alselectro

All about Microcontrollers,Tips on Pc,Android phone,Electronics

Sensor Data on CAN BUS–Arduino with CAN2515

Filed under: ARDUINO — 1 Comment
April 29, 2020

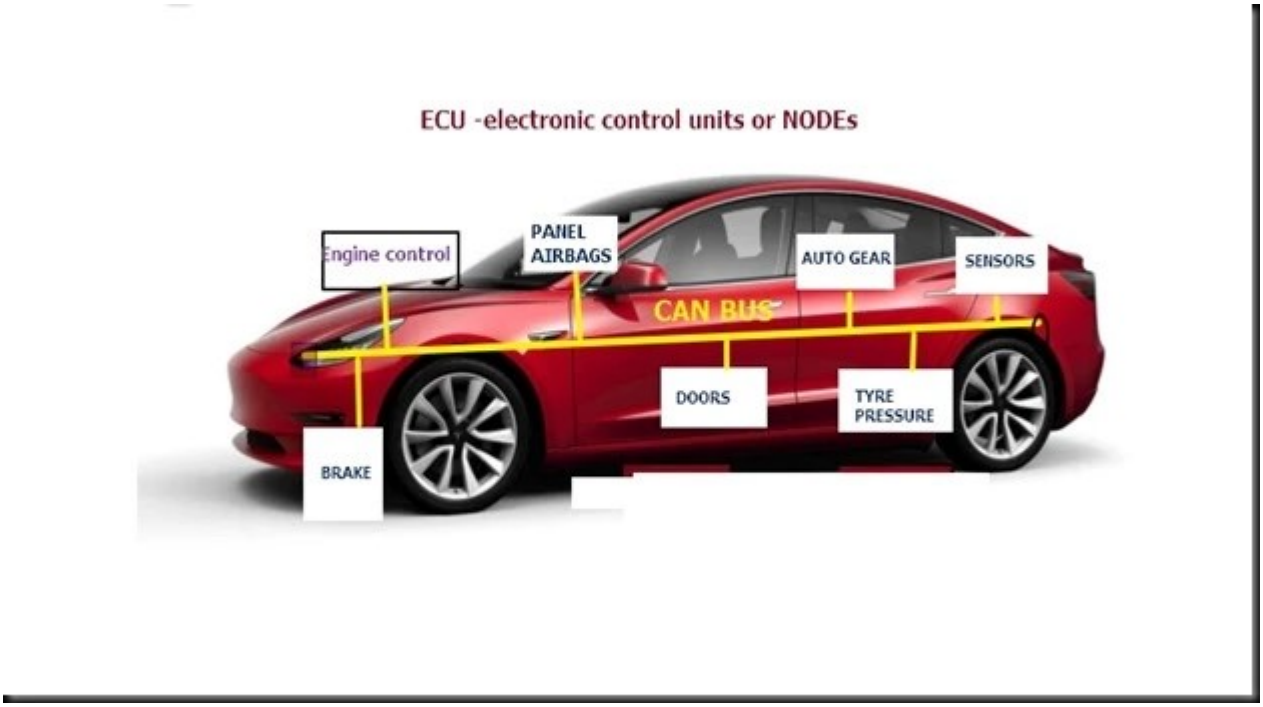
CAN – Controller Area Network bus is widely used in Automobile industry and in other industrial applications.

CAN is like Nervous system in Human body facilitating communication between all parts of body.

In CARs , NODEs or ECUs – Electronic Control Units are connected via CAN bus which acts as a control nervous system.

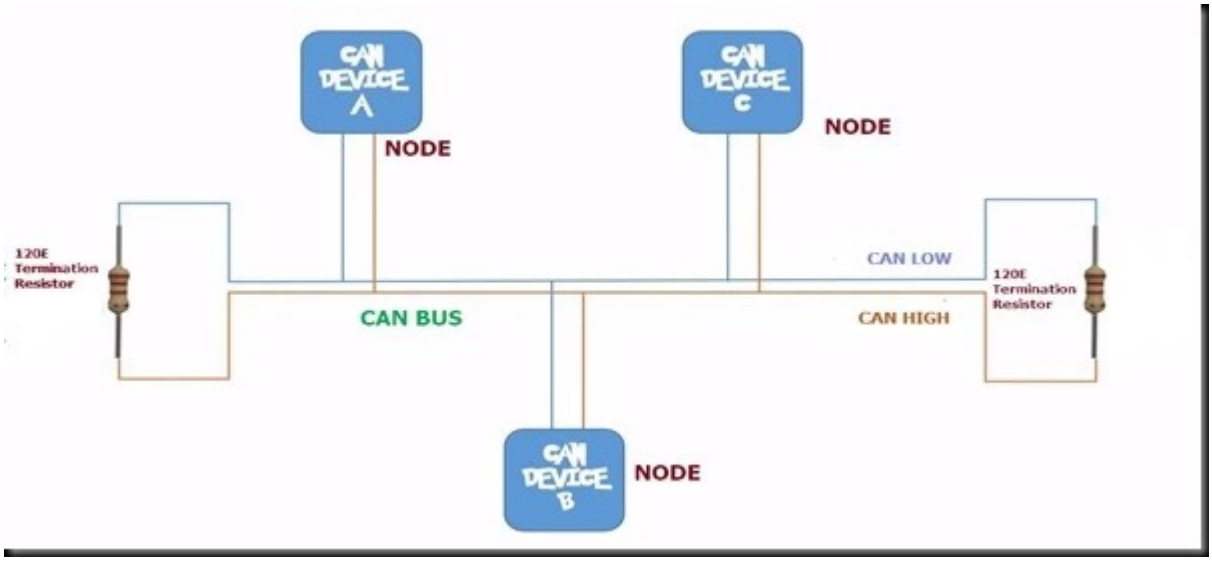
ECUs can be Engine control unit, Airbag control, Audi System , Door position,Brake system, Sensors & others.Each and every function can be assigned a NODE or ECU.

A modern CAR has upto 100 ECUs.



CAN allows ECUs to communicate with each other without complex dedicated wiring in between.

Any ECU can communicate with entire system without causing overload to the controller computer.



CAN is MULTI MASTER setup.Any Node can be a MASTER.

CAN works on a MESSAGING TYPE SYSTEM & no slave address is used to communicate with a node.

Every Node receives the message and the related data will be conceived by the receiver node.

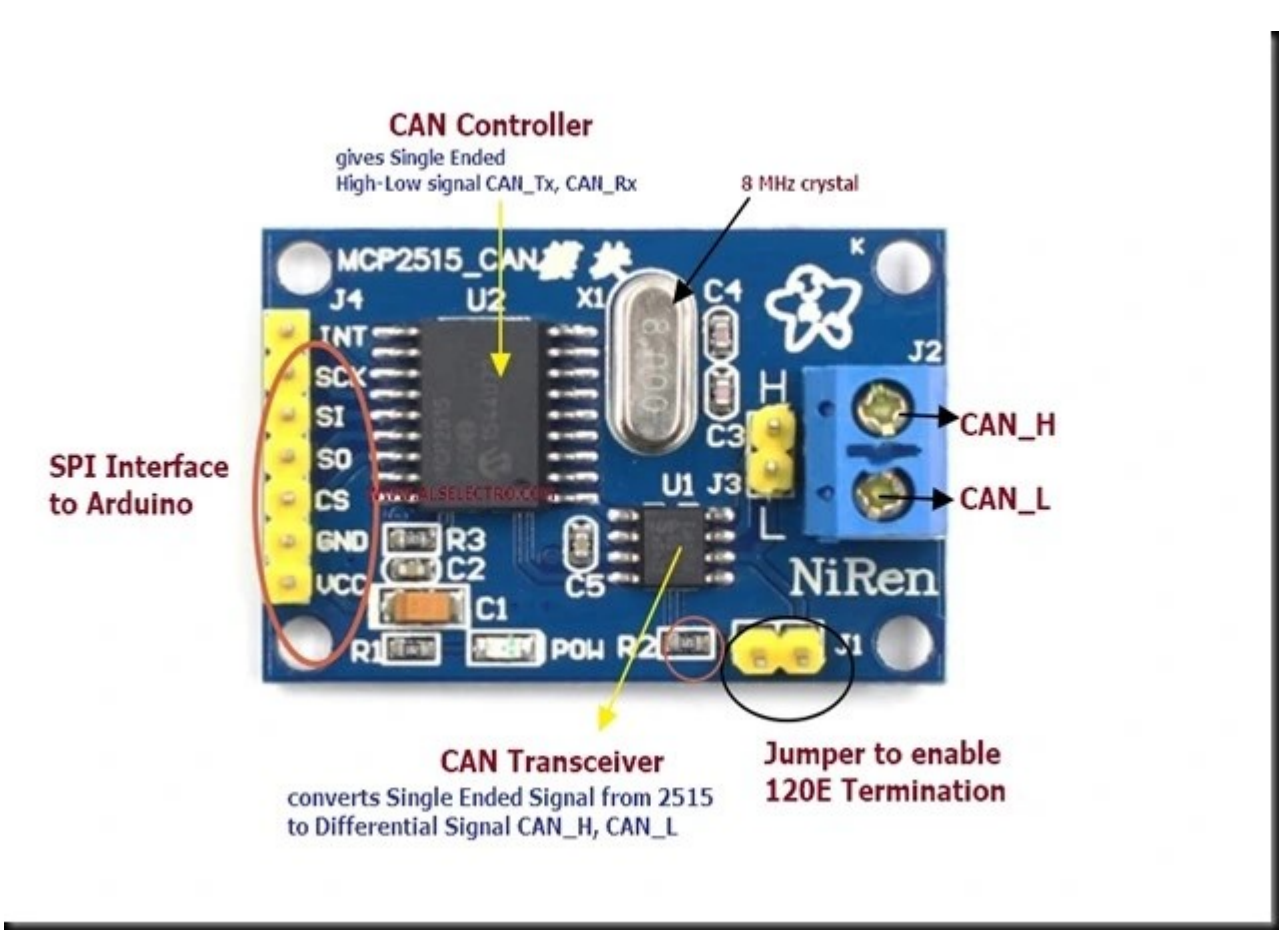
It is like public announcement system in an Airport.Only concerned passengers will take the message.

To balance the CAN BUS and for noise reduction a 120E RESISTOR is used as Terminator.This is must for END NODES. Nodes added in between need not use this Resistor.

In this project we make use of a CAN Board CAN2515.

This board has 2 ICs. One is the CAN Controller MCP2515 , which communicates with Arduino or any MCU by SPI Protocol.

MCP2515 gives SINGLE ENDED DIGITAL signal (High ,Low).



TX-CAN & RX-CAN are at digital levels and are connected to another IC , a TRANSCEIVER.

Here the Transceiver used is TJ1050 . Microchip MCP 2551 is also a transceiver , generally used with Microcontrollers which have inbuilt CAN Controllers.For e.g ESP32 module has in built controller , but it needs TRANSCEIVER like MCP2551 or TJ1050 .Do not confuse with the numbers

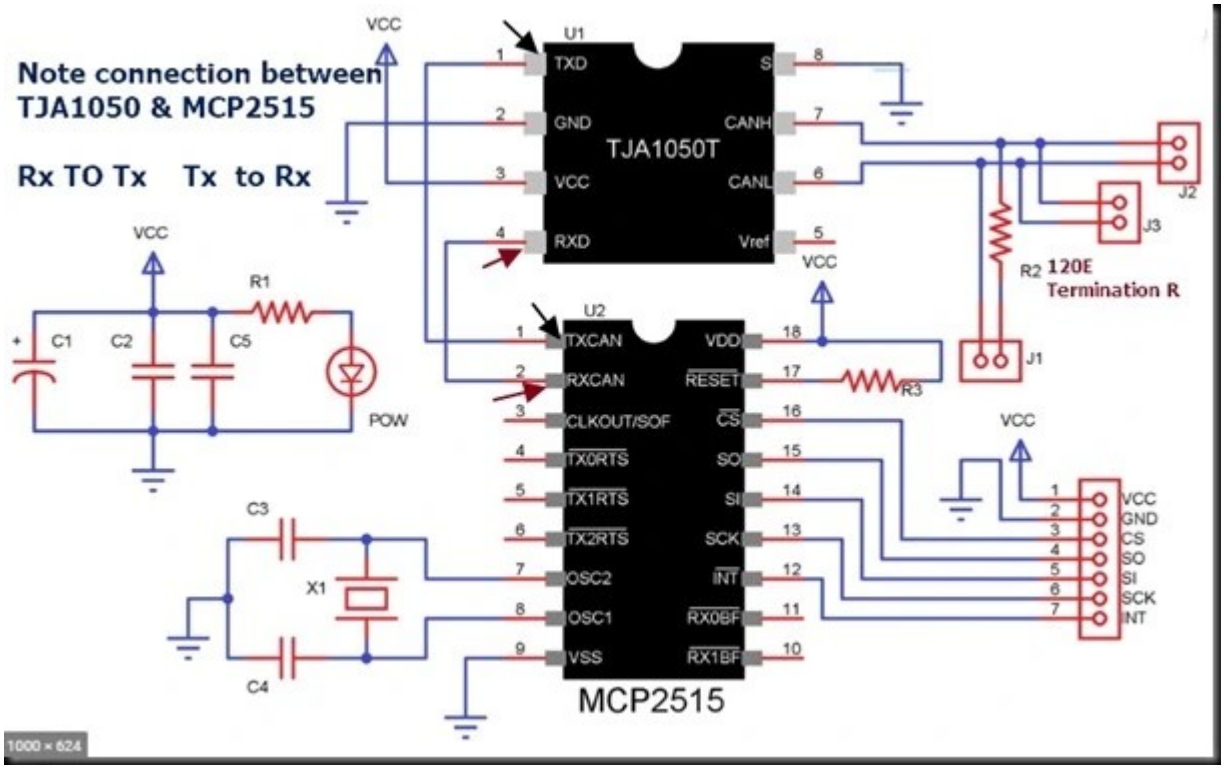
2515 IS CONTROLLER , 2551 IS TRANSCEIVER.

TJ1050 receives Digital signal from CAN2515 and converts tp DIFFERENTIAL SIGNAL acceptable on CAN BUS.

Note the connection between TJA1050 and CAN 2515 .

TX-CAN to TXD , RX-CAN to RXD (not reversed).

The module we use has both ICs on board and we need not care about this connection.



The CAN Message format is as below :

The CAN-ID may be 11 bit or 29 bit.

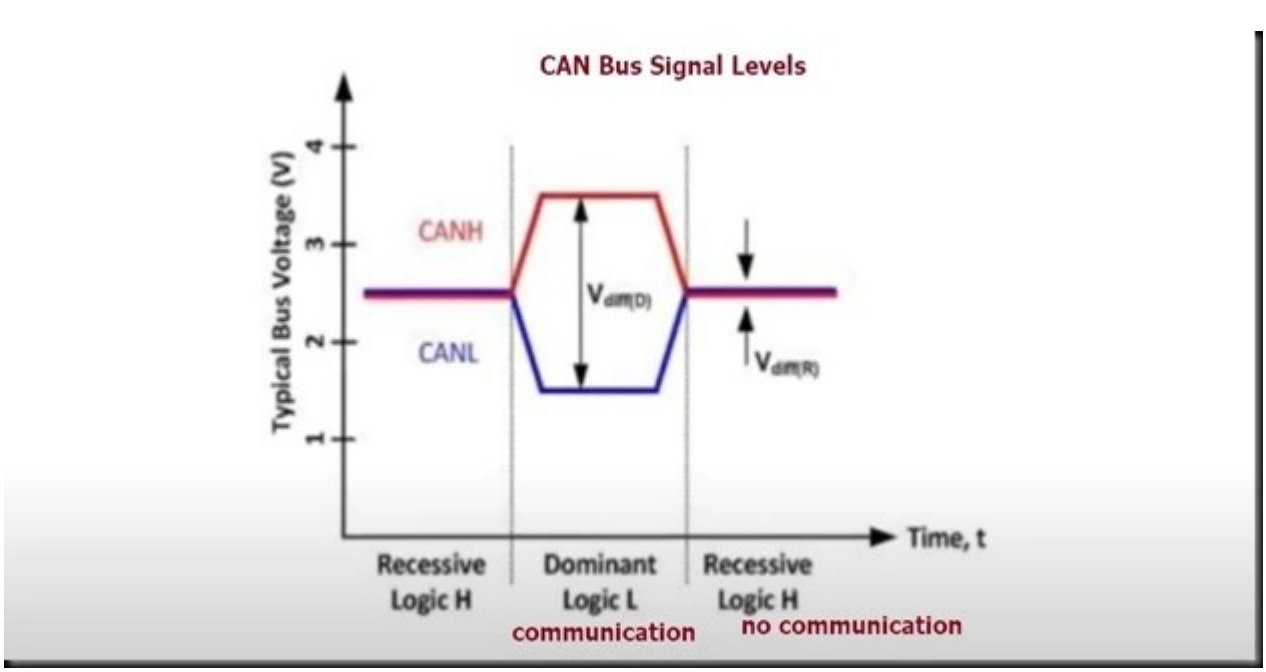
11 BIT can addresses 2048 nodes (2 power 11) hex 0x000 to 0x7FF

29 bit Extended can up to 536 million.

DAT length allowed is max 8 BYTES or 64 bits.

To learn more about CAN Message visit here

Following picture shows the Differential signal level of CAN.



When no information is on BUS , both CAN-H & CAN-L are at half the supply volt (here 2.5 volt on a 5v system).

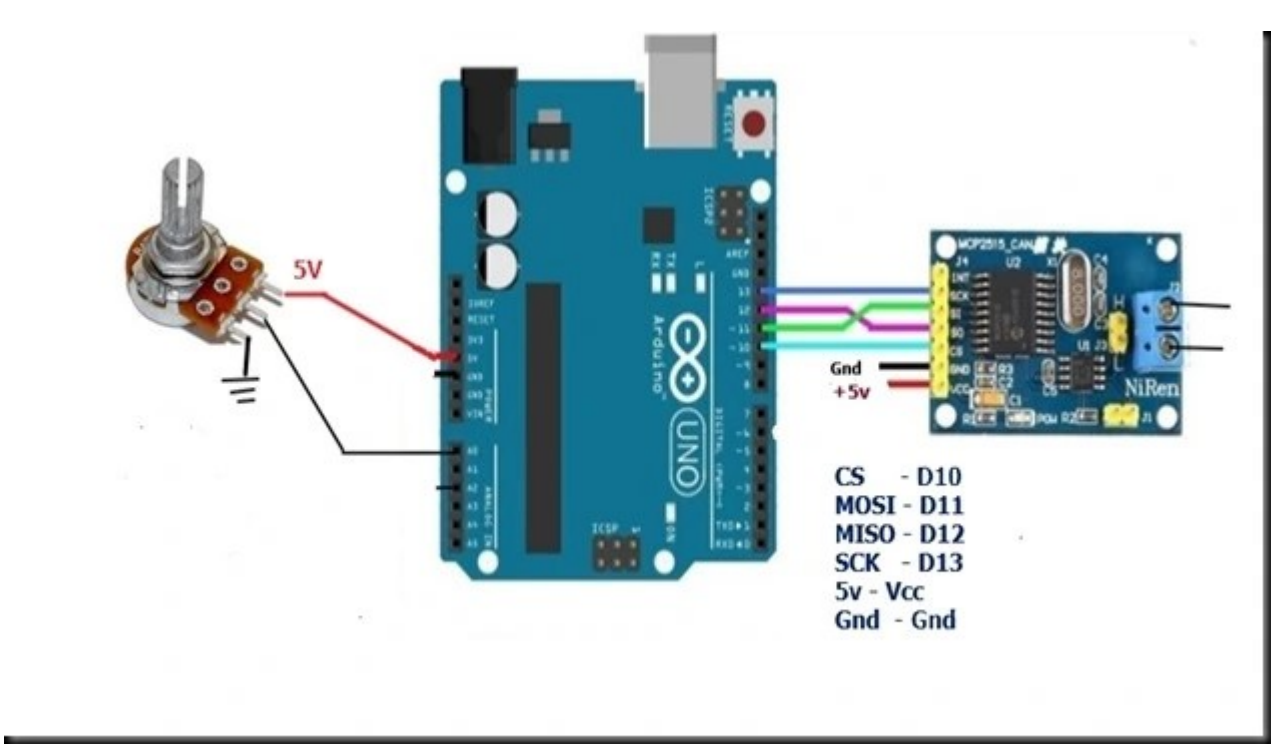
This is RECESSIVE State or BIT 1.

When CAN-H goes to 5v and CAN-L to 0 v , it is DOMINANT State or BIT 0.

To start with the project , connect a potentiometer (center pin) to A0 of Arduino.

One end of pot is connected to 5v and other end to Gnd.

The CAN board connections are as shown SPI connections between Arduino and CAN board.



Before starting we need a good Library for CAN to work with.

SEEED Studio has a Library which works fine with their Shield.

Also a modified version on it is available from Cryjfowler.

I prefer to use one from AUTOWP

arduino-mcp2515

which uses STRUCT for message sending.This is very simple and easy to use.

Please note , the Seed studio library and that of coryjfowler both have the same header file name mcp_can.h. If you install both these libraries , then there is clash on object declarations arising errors.Use any one only.

In this demo we use arduino-mcp2515 library.Download it as ZIP file.

CAN library

https://github.com/autowp/arduino-mcp2515

From Arduino IDE Sketch –> Include Library –> Add ZIP Library

and browse to the location of the downloaded library.Select it to install.

It will be copied to Documents/Arduino/Libraries

On the Receiver side we use an I2C LCD.

The library for the same can be downloaded and installed from :

i2c library :

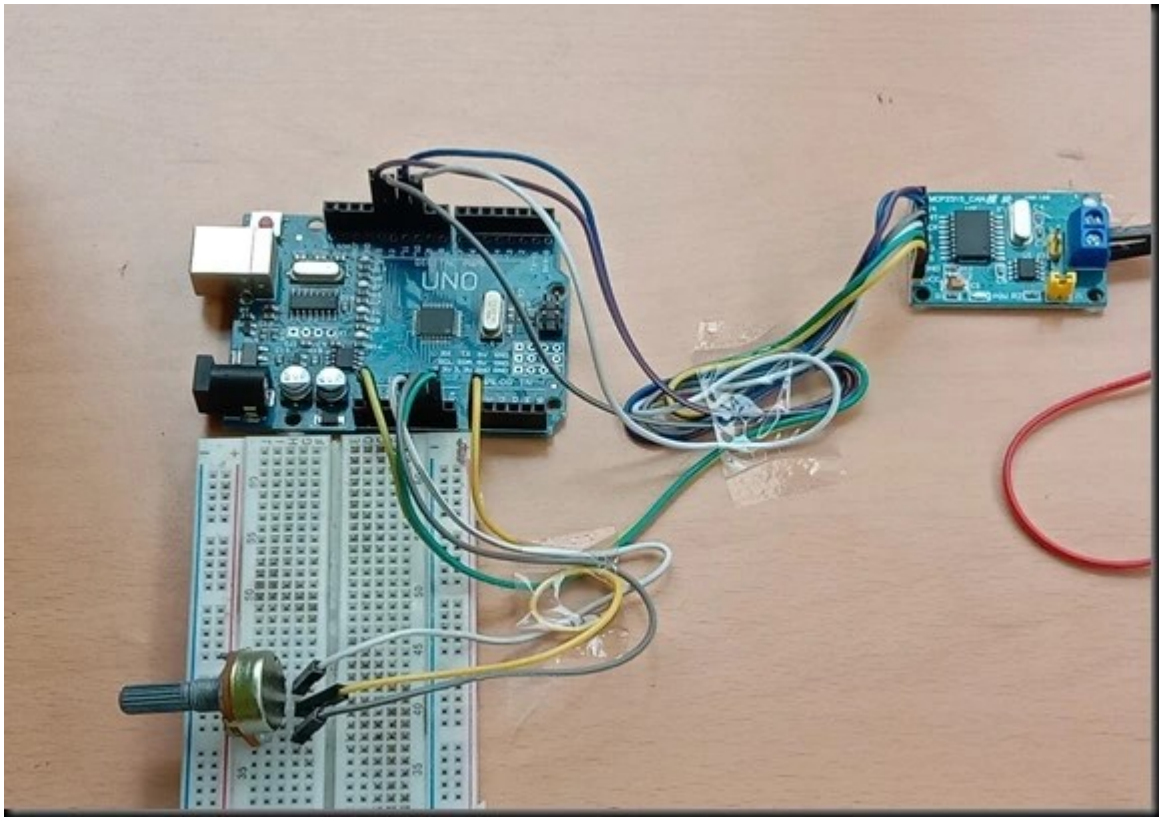
https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library

ARDUINO CODE used in this project can be downloaded from :

http://www.alselectro.com/files/CAN_2515.zip

Following picture shows the Transmitter side connections.

Potentiometer to A0 of Arduino , CAN board to SPI connections.



The code for the Transmitter side is :

File Edit Sketch Tools Help

pot_tx

```
#include <SPI.h>          //Library for using SPI Communication
#include <mcp2515.h>      //Library for using CAN Communication

#define potPin A0
int potValue=0;
//int tempValue=0;

struct can_frame canMsg;
MCP2515 mcp2515(10); // chip select CS pin 10

void setup()
{
  Serial.begin(9600);
  SPI.begin();               //Begins SPI communication

  mcp2515.reset();
  mcp2515.setBitrate(CAN_500KBPS,MCP_8MHZ); //Sets CAN at speed 500KBPS and C
  mcp2515.setNormalMode();

  canMsg.can_id  = 0x036;            //CAN id as 0x036
  canMsg.can_dlc = 1;                //CAN data length as 1 byte

}

void loop()
{
  potValue = analogRead(potPin);
  potValue = map(potValue,0,1023,0,255);
  Serial.println(potValue);

  canMsg.data[0] = potValue;         //Update pot value in [0]
  //canMsg.data[1]= 0x00;

  mcp2515.sendMessage(&canMsg);      //Sends the CAN message
  delay(200);
}
```

Here we include the mcp2515.h header file.

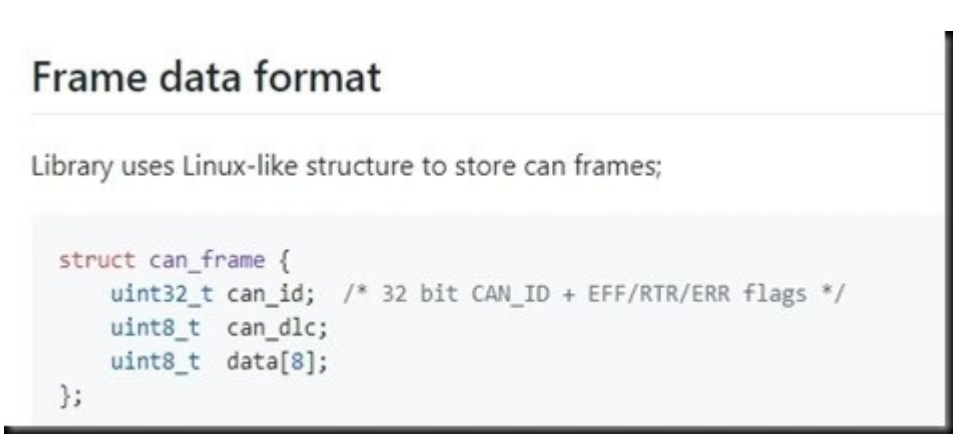
We create an un initialized struct can_frame variable called canMsg.

can_frame is the STRUCT Type name

canMsg is the Variable name.

Data members of struct is assigned using dot operator.

As seen from the library documentation , there are 3 members , can_id , can_dlc & data[].



mcp2515 object is initiated from class MCP2515 by passing the CS pin chip select pin 10 , as argument.

MCP2515 mcp2515(10);

Under set up we call the functions of object mcp2515

First reset() function is called.

Then setBitrate() to set the CAN spped at 500KBPS & frequency 8 MHZ.

setNormalMode() to set the module function in Normal mode.

mcp2515.reset();
mcp2515.setBitrate(CAN_500KBPS,MCP_8MHZ);

mcp2515.setNormalMode();

Using dot operator the members to STRUCT are added.

can_id is HEX ID which is user given.You can provide your own id in HEX.

can_dlc is the length of data.Maximum 8 BYTES allowed.Here we give data length as only one BYTE.

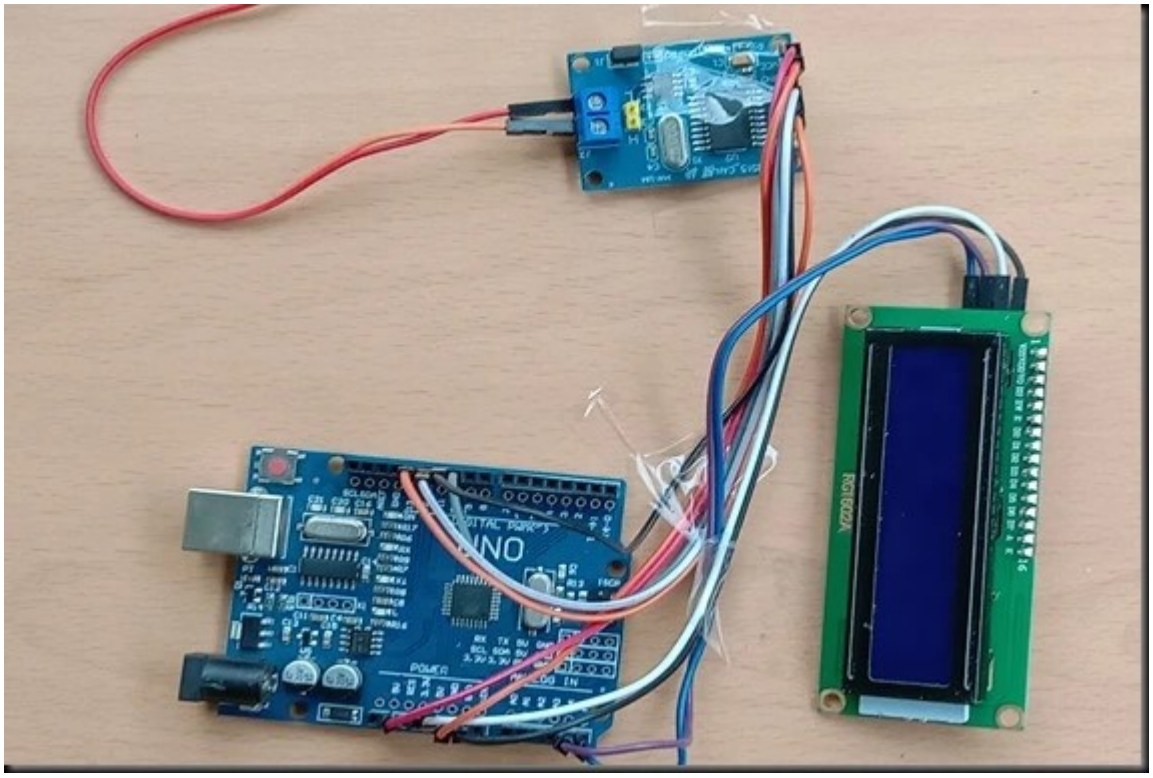
Under loop() , the analog pot value is mapped to value between 0 – 255

and the data sent over CAN bus using sendMessage() function of mcp2515.

Following is the Receiver side connection picture.

I2C LCD has 2 pins apart from Vcc & Gnd

SDA is connected to A4 , SCL is connected to A5.



The CAN board of Tx is connected to CAN board of Rx

CAN_H to CAN_H

CAN_L to CAN_L

Generally a twisted pair wire with shield is used.The shield is connected to GND.

Max length of 40 mtrs can be used.

Following is the Receiver side code :

pot_rx

```
#include <SPI.h>              //Library for using SPI Communication
#include <mcp2515.h>          //Library for using CAN Communication

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x3F,16,2);  // set the LCD address to 0x3F for a 16 chars and 2 line disp

struct can_frame canMsg;
MCP2515 mcp2515(10);                // SPI CS Pin 10

void setup() {
  lcd.init();       //initialize i2c LCD
  lcd.backlight();
  delay(1000);

  SPI.begin();                      //Begins SPI communication

  Serial.begin(9600);               //Begins Serial Communication at 9600 baudrate

  mcp2515.reset();

  mcp2515.setBitrate(CAN_500KBPS,MCP_8MHZ); //Sets CAN at speed 500KBPS and Clock 8MHz
  mcp2515.setNormalMode();                  //Sets CAN at normal mode
}

void loop()
{
  if (mcp2515.readMessage(&canMsg) == MCP2515::ERROR_OK) // To receive data (Poll Read)
  {
    if(canMsg.can_id==0x036)
    {
     int x = canMsg.data[0];
     Serial.println(x);
     lcd.setCursor(1,0);
      lcd.print("CAN Receiver...");
      lcd.setCursor(2,1);
      lcd.print("pot value:");
      lcd.print(x);
      delay(200);
     }
    }

}
```

Here we include the I2C LCD header.

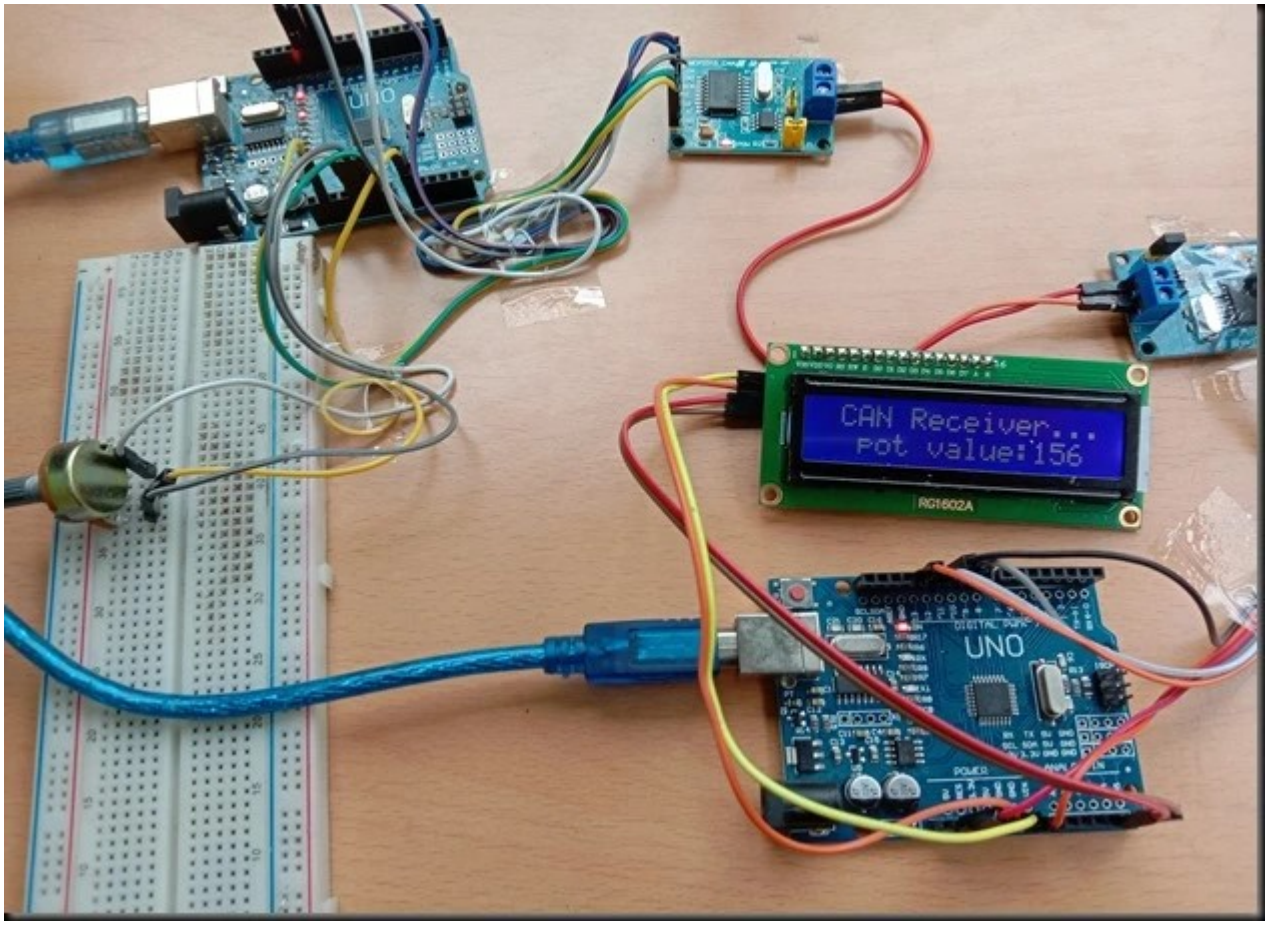
While creating lcd object we pass on the I2C address and the size of LCD (16 X 2)

To know the I2C address you can use i2c scanner from here

Under void loop

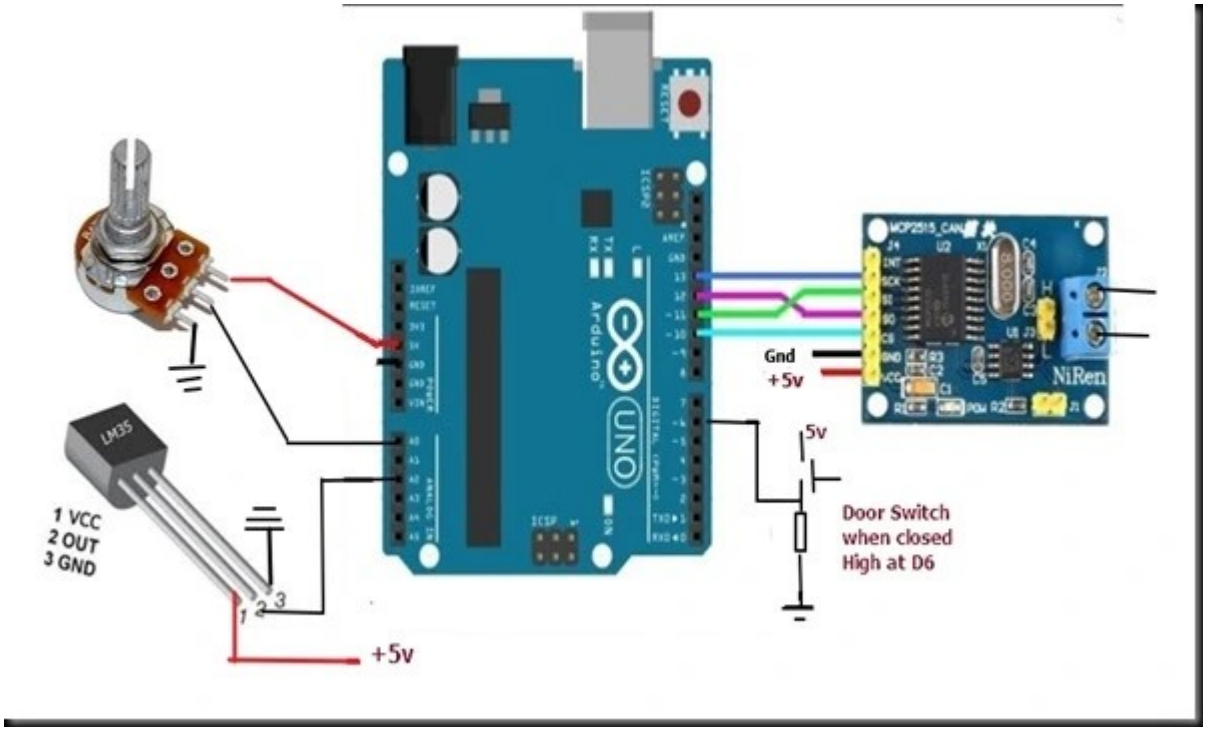
we use the readMessage() function of mcp2515 to verify the incoming message

& then we compare the id , then extract the message to display on LCD.



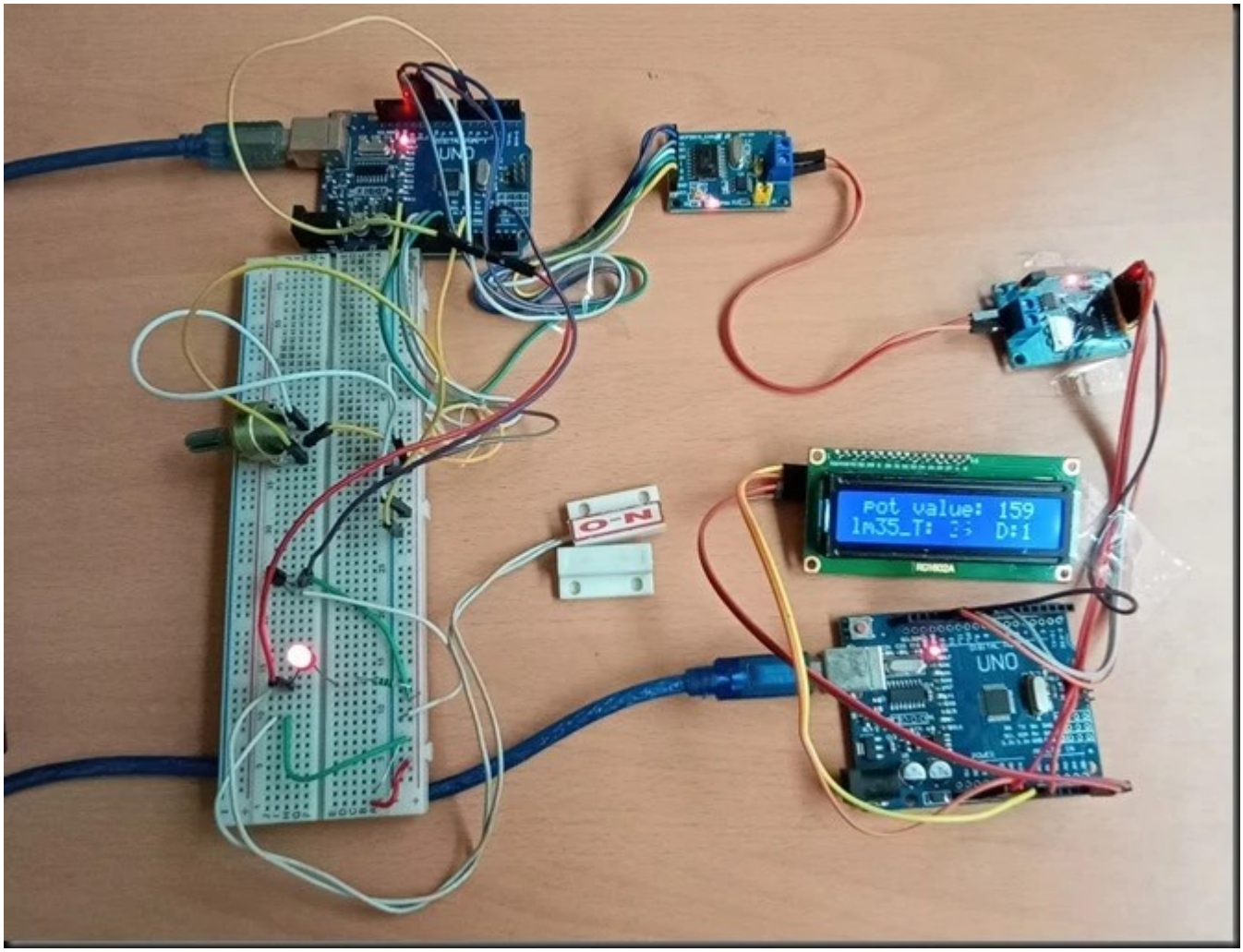
More sensors can be added to the project.

A LM35 temperature sensor at A3 and a Door ON/OFF switch can be added at D6.



ARDUINO CODE used in this project can be downloaded from :

http://www.alselectro.com/files/CAN_2515.zip



video :

Sensor Data over CAN BUS - Arduino with CAN2515

Tags: CAN, CAN2515

Comments RSS (Really Simple Syndication) feed

1 Comment:

- Area82
  September 4, 2020 at 4:02 am
  Thank you so much for taking your time to do this video. I had a couple issues. It needs to be said that it is important to know the address of your display or nothing happens. Mine was 0x27 Hex. You have to change your code to address this in the receiver.

  LiquidCrystal_I2C lcd(0x27,16,2) <—- changed from :

  // set the LCD address to "0x3F" for a 16 chars and 2 line display

  Using the display address finder sketch and saving what it reports to you it is most important

  Additionally the one like says lcd.init and it doesn't work but lcd.begin does.

  Reply