A. Informacje o zespole realizującym ćwiczenie

Nazwa przedmiotu:	Automatyka pojazdowa
Nazwa ćwiczenia:	Centralne systemy sterujące
Data ćwiczenia:	2019-05-29
Czas ćwiczenia:	08:00 – 09:30
Zespół realizujący ćwiczenie:	• Katarzyna Wątorska • Bartłomiej Mróz • Jacek Wójtowicz

B. Sformułowanie problemu

Celem zajęć było zbudowanie aplikacji od wysyłania i odbierania wiadomości w standardzie OBDII.

Należało zaprojektować w programie Canoe firmy Vector dwa węzły z czego jeden z nich miał służyć jako tester diagnostyczny, wysyłający określone zapytania poprzez magistralę CAN do elektronicznego modułu samochodowego, a drugi jako układ samochodu odpowiadający danymi na konkretne żądania.

Zapytania wysyłane przez tester miały być sterowane z panelu dającego możliwość ustawienia wartości poszczególnych sygnałów.

C. Sposób rozwiązania problemu

Kod testera diagnostycznego

```
variables
{
  message REQUEST_ID RQ_ID;
  message RESPONSE_ID RP_ID;
  message REQUEST RQ;
  message RESPONSE RP;
}

on sysvar OBD::BUTTON{
  OBD::ID='7DF';
  OBD::Unused = 0xaa;
  RQ_ID.ID =OBD::ID;
  RQ.LENGTH = OBD::LENGTH;
  RQ.MODE = OBD::MODE;
  RQ.PID = OBD::PID;
  RQ.Ah = OBD::Ah;
  RQ.Bh = OBD::Bh;
  RQ.Ch = OBD::Ch;
  RQ.Dh = OBD::Dh;
  RQ.Unused = OBD::Unused;
  output(RQ);
  output(RQ_ID);
}
```

Kod modułu odpowiadającego

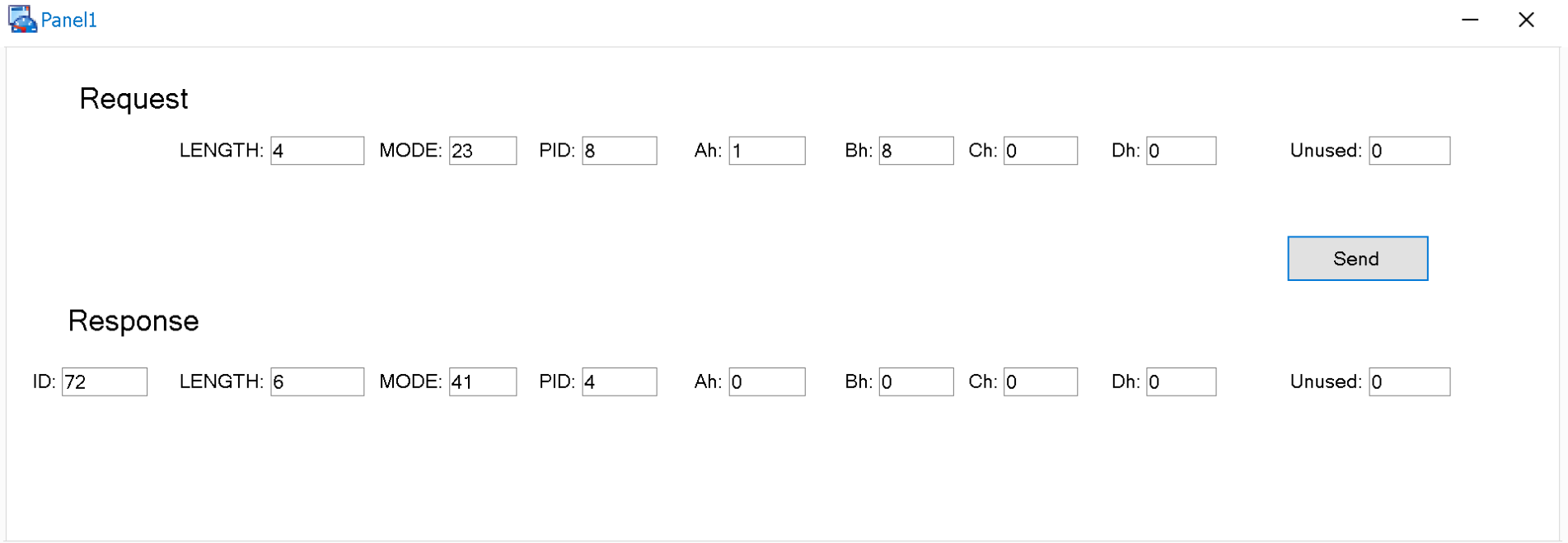
```
includes
{
  #include "TESTER.can"
}

variables
{
  int ENGINE_LOAD = 45;
  int ENGINE_COOLANT_TEMPERATURE = 120;
  double FUEL = 73;
  int FUEL_PRESSURE = 500;
  int INTAKE_PRESSURE = 45;
  double RPM = 12547.25;
  int SPEED = 212;
  int TIMING_ADVANCE = 14;
  int INTAKE_TEMP = 36;
  double MAF = 365.22;
  int THROTLE_POS = 75;
  message REQUEST_ID RQ_ID;
  message RESPONSE_ID RP_ID;
  message REQUEST RQ;
  message RESPONSE RP;
}
 on message REQUEST{
  OBD::ID='7E8';
  RP_ID.ID =OBD::ID;
  OBD::MODE = 0x41;
  OBD::Ch == 0xaa;
  OBD::Dh == 0xaa;
  if(OBD::PID == 0x04){
    OBD::PID == 0x04;
    OBD::LENGTH == 0x03
    OBD::Ah == ENGINE_LOAD*2.55;
    OBD::Bh == 0xaa;
  }
  if(OBD::PID == 0x0E){
    OBD::PID == 0x0E;
    OBD::LENGTH == 0x03
    OBD::Ah == ENGINE_LOAD*2.55;
    OBD::Bh == 0xaa;
  }
  if(OBD::PID == 0x0F){
    OBD::PID == 0x0F;
    OBD::LENGTH == 0x03
    OBD::Ah == ENGINE_LOAD*2.55;
    OBD::Bh == 0xaa;
  }
  if(OBD::PID == 0x10){
    OBD::PID == 0x10;
    OBD::LENGTH == 0x03
    OBD::Ah == ENGINE_LOAD*2.55;
    OBD::Bh == 0xaa;
  }
  if(OBD::PID == 0x11){
    OBD::PID == 0x11;
    OBD::LENGTH == 0x03
    OBD::Ah == ENGINE_LOAD*2.55;
    OBD::Bh == 0xaa;
  }
  RP.LENGTH = OBD::LENGTH;
  RP.MODE = OBD::MODE;
  RP.PID = OBD::PID;
  RP.Ah = OBD::Ah;
  RP.Bh = OBD::Bh;
  RP.Ch = OBD::Ch;
  RP.Dh = OBD::Dh;
  RP.Unused = OBD::Unused;
  output(RP);
  output(RP_ID);
}
```

Sygnały i wiadomości zdefiniowane w bazie danych

- REQUEST (0x0)
  - LENGTH
  - MODE
  - PID
  - Ah
  - Bh
  - Ch
  - Dh
  - Unused
- REQUEST_ID (0x2)
  - ID
- RESPONSE (0x1)
  - LENGTH
  - MODE
  - PID
  - Ah
  - Bh
  - Ch
  - Dh
  - Unused
- RESPONSE_ID (0x3)
  - ID

D. Wyniki

Panel testera diagnostycznego



Niestety nie udało nam się dokończyć aplikacji, aby zasymulować komunikację pomiędzy węzłami.

E. Wnioski

Dzięki temu ćwiczeniu zapoznaliśmy się z podstawami diagnostyki samochodowej, która wykorzystuje protokół diagnostyczny oparty na magistrali CAN. Dowiedzieliśmy się, że standard OBDII obecny w praktycznie każdym samochodzie poruszającym się po drogach jest ciekawym zagadnieniem w kontekście sieci CAN.

Co prawda zbudowana aplikacja nie pozwoliła do końca zasymulować komunikacji pomiędzy węzłami, ale przećwiczyliśmy tworzenie wiadomości i sygnałów w programie CANoe.