Beágyazott és ambiens rendszerek laboratórium

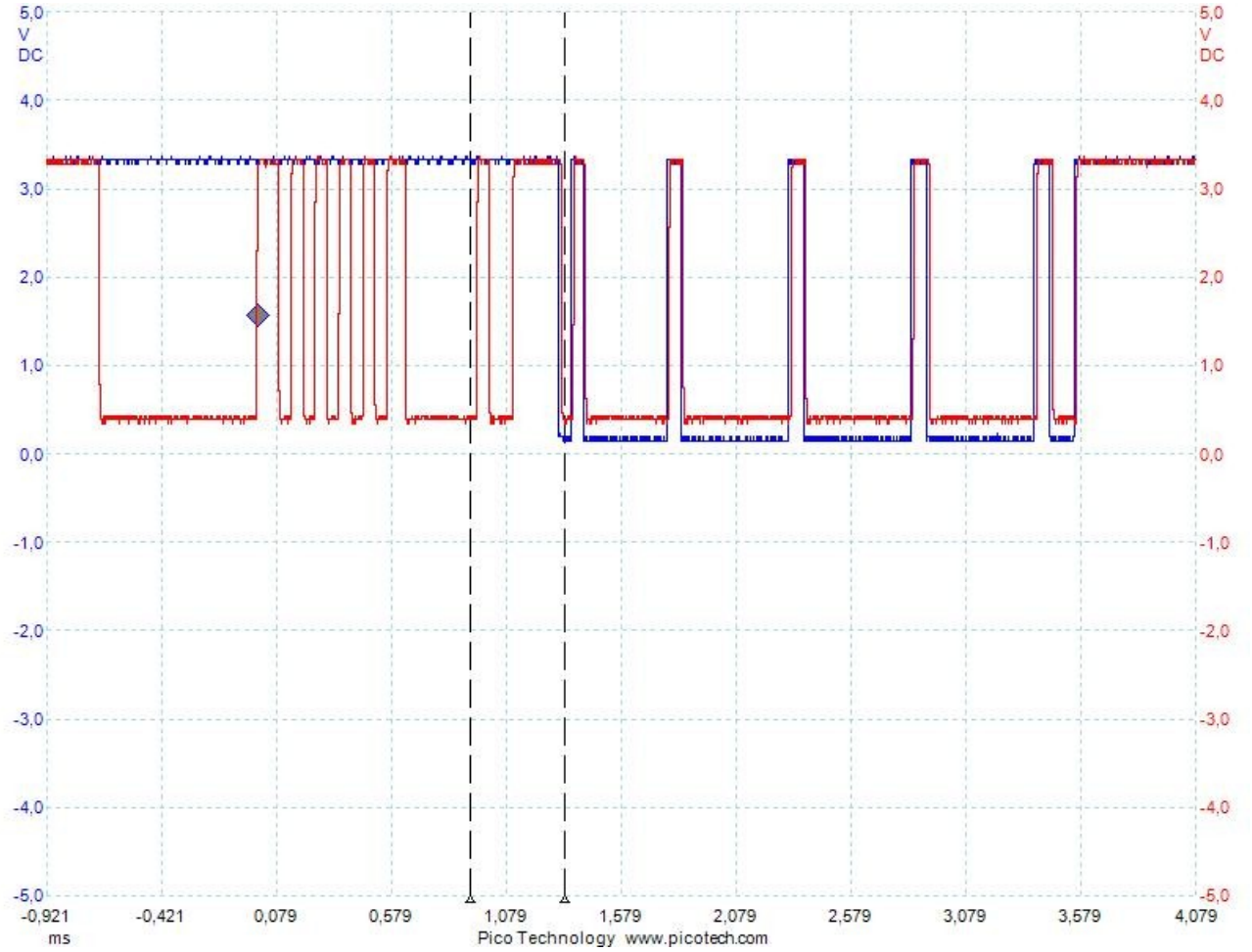
Jegyzőkönyv

7. mérés (LIN)

Készítette: Mező Dániel és Sipos-Takáts Bence

Az első feladatban digitális oszcilloszkóppal megmértük a LIN-buszon a jelszinteket, így a minimális 845mV, a maximális pedig 11.69V. Emellett megmértük az Rxd és Txd értékeit, ezek rendre max: 3.3V, min: 0.61V; max: 3.3V, min: 0.2V. Ezt követően a bitsebességet határoztuk meg, mégpedig úgy, hogy megmértük egy bit hosszát időben, majd ennek reciprokaként a frekvenciát. Eredményül 19.16kHz-et kaptunk, tehát 19.16kbps a sebesség. A névleges érték 19.2kbps, így az eltérés 0.04kbps, ami elhanyagolható.

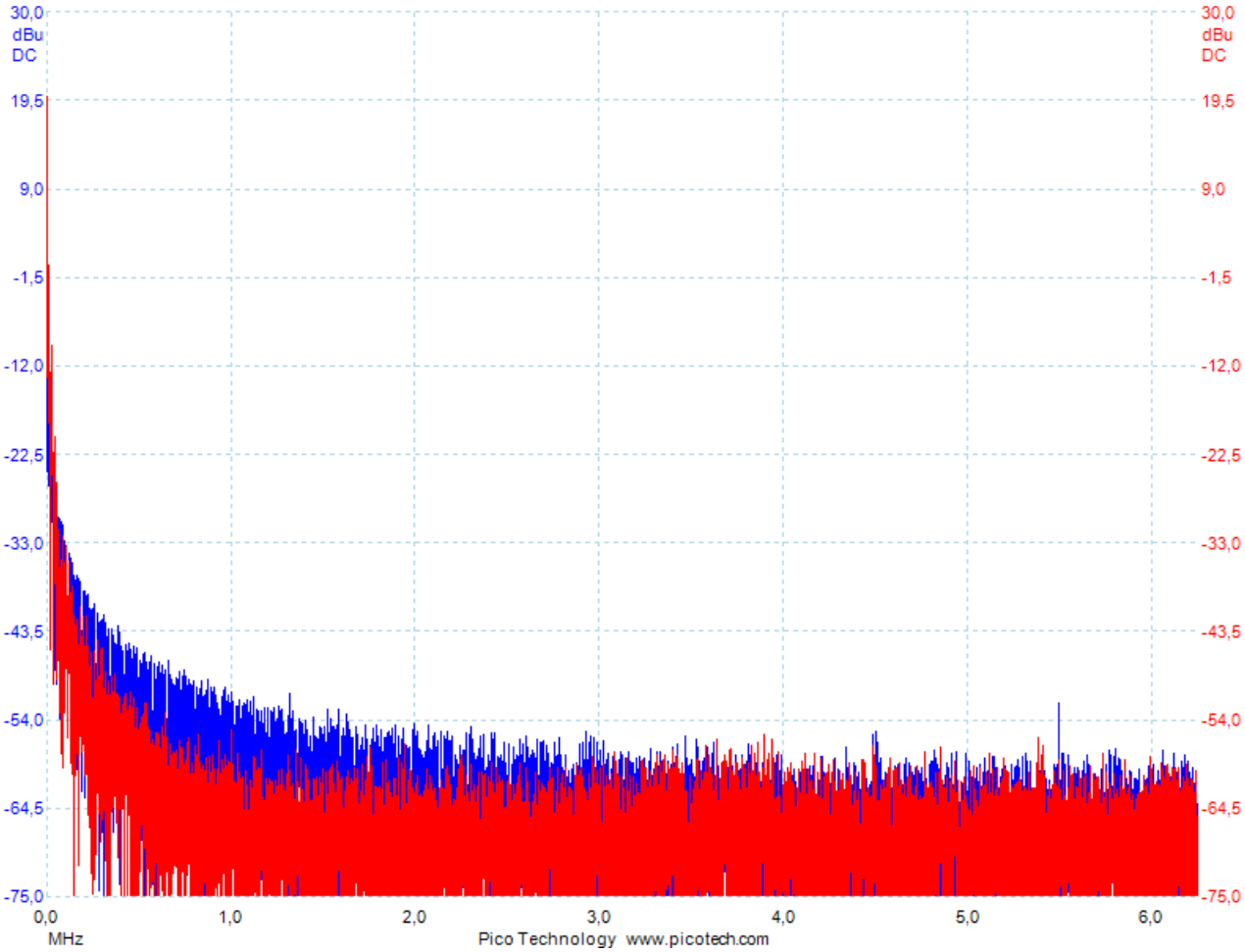
Következő feladatként egy üzenetet kellett visszafejteni, ezt a jegyzőkönyvben található Data Frame alapján csináltuk.



Az elején található 13 db 0, ez a break, utána meg egy szinkronizációs mező, mely a start bitre szinkronizál és értéke 0x55. Ennek célja, hogy a kommunikáció során az egységek órajelei szinkronban legyenek. Következő mező az azonosító, mely tartalmaz 4 bitnyi egyedi azonosítót, értéke 0x00, 2 bitnyi adatot slave task válaszának hosszáról és 2 paritásbitet. Ezután jön a tényleges adatmező. Látszik, hogy jelen esetben 4 byte-ot tartalmaz. Mindegyik egy startbittel kezdődik (0-s) és egy stopbittel végződik (1-es), UART-os kommunikációnak megfelelően. Az adat első bitje az LSB, az adat 0x10. A többi byte 0. Végén található egy ellenőrzőösszeg, ami FF-re egészít ki, ez

most egy darab nulla, utána a többi 1-es.

Meghatároztuk az busz spektrumát, az oszcilloszkóp FFT funkciójával. Íme az ábra:



Ezután nekünk kellett egy rövid program megírásával kapcsolatot teremteni a mote-ok között LIN buszon keresztül. Feladat az egyik slave-re rákötött ablakemelő motor fel és le mozgatása volt. Először egy másik slave üzenetét fogja, és ez alapján vezérelje a motort. Nekünk csak a panelen lévő ledeket kellett kapcsolgatni, hiszen azok közvetlenül rá voltak kötve a motorvezérlőre, így ha a megfelelő led világított, a motor működésbe lépett. Ehhez meg kellett írni, hogy az egyik slave fogadja, a másik meg küldje a csomagokat. Ezután egy fokkal nehezebb feladatot kellett megoldani, mégpedig hogy az a master által küldött csomagokat is fogadja, vagyis hogy azzal is lehessen vezérelni. Ez szimulálja, hogy a hátsó ablakot elöl és hátul is le lehet húzni. Ehhez fel kellett venni a masternél még egy keretet 0x01 címmel és az ütemezőjében még egy feladatot. Ugyanígy a fogadó slave-nél is. Mikor ezt a feladatot is teljesítettük, egy összetett rendszert kellett létrehozni, konkrétan hogy CAN hálózatról jön egy csomag, azt veszi, CAN/LIN gateway-en átalakítja és már eddig megépített rendszerhez csatlakozva az ablakemelőt vezérli. Az autóban is úgy van megoldva, hogy a LIN hálózatok csak lokálisan vannak elhelyezve, ezek vannak felfűzve a CAN gerincre, és ezen keresztül – márha – kommunikálnak a modulok. Ennek a feladatnak a programkódjait tartalmazza a jegyzőkönyv, mivel ez épül minden előtte megírt programra.

A master kódja:

```
int main (void) {
  t_frame MESS_SET_SLAVE;
  t_frame MESS_SET_SLAVE2;
```

```
  U8 number_of_frame ;

  CAN_message rx_message;

  dpy_trm_s01__Init();
  mcp2515_init();

  //SET SLAVE FRAME
  MESS_SET_SLAVE.frame_id    = 0x00 ;
  MESS_SET_SLAVE.frame_size  = 4 ;
  MESS_SET_SLAVE.frame_type  = REMOTE_LIN_FRAME_TYPE;
  MESS_SET_SLAVE.frame_delay  = 1920 ;
  MESS_SET_SLAVE.frame_data  = Buf_SET_SLAVE;

 //SET SLAVE FRAME
  MESS_SET_SLAVE2.frame_id    = 0x01 ;
  MESS_SET_SLAVE2.frame_size  = 4 ;
  MESS_SET_SLAVE2.frame_type  = STANDART_LIN_FRAME_TYPE;
  MESS_SET_SLAVE2.frame_delay  = 1920 ;
  MESS_SET_SLAVE2.frame_data  = Buf_SET_SLAVE2;

  number_of_frame = 2;
  my_schedule.frame_message[0] = MESS_SET_SLAVE;
  my_schedule.frame_message[1] = MESS_SET_SLAVE2;
  my_schedule.number_of_frame = number_of_frame;

  //CONFIG_IO_PORTS();

  // Initialise LIN Controller
  lin_init();// Performs Initialisation of LIN Software Driver

  sei();          /* Interrupts globally enabled */

  while(1) {


        //dpy_trm_s01__7seq_write_number(Buf_SET_SLAVE[0], 0);
     // Buf_SET_SLAVE2[0] = DPY_TRM_S01__BUTTON_1_GET_STATE() +
2*DPY_TRM_S01__BUTTON_2_GET_STATE() + 4*DPY_TRM_S01__BUTTON_3_GET_STATE();
        can_receive_message(&rx_message);
        if (rx_message.id==0x100) {
            if (rx_message.data[5]==8) Buf_SET_SLAVE2[0]=3;
            if (rx_message.data[5]==1) Buf_SET_SLAVE2[0]=6;
            }
        if (rx_message.id==0x100 && rx_message.data[5]!=8 && rx_message.data[5]!
=1) Buf_SET_SLAVE2[0]=7;

        dpy_trm_s01__7seq_write_number(Buf_SET_SLAVE2[0], 0);

  }//while(1)
  return 0;
}//main
```

Itt látható a két külön definiált keret, az ütemező, a buttonok definiálása és a CAN üzenet fogadása. CANalizer-rel megvizsgálva a 0x100-as címen jött adatot kell fogadni, ha az üzenet 5-ös mezőjében 8 vagy 1 érkezik. Ezek ugyanis a laborautó kormányán lévő gombok címe és adata.

A LIN buszra kiküldött adat azért 3 és 6, hiszen mikor a gombok működését és a csomag tartalmát definiáltuk, úgy döntöttük, hogy a gombok állását folyamatosan elküldjük, és mivel alapállapotban a gombok fel vannak húzva 1-be, így bináris helyiértékekkel megszorozva alapvetően 7-et küld (1+2+4=7), de ha lenyomunk egy gombot, lehúzza 0-ba, ezért a szám csökken. Ha a jobb gombot nyomjuk, 1+2-t küld, ha a balt, 2+4-et. Ezért a 3 és a 6.

A slave 1 kódja:

```
  int main (void) {
  U8 number_of_frame ;

  //SET SLAVE FRAME
  MESS_GET_SLAVE.frame_id    = 0x00 ;
  MESS_GET_SLAVE.frame_size  = 4 ;
  MESS_GET_SLAVE.frame_type  = REMOTE_LIN_FRAME_TYPE ;
  MESS_GET_SLAVE.frame_data  = Buf_GET_SLAVE;

  number_of_frame = 1;
  my_schedule.frame_message[0] = MESS_GET_SLAVE;
  my_schedule.number_of_frame = number_of_frame;

  dpy_trm_s01__Init();
  // Initialise LIN Controller
  lin_init();// Performs Initialisation of LIN Software Driver

  sei();          /* Interrupts globally enabled */
  Buf_GET_SLAVE[0]=0;

  while(1) {
      Buf_GET_SLAVE[0] = DPY_TRM_S01__BUTTON_1_GET_STATE() +
2*DPY_TRM_S01__BUTTON_2_GET_STATE() + 4*DPY_TRM_S01__BUTTON_3_GET_STATE();
      dpy_trm_s01__7seq_write_number(Buf_GET_SLAVE[0], 0);
        _delay_ms(10);
  }
  return 0;
}
```

Végül a slave2 kódja, mely az üzenetet fogadja és a motort is vezérli:

```
int main (void) {

  U8 number_of_frame ;
  U8 temp;
  //SET SLAVE FRAME
  MESS_GET_SLAVE.frame_id    = 0x00 ;
  MESS_GET_SLAVE.frame_size  = 4 ;
  MESS_GET_SLAVE.frame_type  = STANDART_LIN_FRAME_TYPE ;
  MESS_GET_SLAVE.frame_data  = Buf_GET_SLAVE;

  MESS_SET_SLAVE2.frame_id    = 0x01 ;
  MESS_SET_SLAVE2.frame_size  = 4 ;
  MESS_SET_SLAVE2.frame_type  = STANDART_LIN_FRAME_TYPE;
  MESS_SET_SLAVE2.frame_data  = Buf_SET_SLAVE;

  number_of_frame = 2;
  my_schedule.frame_message[0] = MESS_GET_SLAVE;
  my_schedule.frame_message[1] = MESS_SET_SLAVE2;
  my_schedule.number_of_frame = number_of_frame;

  dpy_trm_s01__Init();

  // Initialise LIN Controller
  lin_init();// Performs Initialisation of LIN Software Driver

  sei();          /* Interrupts globally enabled */
  Buf_GET_SLAVE[0]=0;
  Buf_SET_SLAVE[0]=0;

 // Buf_SET_SLAVE[0]=0;
```

```
  while(1) {
      temp = Buf_GET_SLAVE[0] + Buf_SET_SLAVE[0];

        dpy_trm_s01__7seq_write_number(temp,0);

        DPY_TRM_S01__LED_3_ON();


        if(temp == 13 || temp == 12) DPY_TRM_S01__LED_2_ON();
            else DPY_TRM_S01__LED_2_OFF();
        if(temp == 10 || temp == 6) DPY_TRM_S01__LED_1_ON();
            else DPY_TRM_S01__LED_1_OFF();

  }
  return 0;
}
```

A 6, 10, 12 és 13 úgy jön össze, hogy a két üzenetet értékét összegzi és azt vizsgálja. Ugye egyik csomagban folyamatosan 7-et küld, így ha jön egy 3-as vagy egy 6-os, összeadva 10 vagy 13. De ha mondjuk mindkét modulon 6-ot küld, akkor 12, ugyanígy a 3-mal. Ezzel biztosítva van, hogy ne lehessen a két egységen szembevezérelni.

Budapest, 2010.05.03.