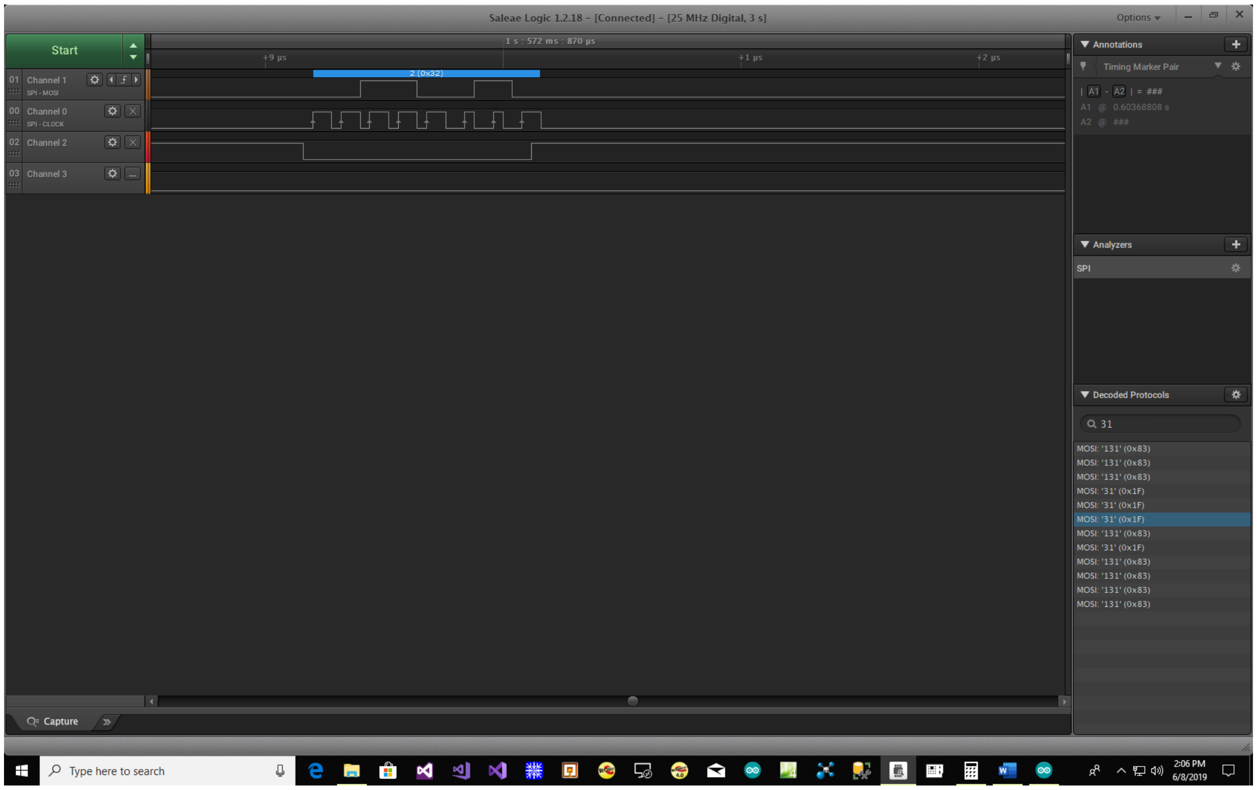


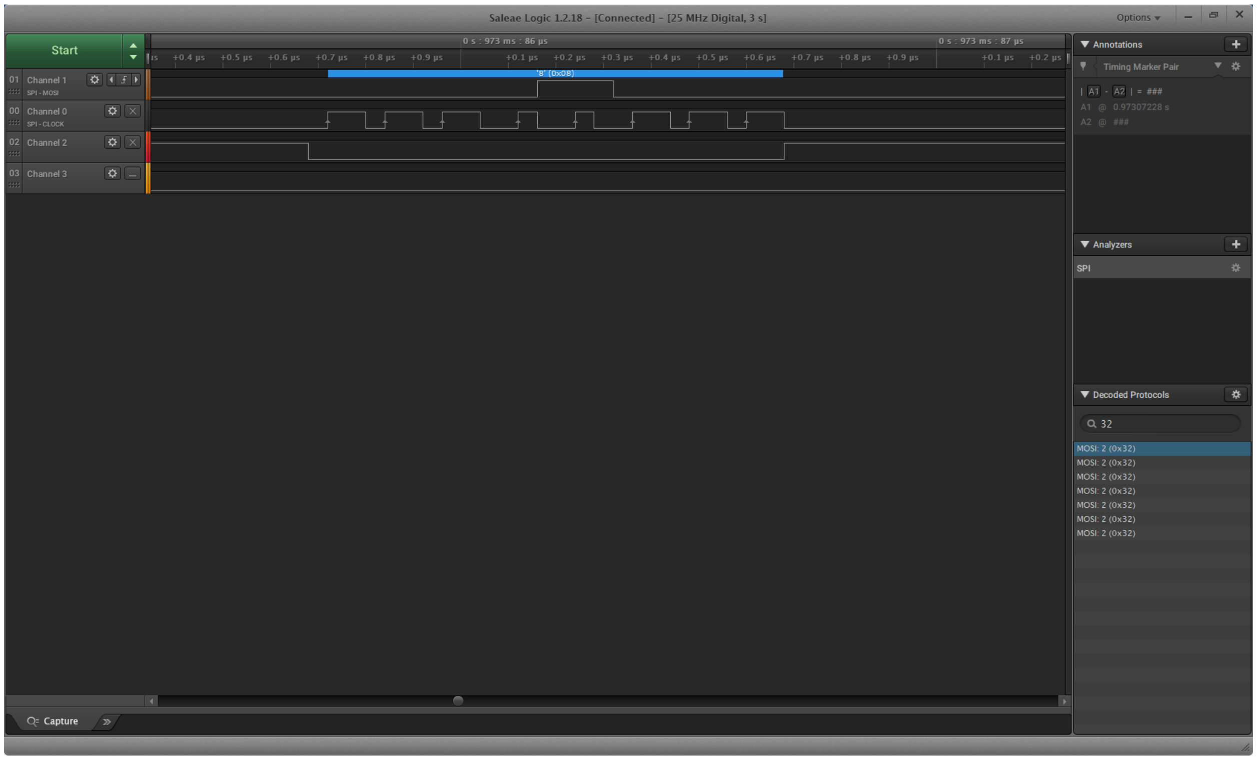
Figure 1 View of my CANBUS Demo Board. I am sending three sensor values and decoding them at the receiving end (Arduino UNO) with the Saleae Logic 8.

1. Here we view the first byte in my packet. I set the Can ID = 0x20 (32 D)



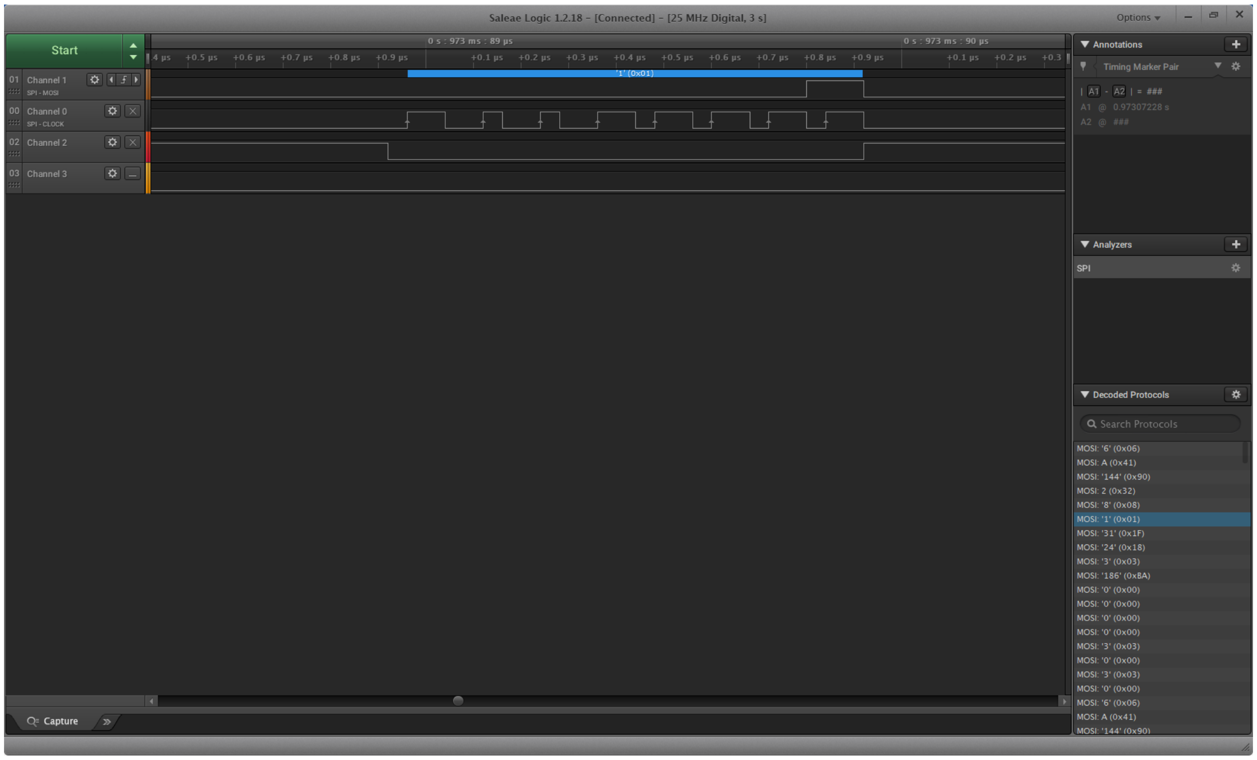
```
31
MOSI: '131' (0x83)
MOSI: '131' (0x83)
MOSI: '131' (0x83)
MOSI: '31' (0x1F)
MOSI: '31' (0x1F)
MOSI: '31' (0x1F)
MOSI: '131' (0x83)
MOSI: '31' (0x1F)
MOSI: '131' (0x83)
MOSI: '131' (0x83)
MOSI: '131' (0x83)
MOSI: '131' (0x83)
```

2. Byte 2 = 8 (# of bytes sent in my packet. CANBUS needs to know this.)



```
32
MOSI: 2 (0x32)
MOSI: 2 (0x32)
MOSI: 2 (0x32)
MOSI: 2 (0x32)
MOSI: 2 (0x32)
MOSI: 2 (0x32)
MOSI: 2 (0x32)
```

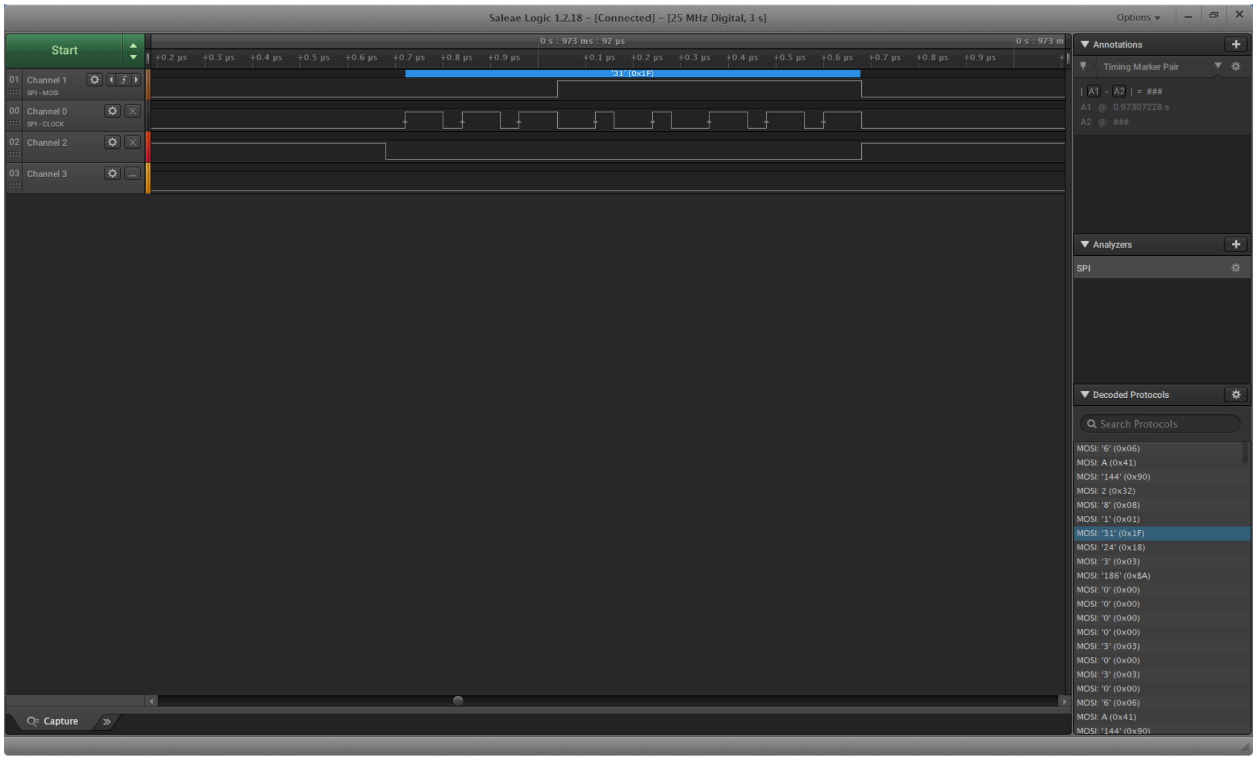
3. Next comes RPM which was 287. This shows the RPM $_{\text{Highbyte}}$ = 1 (RPM$_{\text{highbyte}}$ = 287 >> 8)



Saleae Logic 1.2.18 - [Connected] - [25 MHz Digital, 3 s]

```
MOSI: '6' (0x06)
MOSI: A (0x41)
MOSI: '144' (0x90)
MOSI: 2 (0x32)
MOSI: '8' (0x08)
MOSI: '1' (0x01)
MOSI: '31' (0x1F)
MOSI: '24' (0x18)
MOSI: '3' (0x03)
MOSI: '186' (0xBA)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '3' (0x03)
MOSI: '0' (0x00)
MOSI: '3' (0x03)
MOSI: '0' (0x00)
MOSI: '6' (0x06)
MOSI: A (0x41)
MOSI: '144' (0x90)
```

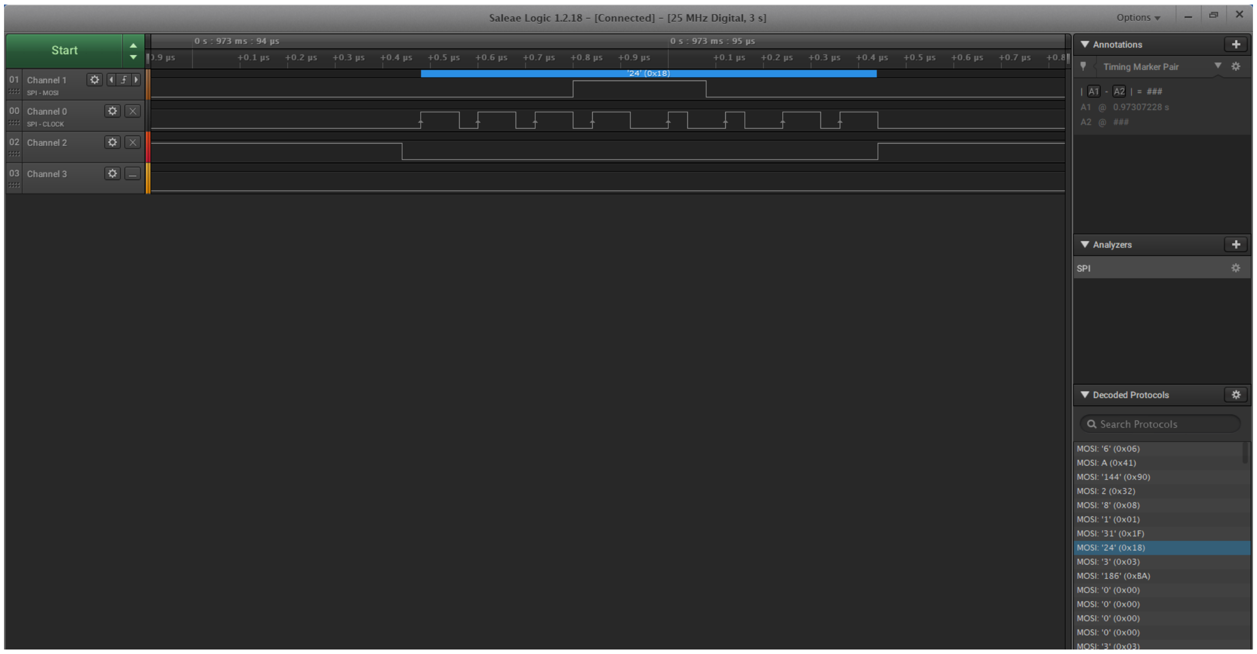
4. RPM $_{\text{Lowbyte}}$ = 31 (287 AND 255)



Saleae Logic 1.2.18 - [Connected] - [25 MHz Digital, 3 s]

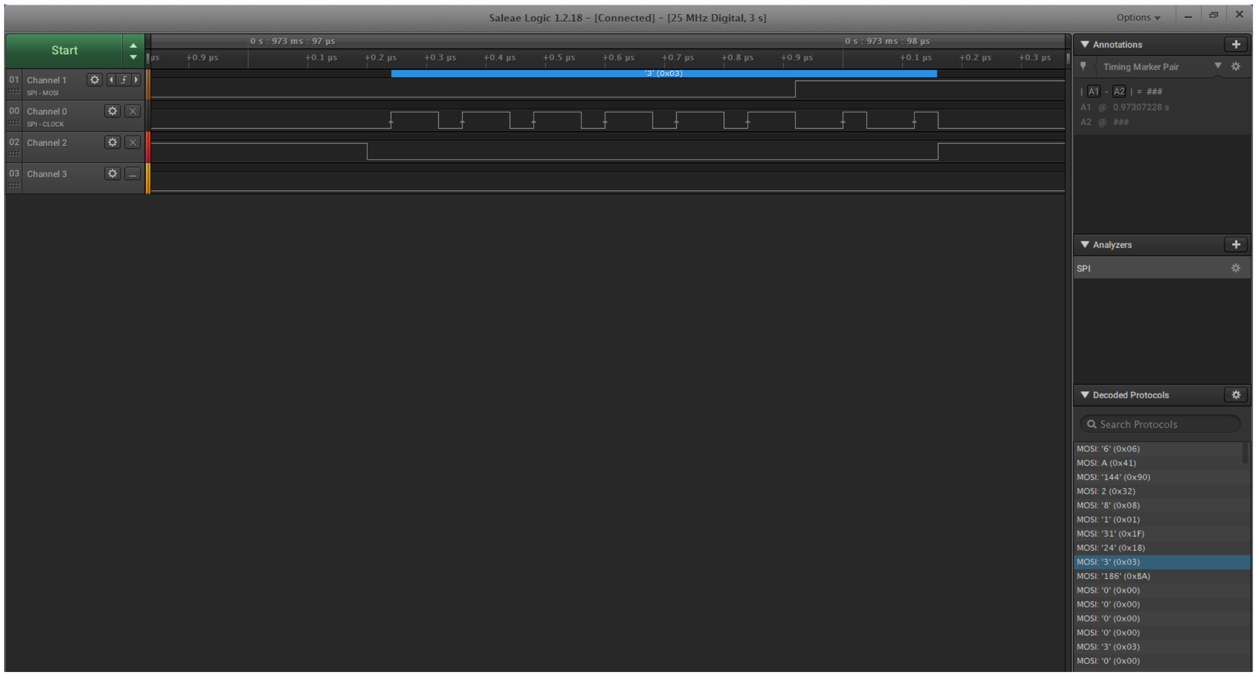
```
MOSI: '6' (0x06)
MOSI: A (0x41)
MOSI: '144' (0x90)
MOSI: 2 (0x32)
MOSI: '8' (0x08)
MOSI: '1' (0x01)
MOSI: '31' (0x1F)
MOSI: '24' (0x18)
MOSI: '3' (0x03)
MOSI: '186' (0xBA)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '3' (0x03)
MOSI: '0' (0x00)
MOSI: '3' (0x03)
MOSI: '0' (0x00)
MOSI: '6' (0x06)
MOSI: A (0x41)
MOSI: '144' (0x90)
```

5. Next comes the Temperature in Celsius = 24 (75 F). Since it fits in one byte, I sent it as is and converted it to Fahrenheit at the receiving end where it is displayed on the LCD.



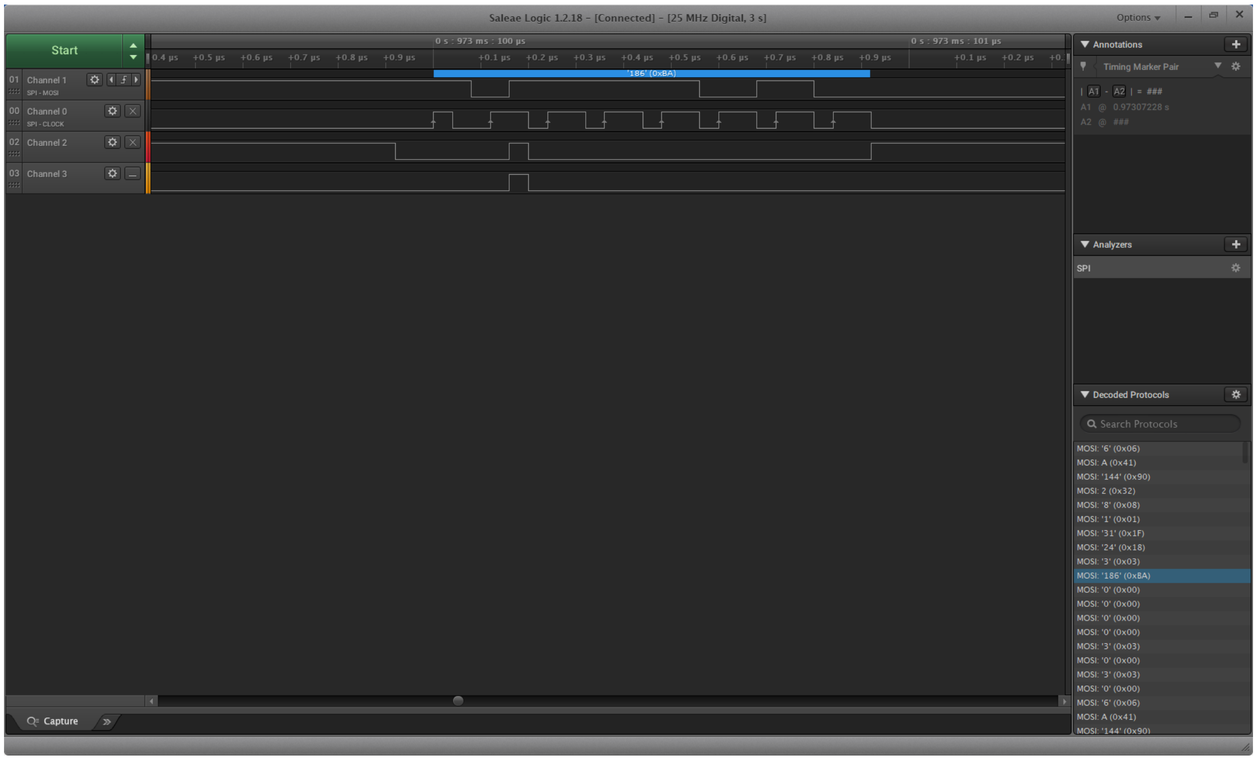
```
MOSI: '6' (0x06)
MOSI: A (0x41)
MOSI: '144' (0x90)
MOSI: 2 (0x32)
MOSI: '8' (0x08)
MOSI: '1' (0x01)
MOSI: '31' (0x1F)
MOSI: '24' (0x18)
MOSI: '3' (0x03)
MOSI: '186' (0xBA)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '3' (0x03)
```

6. I use a 9V battery as my source. The Voltage of 9.54 is transmitted as 954 to remove decimals. Voltage $_{\text{Highbyte}}$ = 3 (954 >> 8).



```
MOSI: '6' (0x06)
MOSI: A (0x41)
MOSI: '144' (0x90)
MOSI: 2 (0x32)
MOSI: '8' (0x08)
MOSI: '1' (0x01)
MOSI: '31' (0x1F)
MOSI: '24' (0x18)
MOSI: '3' (0x03)
MOSI: '186' (0xBA)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '3' (0x03)
MOSI: '0' (0x00)
```

7. The Voltage $_{\text{Lowbyte}}$ = 186 (954 AND 255)



```
MOSI: '6' (0x06)
MOSI: A (0x41)
MOSI: '144' (0x90)
MOSI: 2 (0x32)
MOSI: '8' (0x08)
MOSI: '1' (0x01)
MOSI: '31' (0x1F)
MOSI: '24' (0x18)
MOSI: '3' (0x03)
MOSI: '186' (0xBA)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '0' (0x00)
MOSI: '3' (0x03)
MOSI: '0' (0x00)
MOSI: '3' (0x03)
MOSI: '0' (0x00)
MOSI: '6' (0x06)
MOSI: A (0x41)
MOSI: '144' (0x90)
```

8. To display the actual RPM and voltage value on the LCD, I recombined the two bytes. Here I show just the voltage bitwise operations. The RPM calculations are done in exactly the same manner.
   a. Voltage = 3 $_{\text{Highbyte}}$ <<8 = 768.
   b. 768 | 186 = 954. (| = OR bitwise operator).
   c. 954/100 = 9.54 which is the actual voltage of my 9V battery I was testing.

The remaining bytes in the packet = 0x0 and are not shown here.

Final thoughts (Or I love it when it works!)

While collecting and decoding my packets, I kept getting wrong results. The analyzer was decoding but the data made no sense. After speaking with the Saleae support team, I was directed to an article about how to decode SPI. In that article, they show how the clock and MOSI lines can be out of sync. This can happen if the analyzer starts reading in the middle of the packet stream.

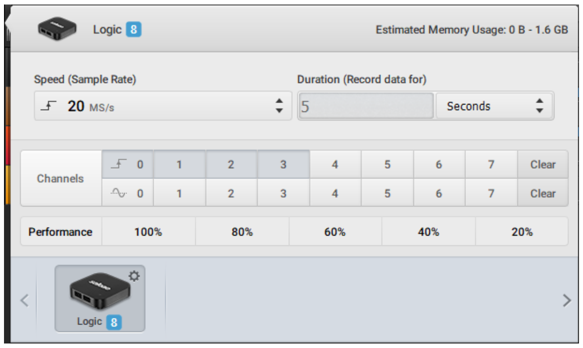
To solve this, you tell the analyzer to start decoding when the clock line in LOW. Then you re-run the decoding from that marker you place on the screen. YOU DO NOT CAPTURE AGAIN at that point. Once I did this, I could see that the clock was starting and stopping in sync with the data. This is easy to view on any of the above screen shots. You can count 8 CLK cycles directly underneath the data packet being decoded. I spent as whole day working on this issue, so I hope I saved you that frustration! Here is their article https://support.saleae.com/protocol-analyzers/analyzer-user-guides/using-spi

Packet Capturing and Analyzer Setup Details

For completeness, I have included some information that may assist you in gathering CANBUS packets like I did. I used a *Saleae Logic 8* analyzer and chose the SPI decoder protocol. The sample rates are shown in this image. I used 25MS/s capture speed for this discussion but I have had excellent success using the settings shown.

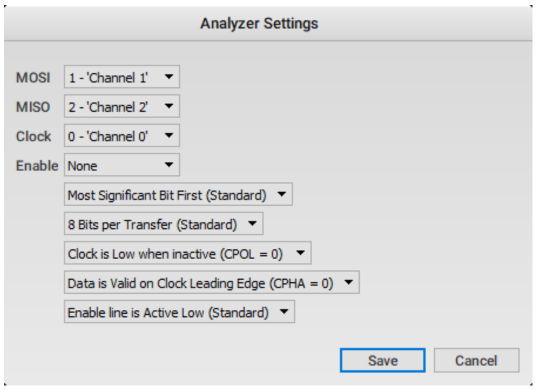
The packets were captured from the SPI ports on the Arduino UNO since it was the receiving end of the network as shown on the next page.

Be sure to set the ENABLE line to NONE as shown.



Speed (Sample Rate): 20 MS/s
Duration (Record data for): 5 Seconds

Performance	100%	80%	60%	40%	20%



MOSI: 1 - 'Channel 1'
MISO: 2 - 'Channel 2'
Clock: 0 - 'Channel 0'
Enable: None
Most Significant Bit First (Standard)
8 Bits per Transfer (Standard)
Clock is Low when inactive (CPOL = 0)
Data is Valid on Clock Leading Edge (CPHA = 0)
Enable line is Active Low (Standard)

Here is a closer look at the SPI Arduino connections for the Logic analyzer.

Saleae Pins	SPI	Arduino Pins
Green = 0	| SPI Clock	13
Gray = 1	| SI	11
Purple = 2	| SO	12
Blue = 3	| CS	10
Black = Ground		GND