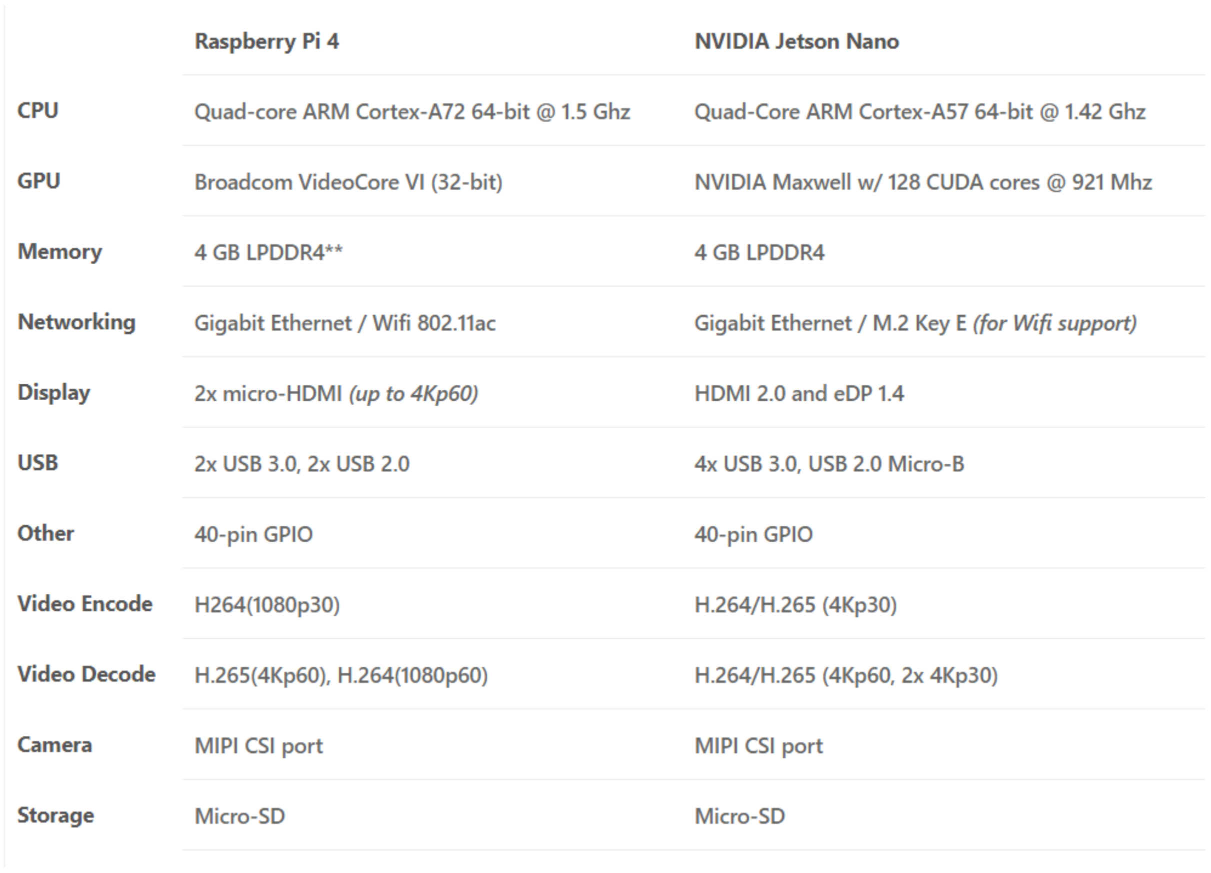
TASK 2

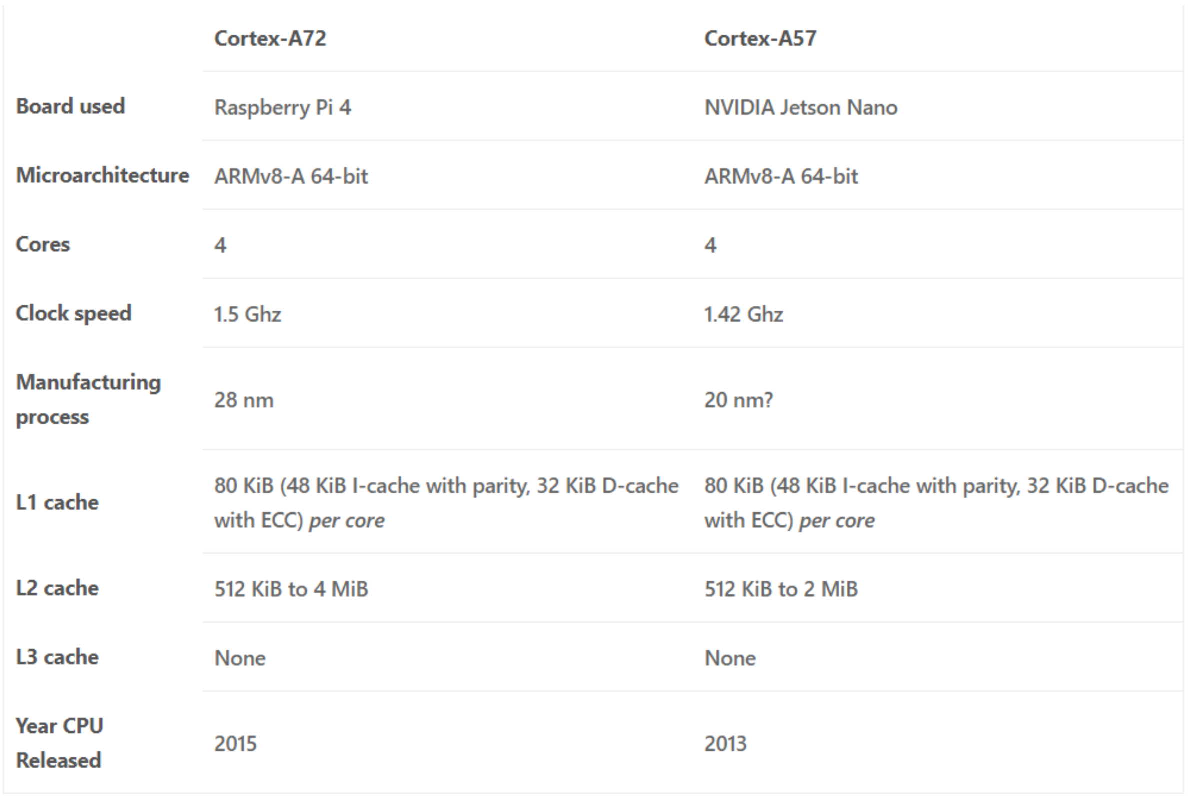
A microcontroller is used when a specific or a single task needs to be handled with precision also when added features like ADC and DAC are needed along with a range of I/O pins and various communication compatibility like I2C, UART and SPI. Also with many added features like timers and counters but microcontrollers fails when it comes to handling a big task which requires memory as well speed this is where Microprocessors are needed since microprocessor can handle big task and provide lot of memory for the task management also they provide us the flexibility of changing their RAM, ROM and also adding extra features to them which microcontrollers don't.

The various parameter that we need to look out in the microprocessor is it's power compatibility that is if it is suitable for our task...which here is ROVER. The speed of the processor , their RAM and ROM , also their operating temperature range, their bandwidth , power efficiency, heating issues also that whatever task we want to do that whether they are compatible of doing it or not.

The best microprocessor that we can get which is better than R-pi 4 which is the currently the latest model of the Raspberry Foundation is NVIDIA Jetson Nano.



	Raspberry Pi 4	**NVIDIA Jetson Nano**
CPU	Quad-core ARM Cortex-A72 64-bit @ 1.5 Ghz	Quad-Core ARM Cortex-A57 64-bit @ 1.42 Ghz
GPU	Broadcom VideoCore VI (32-bit)	NVIDIA Maxwell w/ 128 CUDA cores @ 921 Mhz
Memory	4 GB LPDDR4**	4 GB LPDDR4
Networking	Gigabit Ethernet / Wifi 802.11ac	Gigabit Ethernet / M.2 Key E *(for Wifi support)*
Display	2x micro-HDMI *(up to 4Kp60)*	HDMI 2.0 and eDP 1.4
USB	2x USB 3.0, 2x USB 2.0	4x USB 3.0, USB 2.0 Micro-B
Other	40-pin GPIO	40-pin GPIO
Video Encode	H264(1080p30)	H.264/H.265 (4Kp30)
Video Decode	H.265(4Kp60), H.264(1080p60)	H.264/H.265 (4Kp60, 2x 4Kp30)
Camera	MIPI CSI port	MIPI CSI port
Storage	Micro-SD	Micro-SD

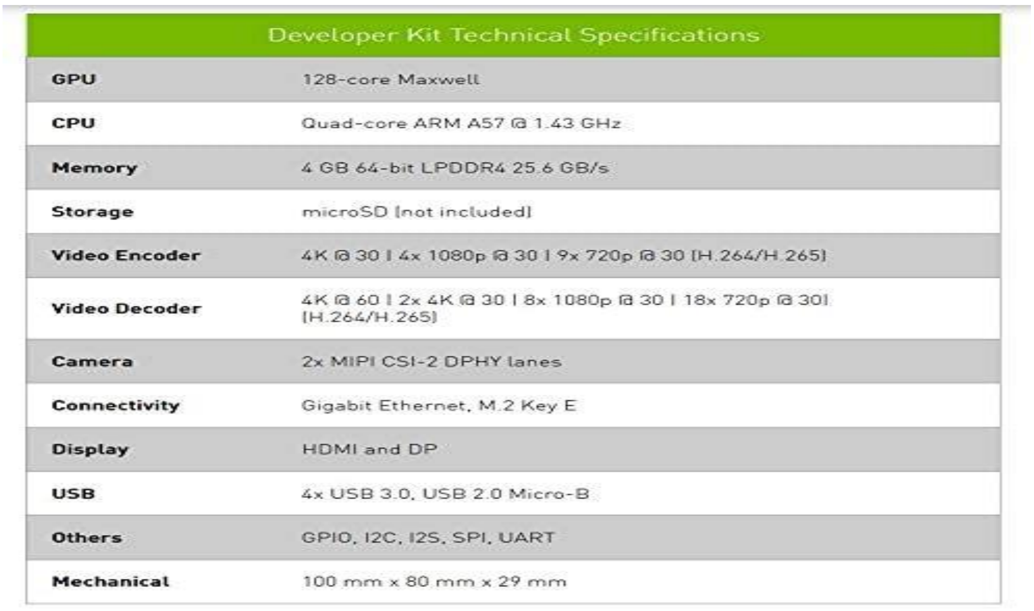


	Cortex-A72	Cortex-A57
Board used	Raspberry Pi 4	NVIDIA Jetson Nano
Microarchitecture	ARMv8-A 64-bit	ARMv8-A 64-bit
Cores	4	4
Clock speed	1.5 Ghz	1.42 Ghz
Manufacturing process	28 nm	20 nm?
L1 cache	80 KiB (48 KiB I-cache with parity, 32 KiB D-cache with ECC) *per core*	80 KiB (48 KiB I-cache with parity, 32 KiB D-cache with ECC) *per core*
L2 cache	512 KiB to 4 MiB	512 KiB to 2 MiB
L3 cache	None	None
Year CPU Released	2015	2013

The Jetson Nano cost around 7.5K .

No not just the bandwidth but also the power efficiency ,it’s compatibility with requirement and also the heating issues.

Microcontrollers are embedded with ADC and DAC this is because many sensors require analog input or we require digital output from many sensors thus we need ADC and DAC. Also in order to use motor drivers we need PWM signals and for that we need ADC and DAC . For various serial communication protocols as well.



Developer Kit Technical Specifications	
GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (not included)
Video Encoder	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decoder	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30| (H.264/H.265)
Camera	2x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI and DP
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I2C, I2S, SPI, UART
Mechanical	100 mm x 80 mm x 29 mm

The reason why STM32 is preferred over Arduino is because-:

STM32F3DISCOVERY Discovery kit provides the following hardware components:

- STM32F303VCT6 in LQFP100 package
- ARM® 32-bit Cortex® -M4 CPU with FPU
- 72 MHz max CPU frequency
- VDD from 2.0 V to 3.6 V
- 256 KB Flash
- 40 KB SRAM
- Routine booster: 8 Kbytes of SRAM on instruction and data bus
- GPIO with external interrupt capability
- 4x12-bit ADC with 39 channels
- 2x12-bit D/A converters
- RTC
- General Purpose Timers (13)
- USART/UART (5)
- I2C (2)
- SPI (3)
- CAN
- USB 2.0 full speed interface
- Infrared transmitter
- DMA Controller

Whereas For Arduino Mega which is one of the powerful development board by Arduino having 54GPIO has-:

- Flash Memory 256 KB of which 8 KB used by bootloader
- SRAM 8 KB
- EEPROM 4 KB
- Clock Speed 16 MHz

As we can see that STM board provide more speed and memory than Arduino boards also in Arduino they have 10 bit analogue to digital convertor. Whereas in STM32 it's 12 bit.

In order to do the communication between the microprocessor and microcontroller they must have compatible communication protocols in them that is UART or I2C or SPI or CAN so that the communication can be done between them.

No we cannot connect two devices with unique functionalities to the same port this is because both the devices will receive same signals at the same time also while communicating with them the data we will receive will be combined and we won't be able to differentiate.

A board worth to mention and can be checked out is Rock Pi N 10 model.

CAN BUS

CAN bus is currently widely used in automobiles. The major advantages that are found in CAN bus are-:

- Low Cost-: It is because many nodes can be connected with just two wires. Thus it reduces the wiring cost and also the complexity.
- Robust-: The system is robust towards failure of various subsystems and also is not affected by electromagnetic interference.
- The CAN messages are prioritized with ID's and the highest ID's are uninterrupted.

CAN networks use a shared bus to connect all nodes in the network . There is no Master-Slave relationship in the CAN protocol. Instead, all nodes have access to the same bus, and bit-wise arbitration of each message is used to determine priority and avoid bus-collisions. Each message contains a unique identifier along with a priority level. If two nodes try using the bus simultaneously, the message with higher priority will get the priority bus access while the lower priority message will abort. In CAN bus the transceiver is always receiving data and in the end they filter out the required data. Thus all the nodes in the CAN is receiving data all time.

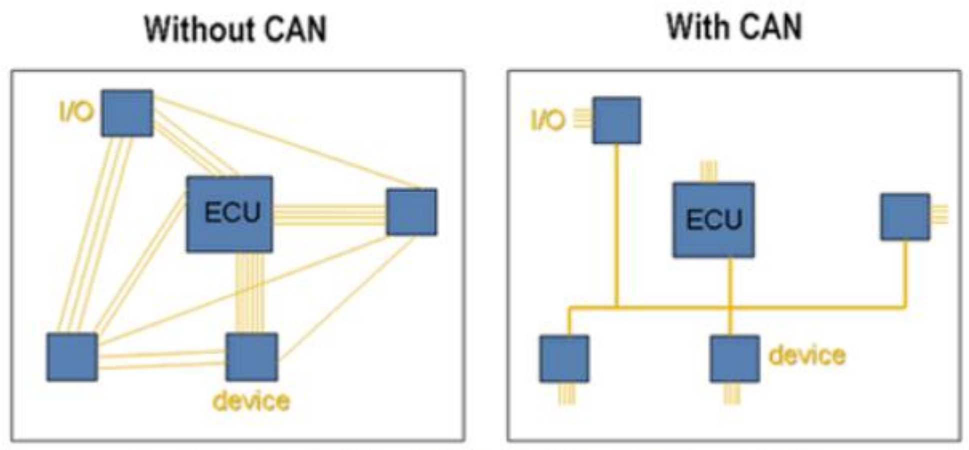
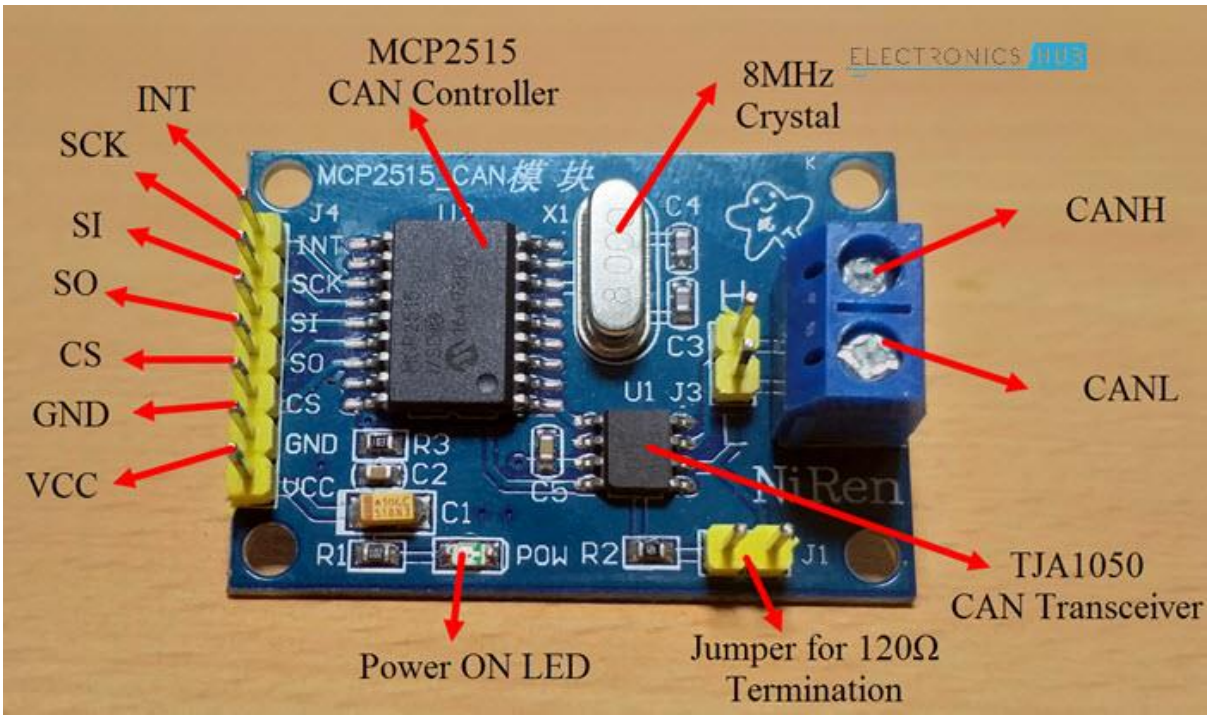
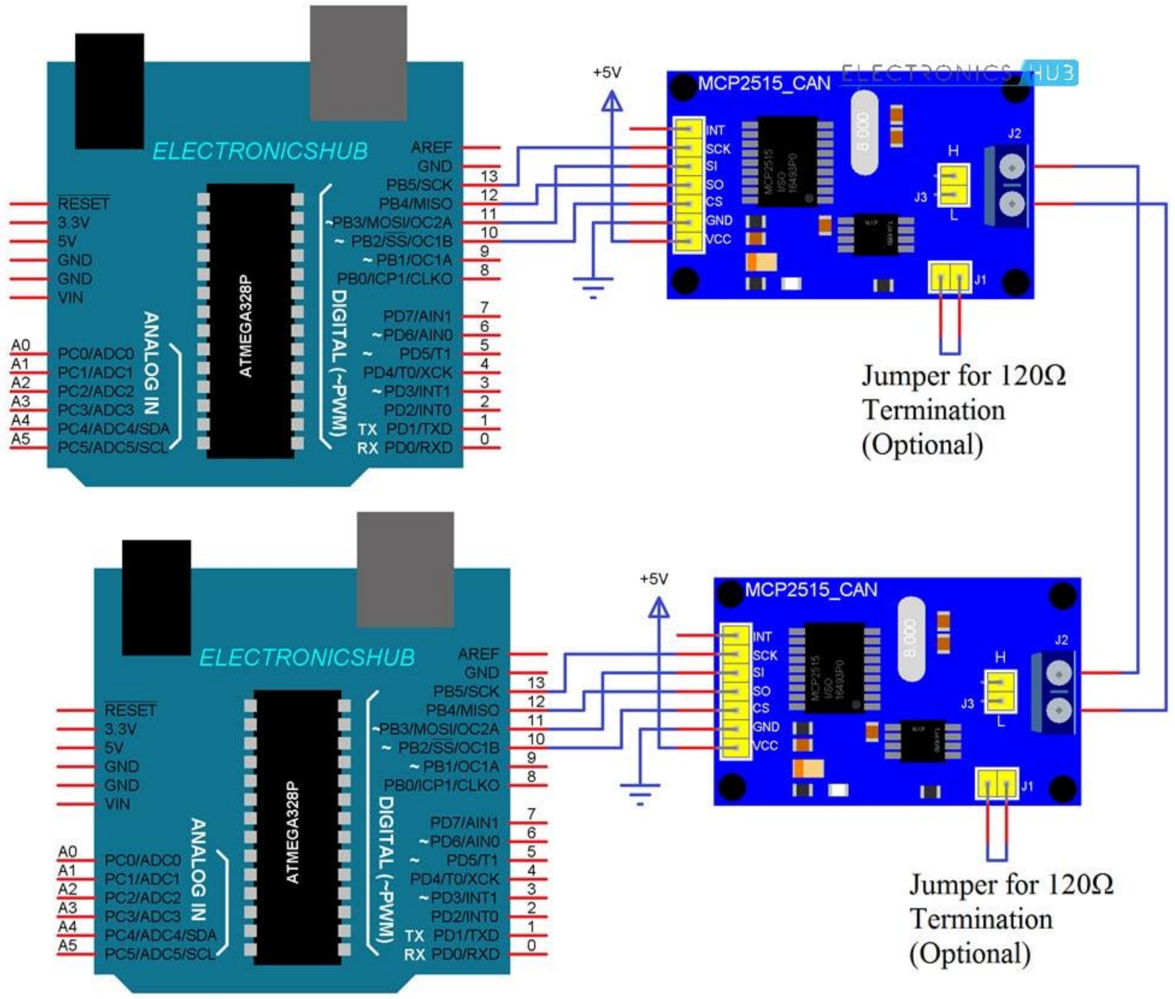


Fig. 1: Example of CAN and Without CAN in automotive application

In order to implement CAN bus communication we need CAN Transceiver. Also make the required connections necessary for the particular board. Below is the example of CAN bus communication between two Arduino UNO.



MCP2515
CAN Controller
8MHz
Crystal
ELECTRONICS HUB
INT
SCK
SI
SO
CS
GND
VCC
CANH
CANL
TJA1050
CAN Transceiver
Power ON LED
Jumper for 120Ω
Termination



ELECTRONICSHUB
ATMEGA328P
DIGITAL (~PWM)
ANALOG IN
+5V
MCP2515_CAN
INT
SCK
SI
SO
CS
GND
VCC
J1
J2
J3
H
L
Jumper for 120Ω
Termination
(Optional)

Here we are using MCP2515 CAN Transceiver.

Below is the Transmitter Code.

```
#include <SPI.h>
#include <mcp_can.h>

const int spiCSPin = 10;
int ledHIGH    = 1;
int ledLOW     = 0;

MCP_CAN CAN(spiCSPin);

void setup()
{
    Serial.begin(115200);

    while (CAN_OK != CAN.begin(CAN_500KBPS))
    {
        Serial.println("CAN BUS init Failed");
        delay(100);
    }
    Serial.println("CAN BUS Shield Init OK!");
}

unsigned char stmp[8] = {ledHIGH, 1, 2, 3, ledLOW, 5, 6, 7};

void loop()
{
  Serial.println("In loop");
  CAN.sendMsgBuf(0x43, 0, 8, stmp);
  delay(1000);
}
```

This is The Receiver Code-

```
#include <SPI.h>
#include "mcp_can.h"

const int spiCSPin = 10;
const int ledPin = 2;
boolean ledON = 1;

MCP_CAN CAN(spiCSPin);

void setup()
{
    Serial.begin(115200);
    pinMode(ledPin,OUTPUT);

    while (CAN_OK != CAN.begin(CAN_500KBPS))
    {
        Serial.println("CAN BUS Init Failed");
        delay(100);
    }
    Serial.println("CAN BUS  Init OK!");
}
```

```
void loop()
{
    unsigned char len = 0;
    unsigned char buf[8];

    if(CAN_MSGAVAIL == CAN.checkReceive())
    {
        CAN.readMsgBuf(&len, buf);

        unsigned long canId = CAN.getCanId();

        Serial.println("-----------------------------");
        Serial.print("Data from ID: 0x");
        Serial.println(canId, HEX);

        for(int i = 0; i<len; i++)
        {
            Serial.print(buf[i]);
            Serial.print("\t");
            if(ledON && i==0)
            {

                digitalWrite(ledPin, buf[i]);
                ledON = 0;
                delay(500);
            }
            else if((!(ledON)) && i==4)
            {

                digitalWrite(ledPin, buf[i]);
                ledON = 1;
            }
        }
        Serial.println();
    }
}
```