cyberelectronics commented 19 days ago • edited by PowerBroker2

@KoYoT77 I think we have similar issues.
Your data is there, after 7F.... bytes, but probable the parser stops after a few bytes.
One of my friends sent me a parser, which worked great for ~15seconds :) , after this the ESP32 module crashed (picture attached at the end, where you can see where is the issue, but I don't know how to fix it and he is busy now).
Here is the code for parsing:

```
void clearData() {
  dataFrame0 = "";
  dataFrame1 = "";
  dataFrame2 = "";
  dataFrame3 = "";
  dataFrame4 = "";
  dataFrame5 = "";
  dataFrame6 = "";
  dataFrame7 = "";
  dataFrame8 = "";

}

char* appendCharToCharArray(char* array, char c) {
  int len = strlen(array);

  char ret[len + 2];

  strcpy(ret, array);
  ret[len] = c;
  ret[len + 1] = '\0';

  return strdup(ret);
}

void addToFrame(int frame, char c) {
  switch (frame) {
    case 0:
      dataFrame0 = appendCharToCharArray(dataFrame0, c);
      break;
    case 1:
      dataFrame1 = appendCharToCharArray(dataFrame1, c);
      break;
    case 2:
      dataFrame2 = appendCharToCharArray(dataFrame2, c);
      break;
    case 3:
      dataFrame3 = appendCharToCharArray(dataFrame3, c);
      break;
    case 4:
      dataFrame4 = appendCharToCharArray(dataFrame4, c);
      break;
    case 5:
      dataFrame5 = appendCharToCharArray(dataFrame5, c);
      break;
```

```
      case 6:
        dataFrame6 = appendCharToCharArray(dataFrame6, c);
        break;
      case 7:
        dataFrame7 = appendCharToCharArray(dataFrame7, c);
        break;
      case 8:
        dataFrame8 = appendCharToCharArray(dataFrame8, c);
        break;
    }
  }


  // get substring between m and n (excluding n)
  char* substr(const char *src, int m, int n) {
    n += 1;
    int len = n - m;

    char *dest = (char*)malloc(sizeof(char) * (len + 1));

    for (int i = m; i < n && (*(src + i) != '\0'); i++)
    {
      *dest = *(src + i);
      dest++;
    }

    *dest = '\0';

    return dest - len;
  }

  void parse(char *raw) {
    int frame = -1;

    int len = strlen(raw);

    for (int i = 0; i < len; i++) {

      if (raw[i + 1] == ':') { //start frame
        frame = (int) raw[i] - '0';
        continue;
      }

      if (raw[i] == ':') {
        continue;
      }

      if (frame == -1) {
        continue;
      }

      if (raw[i] == '>') {
        frame = -1;
        continue;
      }

      addToFrame(frame, raw[i]);
    }
```

```
}

int convertToInt(char *hexStr) {
  return (int)strtol(hexStr, NULL, 16);

}
```

And here is how to use it in the **loop()**, with some **delay/timer** (in my case, this will return/print battery voltage):

```
myELM327.sendCommand("AT SH 7E4");      // Set Header BMS (Battery Management System)

if (myELM327.queryPID("220101")) {      // BMS PID
  read_rawdata();

  BATTv = ((convertToInt(substr(dataFrame2, 4, 5)) << 8) + convertToInt(substr(dataFrame2, 6,
  DEBUG_PORT.print("Main Battery Voltage (V): ");
  DEBUG_PORT.println(BATTv);
  clearData();
}
```

Read_rawdata function looks like this (will print the rawdata coming from OBD, then will parse it):

```
void read_rawdata() {
  // move data from OBD to Rawdata array
  char rawData[myELM327.recBytes];
  int n = 0;

  DEBUG_PORT.print("Payload received: ");

  for (int i = 0; i < myELM327.recBytes; i++) {
    rawData[n++] = myELM327.payload[i];
    DEBUG_PORT.print(myELM327.payload[i]);
  }
  DEBUG_PORT.println();

  parse(rawData);  // parse data received from OBD
}
```

Rawdata from OBD:

```
 Payload received:
7F22127F22127F22127F22127F221203E0:620101FFF7E71:FF8A00000000832:00240ED71713133:141713000010C1
4:22C12800008B005:00450B0000434B6:000019A80000187:1200200EE70D018:7B0000000003E8
```

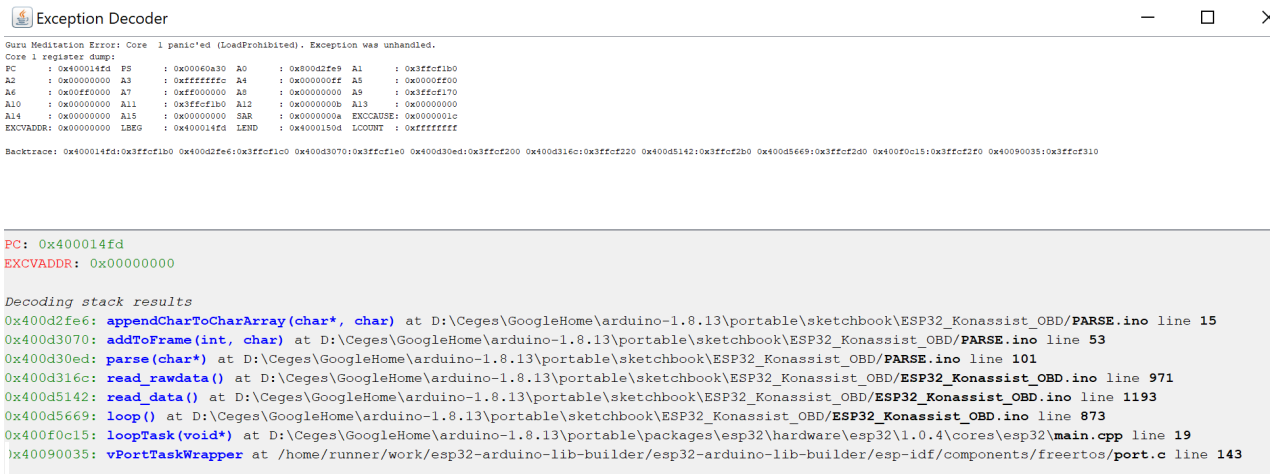
Parse(rawData), will look for dataframes, which begins with **0: 1: 2: 3:** etc... then I can select any byte in any dataframe:
Ex.:
Dataframe2 >>>>>>>>>>> `2:00240ED7171313`

`convertToInt(substr(dataFrame2, 4, 5))` >>>> will return a single byte on location 4 and 5 (from 0) in dataframe 2.
`convertToInt(substr(dataFrame2, 6, 7))` >>>> will return a single byte on location 6 and 7 (from 0) in dataframe 2.

Result >>>
0E =14;
D7 = 215;
Formula for Battery voltage = (14 * 256 + 215) / 10.0 = 379.9V

Now the problem is the parser will crash (possible memory leak) after 10-15seconds, until the crash, the parser works great.

Maybe **@PowerBroker2** can help both of us, with this parser :) .
As I know we should not use **malloc()** at all (or use it with **free()**).



cyberelectronics commented 16 days ago • edited by PowerBroker2

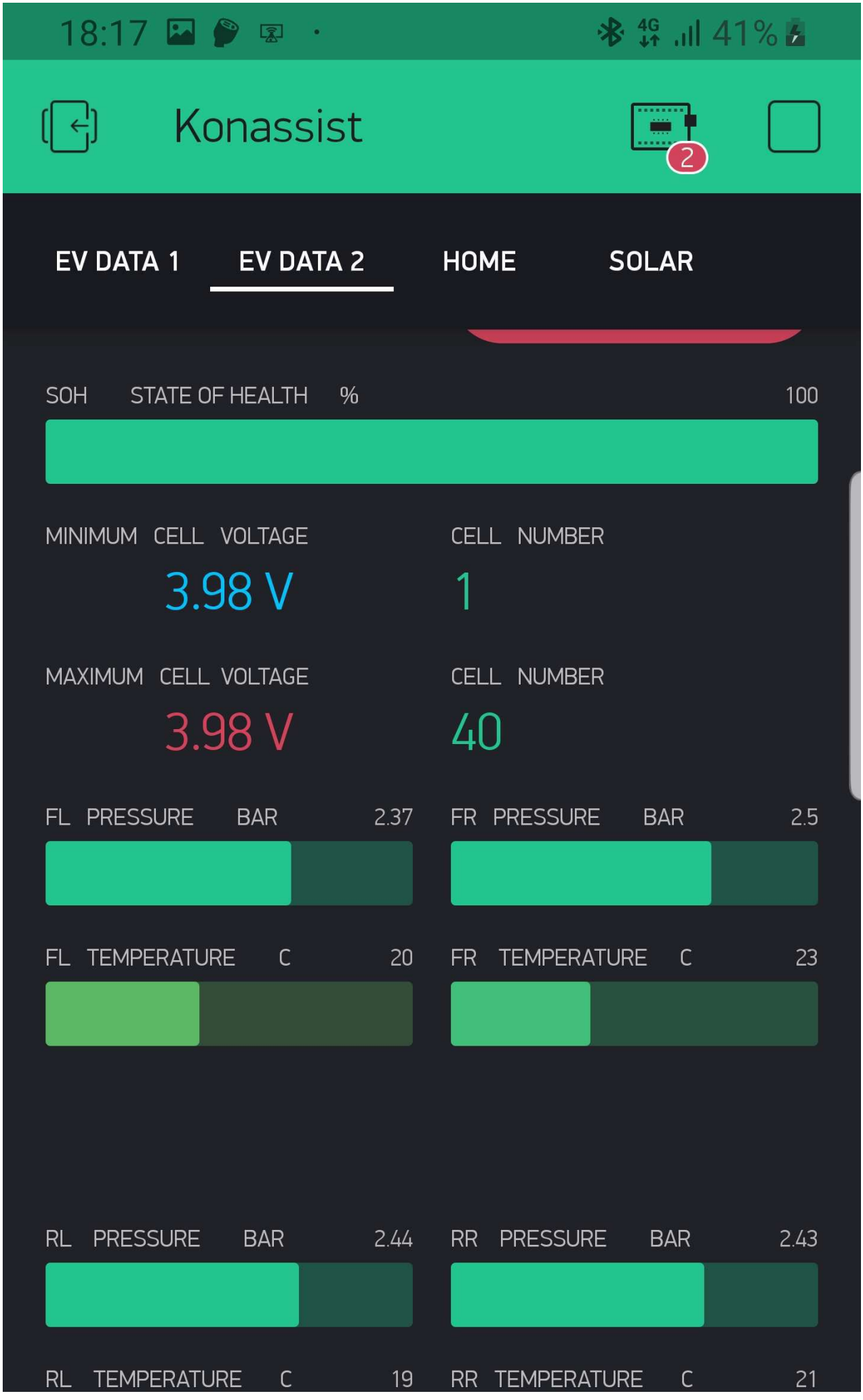
Fortunately, Matthew, who wrote the SafeString library, helped me with this issue.
Just finished a live test, while charging the car for 50 minutes.
No reboot or WiFi and BT connection issues at all.
OBD BT queried once/sec and the datas (in multiple groups) was sent every 2 seconds to Blynk.
So here is the working parser for OBD protocol 6 with multiple dataframes (from 0: 1: 2: to 8:).
I hope this will help others (maybe **@PowerBroker2** will add this to a new release)
Thanks guys!

```
#include "SafeString.h"

createSafeString(dataFrame0, 14);   // will also add space for the terminating null =>15 sized
createSafeString(dataFrame1, 14);
createSafeString(dataFrame2, 14);
createSafeString(dataFrame3, 14);
createSafeString(dataFrame4, 14);
createSafeString(dataFrame5, 14);
createSafeString(dataFrame6, 14);
createSafeString(dataFrame7, 14);
createSafeString(dataFrame8, 14);
```

```
void clearData() {
                    dataFrame0.clear();
                    dataFrame1.clear();
                    dataFrame2.clear();
                    dataFrame3.clear();
                    dataFrame4.clear();
                    dataFrame5.clear();
                    dataFrame6.clear();
                    dataFrame7.clear();
                    dataFrame8.clear();

}

void addToFrame(int frame, char c) {
        switch(frame) {
                case 0:
                        dataFrame0 += c;
                        break;
                case 1:
                        dataFrame1 += c;
                        break;
                case 2:
                        dataFrame2 += c;
                        break;
                case 3:
                        dataFrame3 += c;
                        break;
                case 4:
                        dataFrame4 += c;
                        break;
                case 5:
                        dataFrame5 += c;
                        break;
                case 6:
                        dataFrame6 += c;
                        break;
                case 7:
                        dataFrame7 += c;
                        break;
                case 8:
                        dataFrame8 += c;
                        break;
        }
}

void frameSubstr(SafeString& frame, int m, int n, SafeString& subStr) {
    frame.substring(subStr,m,n);   // SafeString substring is inclusive m to n
}

int convertToInt(SafeString& dataFrame, size_t m, size_t n) {
  // define a local SafeString on the stack for this method
    createSafeString(hexSubString, 14);  // allow for taking entire frame as a substring
    frameSubstr(dataFrame, m, n, hexSubString);
    return (int)strtol(hexSubString.c_str(), NULL, 16);
}

void parse(char *raw) {
        int frame = -1;
```

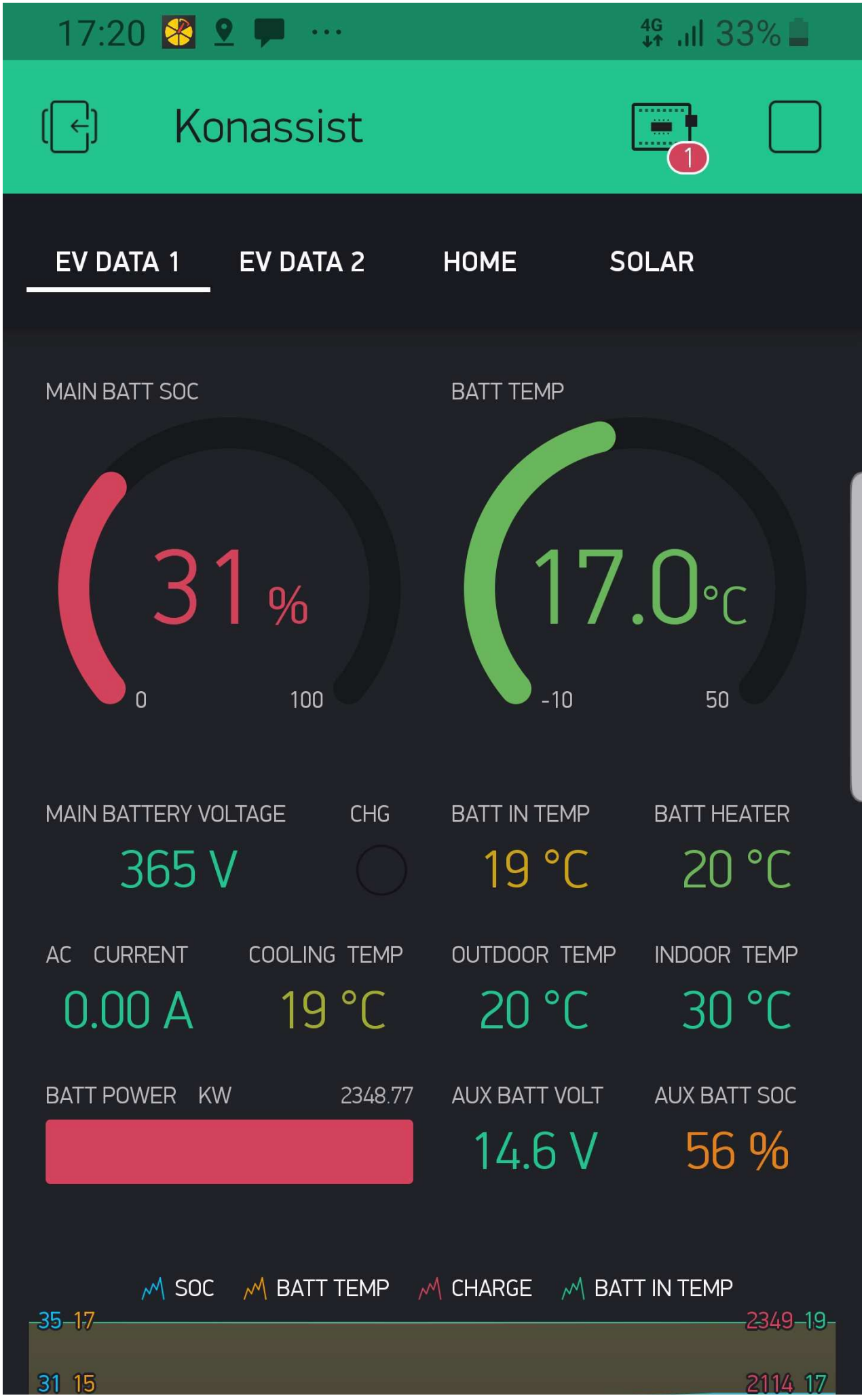
```
        int len = strlen(raw);

        for(int i=0; i<len; i++) {

                if(raw[i+1] == ':') { //start frame
                        frame = (int) raw[i] - '0';
                        continue;
                }

                if(raw[i] == ':') {
                        continue;
                }

                if(frame == -1) {
                        continue;
                }

                if(raw[i] == '>') {
                        frame = -1;
                        continue;
                }

                addToFrame(frame, raw[i]);
        }

}

void read_rawdata(){
                        // move data from OBD to Rawdata array
                        char rawData[myELM327.recBytes];
                        int n = 0;

                        DEBUG_PORT.print("Payload received: ");

                        for (int i=0; i<myELM327.recBytes; i++) {
                                                                rawData[n++] = myELM327.payl
                                                                DEBUG_PORT.print(myELM327.pa
                                                        }
                        DEBUG_PORT.println();

                        parse(rawData);  // parse data received from OBD

  }

// loop
void loop(){
             // insert timer/delay (ex. 1sec)
             ...
             myELM327.sendCommand("AT SH 7E4");       // Set Header BMS

             if (myELM327.queryPID("220101")) {      // BMS PID = hex 22 0101 => dec 34, 257

                          read_rawdata();
                          BATTv = ((convertToInt(dataFrame2, 4, 5)<<8) + convertToInt(dataFra
                          DEBUG_PORT.print("Main Battery Voltage (V): ");
                          DEBUG_PORT.println(BATTv);
                          clearData();
```

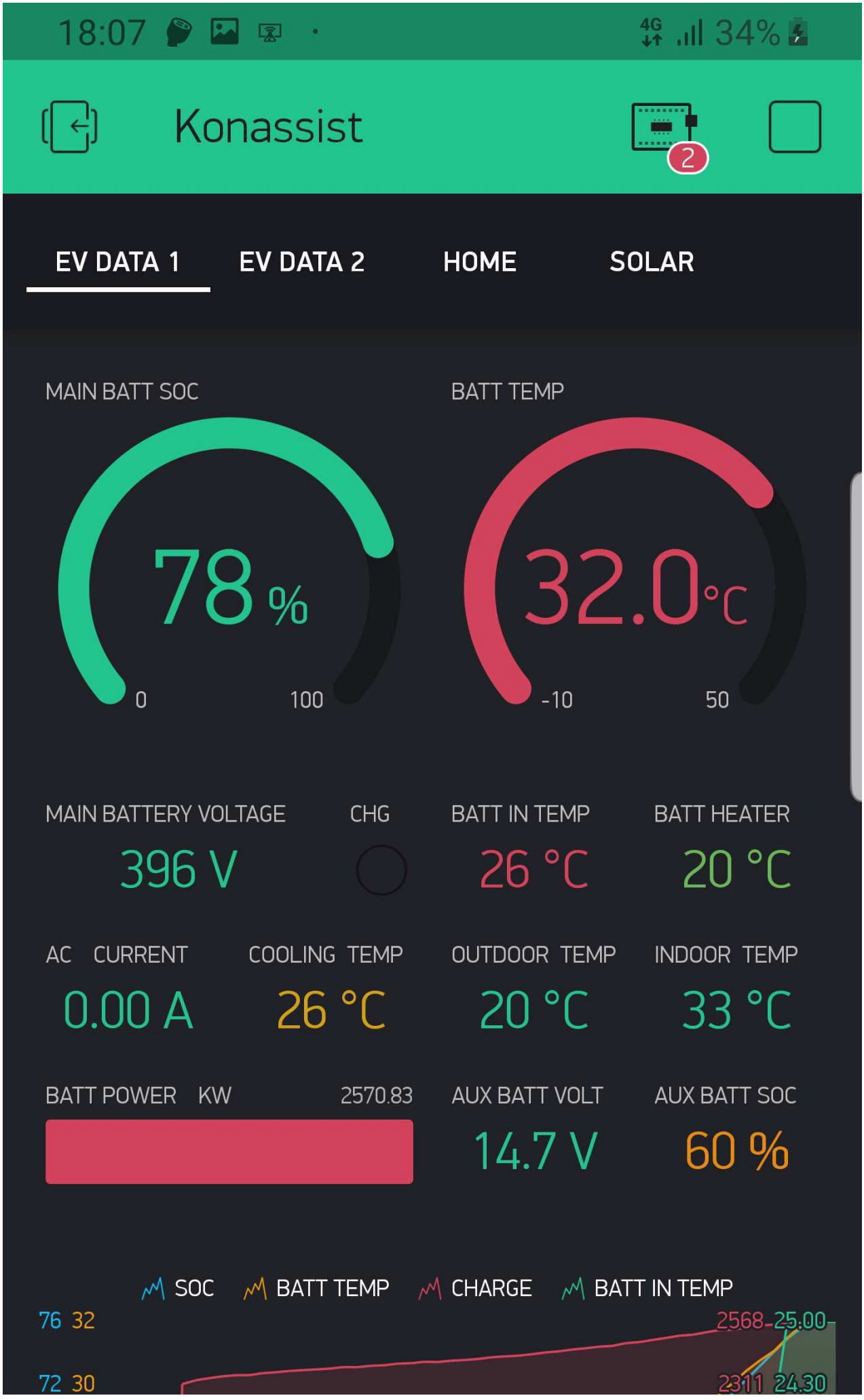
```
                        }
    }
```


18:17
41%
Konassist
2
EV DATA 1
EV DATA 2
HOME
SOLAR
SOH STATE OF HEALTH %
100
MINIMUM CELL VOLTAGE
3.98 V
CELL NUMBER
1
MAXIMUM CELL VOLTAGE
3.98 V
CELL NUMBER
40
FL PRESSURE BAR
2.37
FR PRESSURE BAR
2.5
FL TEMPERATURE C
20
FR TEMPERATURE C
23
RL PRESSURE BAR
2.44
RR PRESSURE BAR
2.43
RL TEMPERATURE C
19
RR TEMPERATURE C
21


TTGO
SOC %
50
Batt TempC
23.7


P
394 km
Remaining time
80%
3 min
78%
23 kW
3376 km


17:20
33%
Konassist
1
EV DATA 1
EV DATA 2
HOME
SOLAR
MAIN BATT SOC
31 %
0
100
BATT TEMP
17.0°C
-10
50
MAIN BATTERY VOLTAGE
365 V
CHG
BATT IN TEMP
19 °C
BATT HEATER
20 °C
AC CURRENT
0.00 A
COOLING TEMP
19 °C
OUTDOOR TEMP
20 °C
INDOOR TEMP
30 °C
BATT POWER KW
2348.77
AUX BATT VOLT
14.6 V
AUX BATT SOC
56 %
SOC
BATT TEMP
CHARGE
BATT IN TEMP
35 17
31 15
2349 19
2114 17


18:07
4G 34%
Konassist
2
EV DATA 1
EV DATA 2
HOME
SOLAR
MAIN BATT SOC
78 %
0
100
BATT TEMP
32.0°C
-10
50
MAIN BATTERY VOLTAGE
396 V
CHG
BATT IN TEMP
26 °C
BATT HEATER
20 °C
AC CURRENT
0.00 A
COOLING TEMP
26 °C
OUTDOOR TEMP
20 °C
INDOOR TEMP
33 °C
BATT POWER KW
2570.83
AUX BATT VOLT
14.7 V
AUX BATT SOC
60 %
SOC
BATT TEMP
CHARGE
BATT IN TEMP
76 32
72 30
2568 25.00
2311 24.30

PowerBroker2 commented 15 days ago Owner

This is dope! I'll take a closer look later and see if/how I can incorporate it.

However, **@cyberelectronics**, feel free to fork and PR if you want to do it yourself. Either way, thanks!

cyberelectronics commented 15 days ago

Thanks, but I don't know how to use github :D .
Also if you can, please add/modify the code for other cases, where the dataframe delimiters are missing (no 0: , when all datas are in a single frame package) like in **@KoYoT77** example.
Ex. PID >223275
Response: 62 32 75 4E
Skip any 7F... bytes at the beginning sent by OBD adapter, and Search/Compare for the first byte of PID 22hex + 40hex = 623275hex >>> 4E..... hex data bytes (dataframe).

KoYoT77 commented 15 days ago

Gents,
With Jaukb's help - our sketch works.
The issue was in this line of code:

```
sprintf(command, SET_HEADER, "7E0");
```

Elmduino.h const as below:

```
const char * const SET_HEADER = "AT SH"; // OBD
```

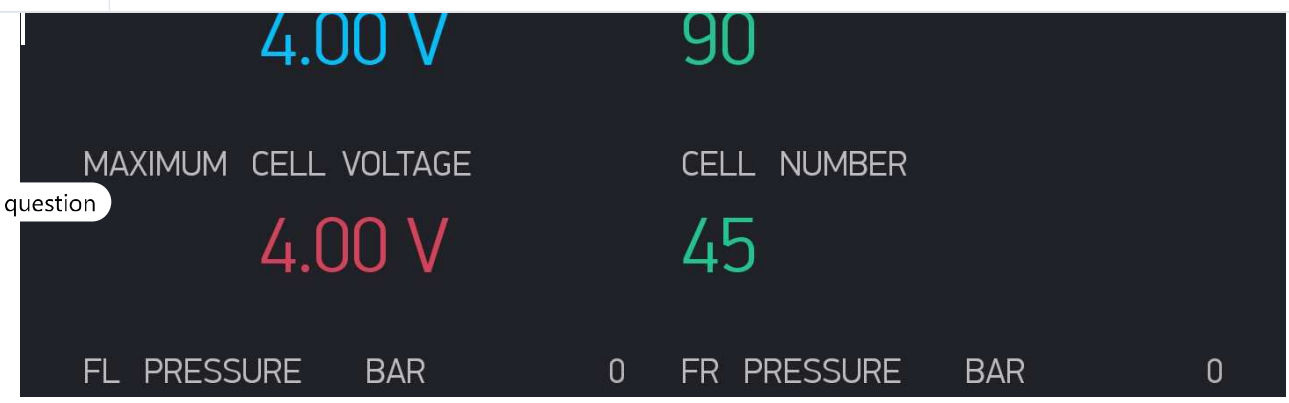
has been edited to this line

```
const char * const SET_HEADER = "AT SH %s"; // OBD
```

finally we have received proper data, same as in the App DPM Monitor.

Thanks to both of you and Jakub too!

👍 1



question

👍 1 ☺

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

4 participants