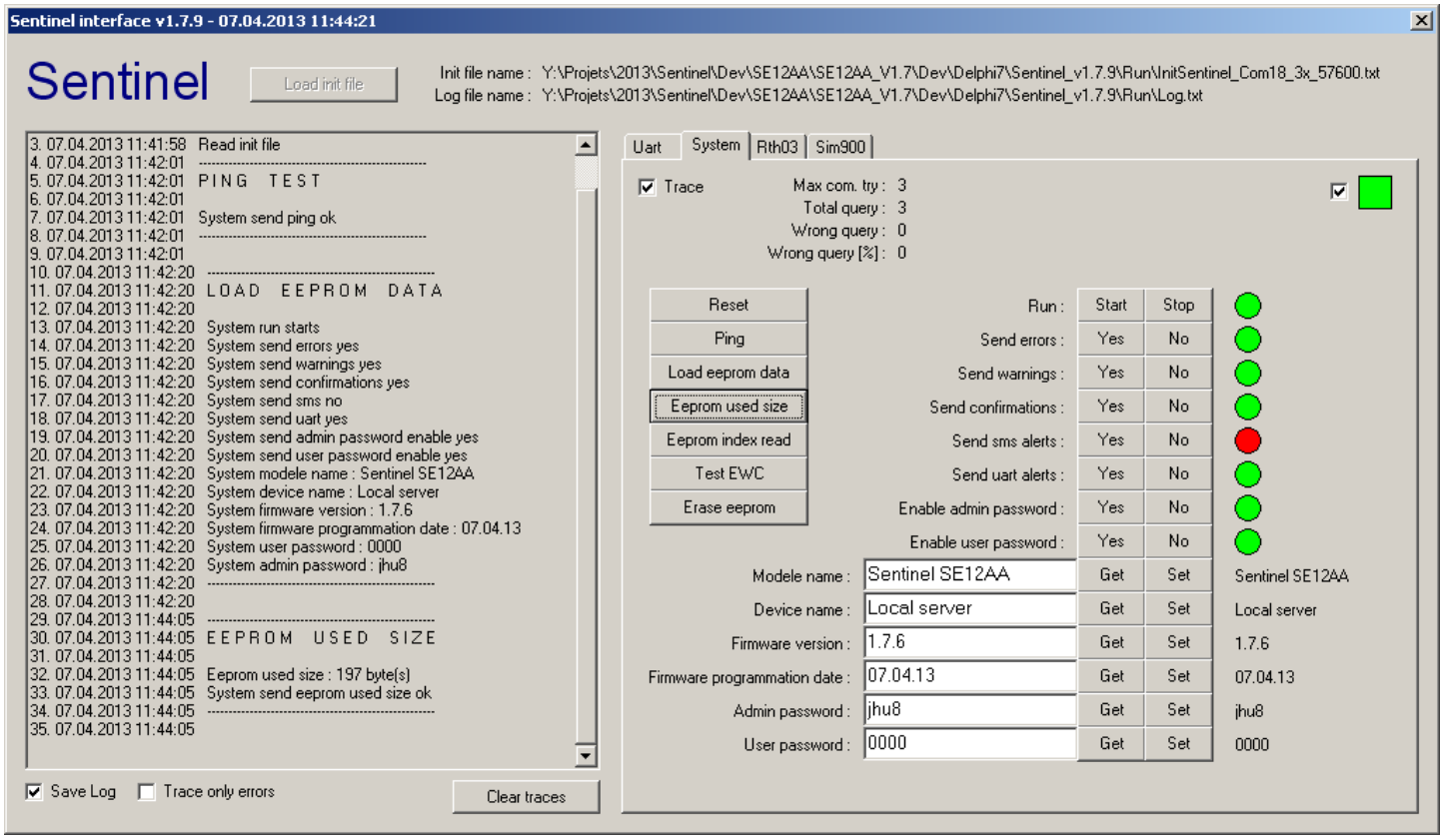
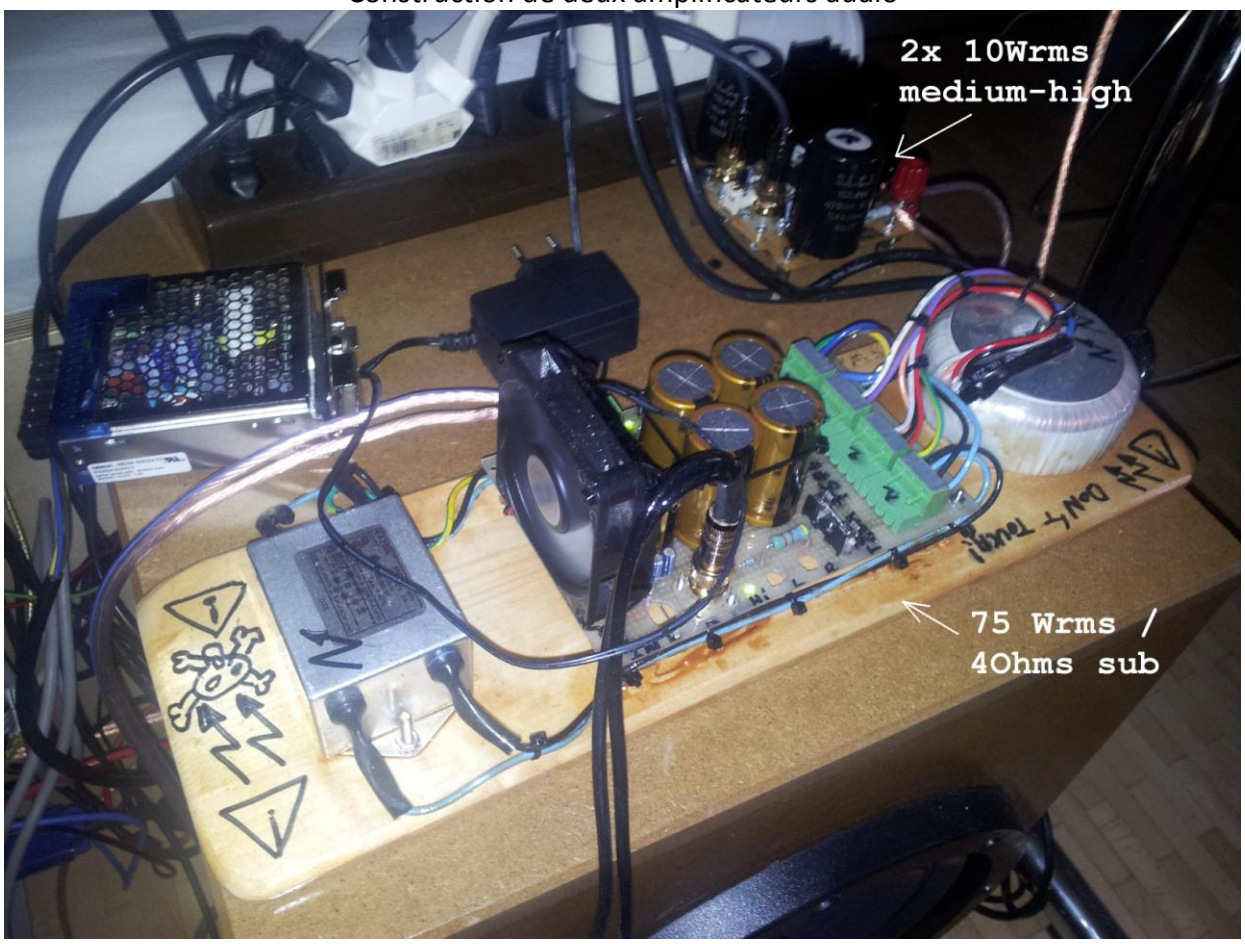
Surveillance humidité / température par GSM, gestion SMS et PHONEBOOK avec un PIC18F26K22 à 64Mhz

Driver PC via USB de « Sentinel »



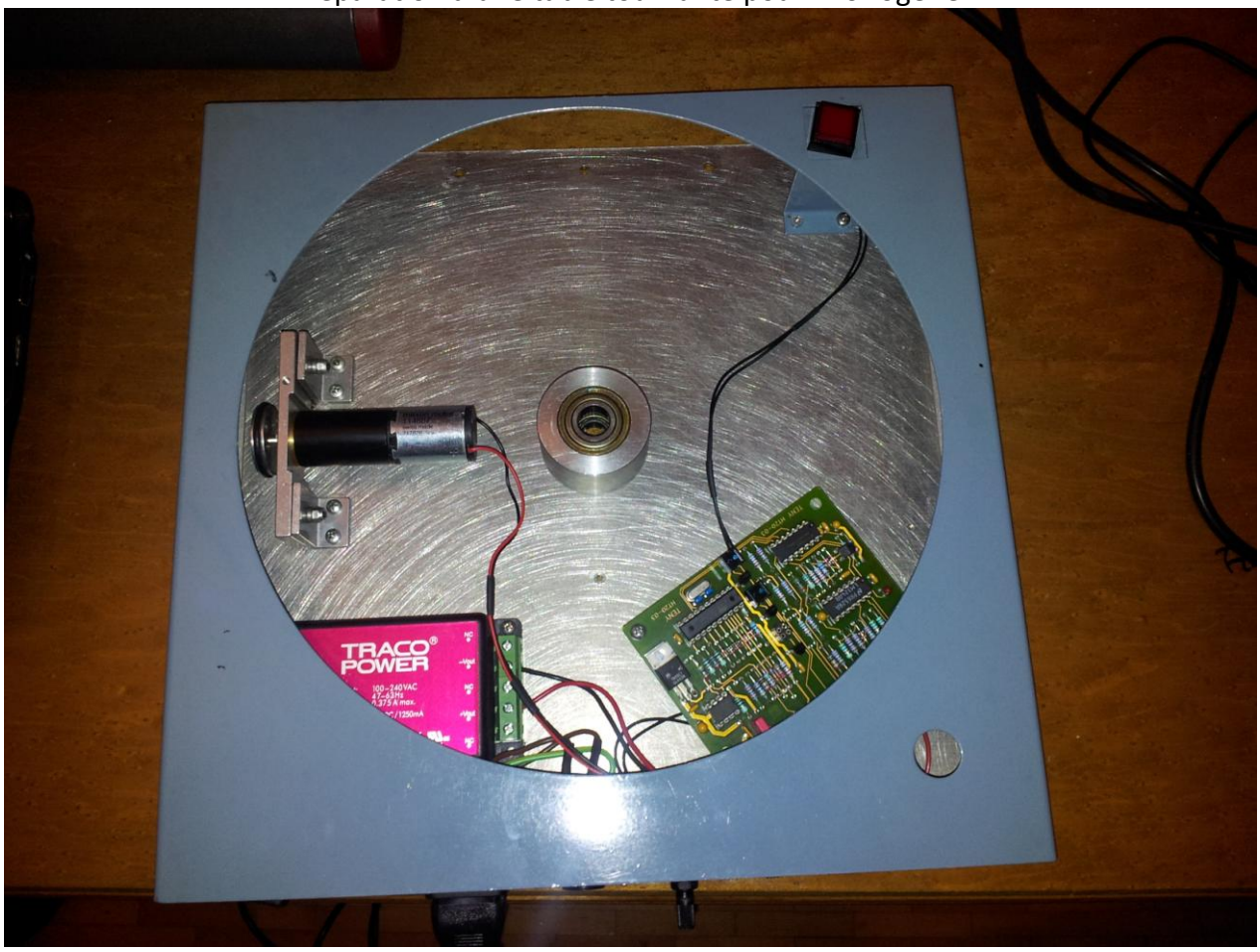
Construction d’un caisson de bass dans un atelier de menuiserie

Construction de deux amplificateurs audio

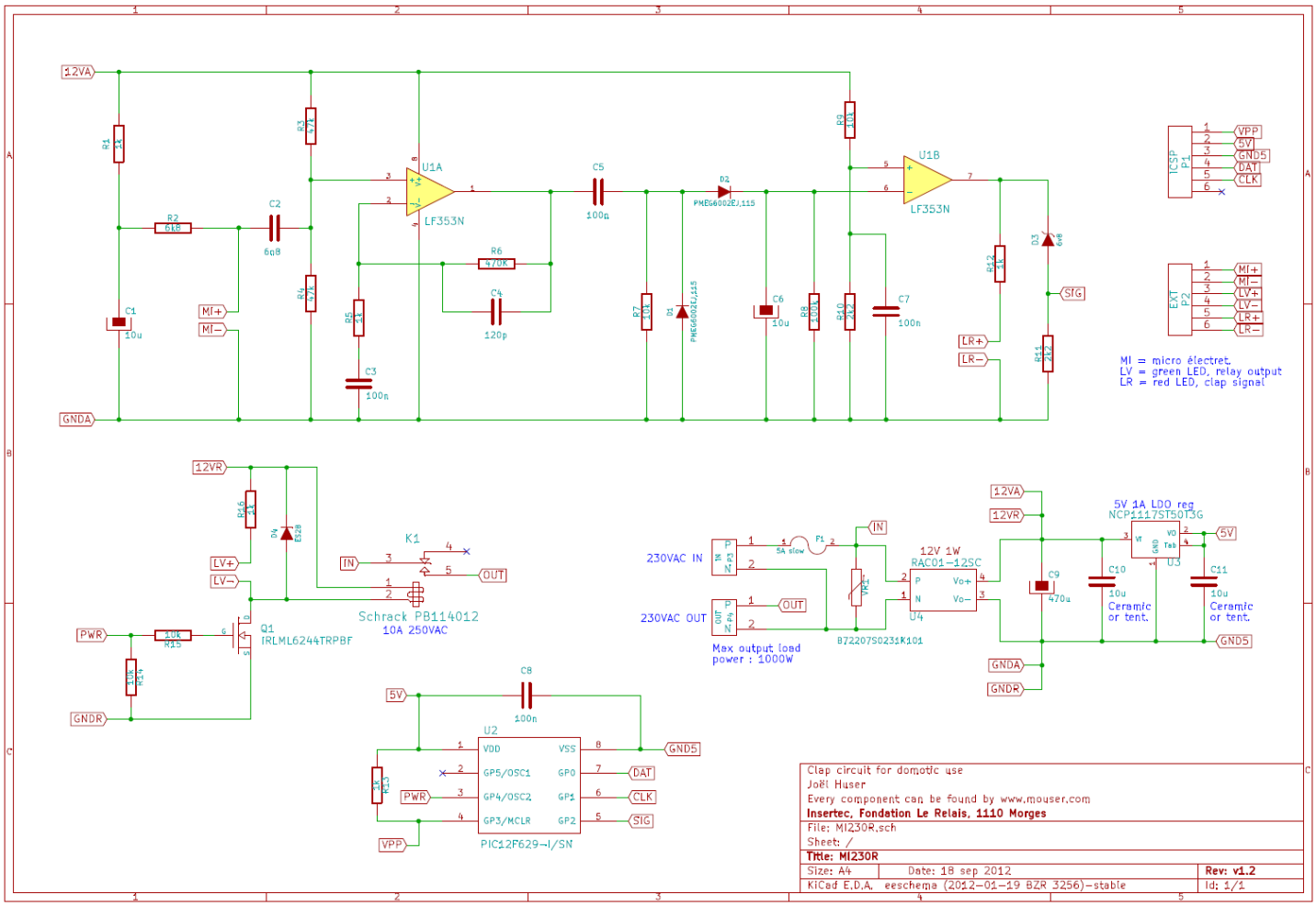


Construction d’un crossover 24 dB/Octave

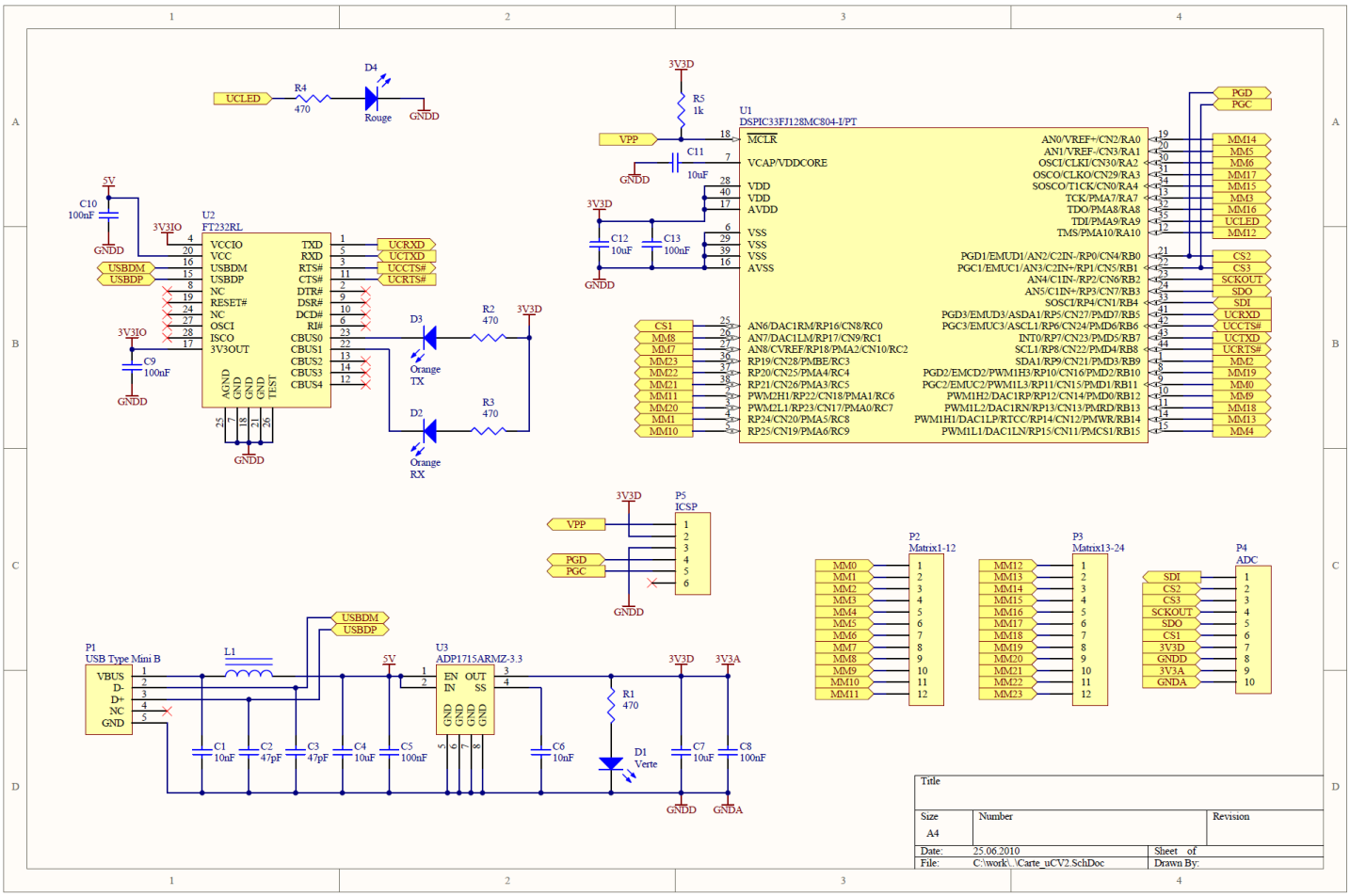
Réparation d’une table tournante pour l’horlogerie



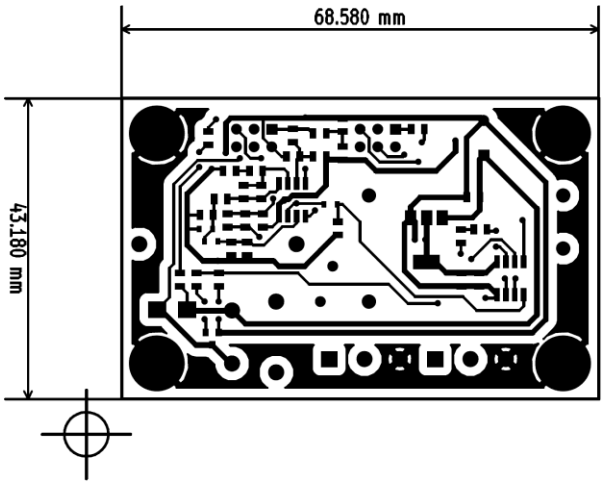
Développement du « Clap circuit » avec Kikad



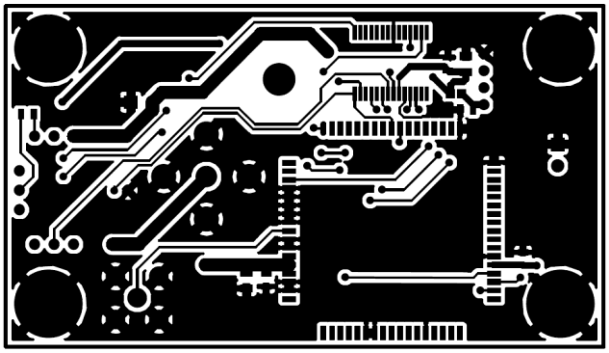
Interface USB pour matrice analogique avec Altium



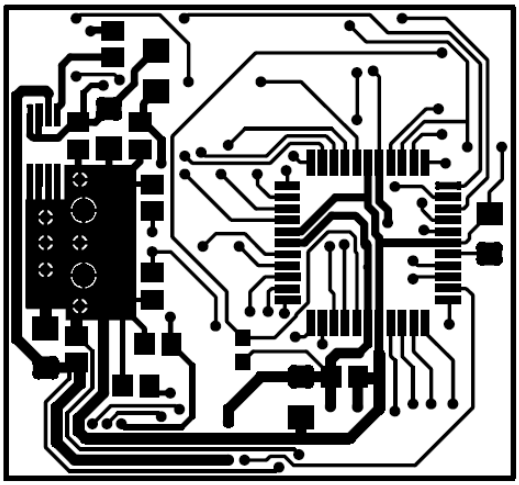
GBL du « Clap circuit »



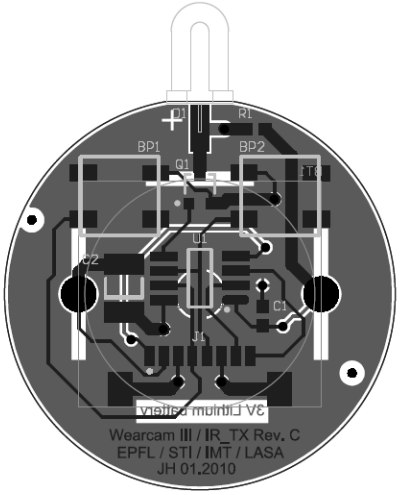
GTL de « Sentinel »



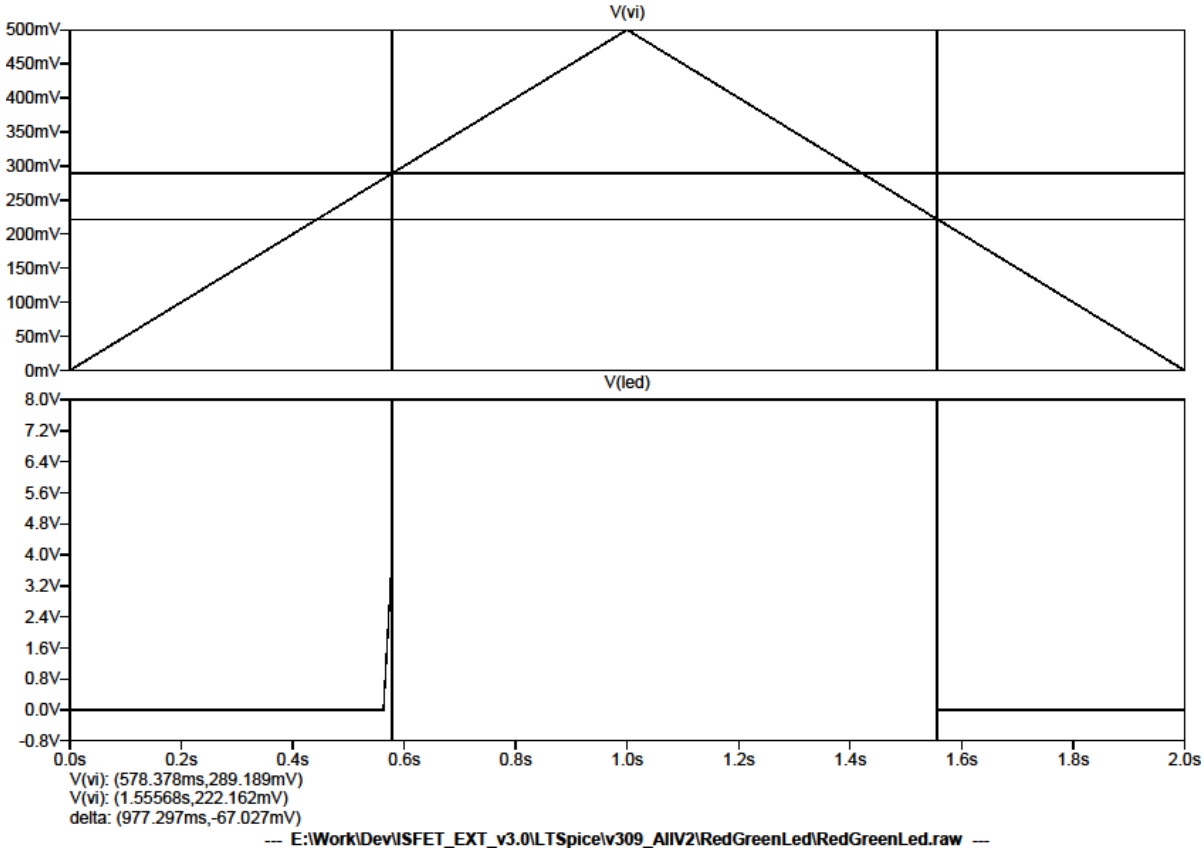
GTL de l'interface matricielle



Vue globale de la télécommande IR

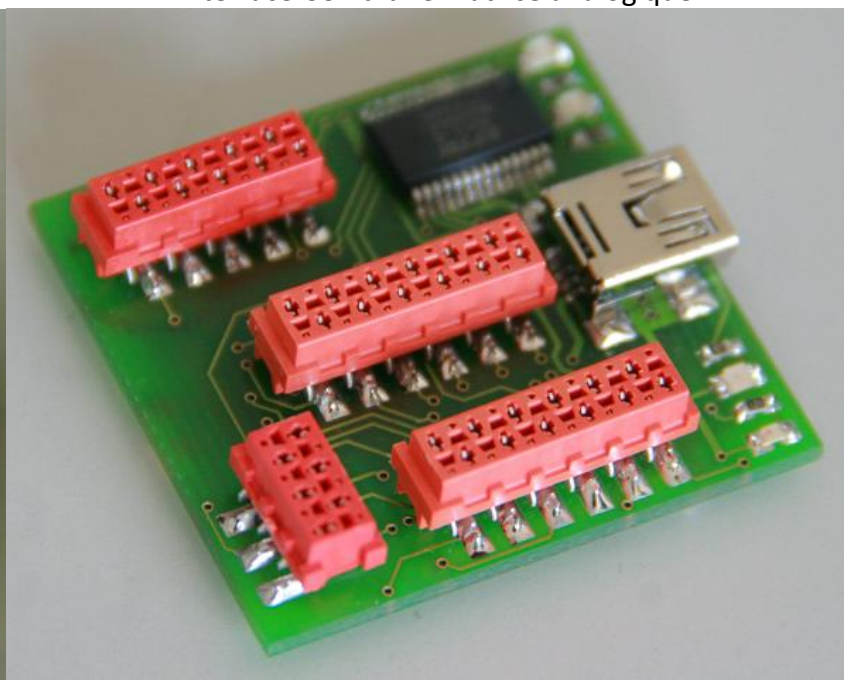


Simulations avec LTSpice IV

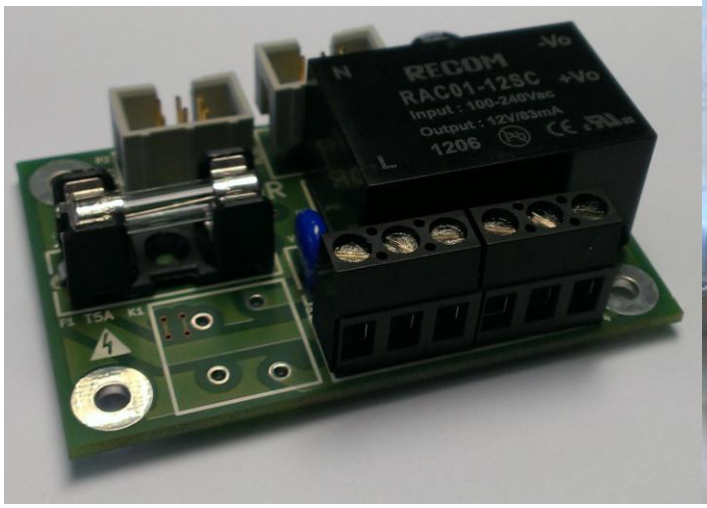


Télécommande IR

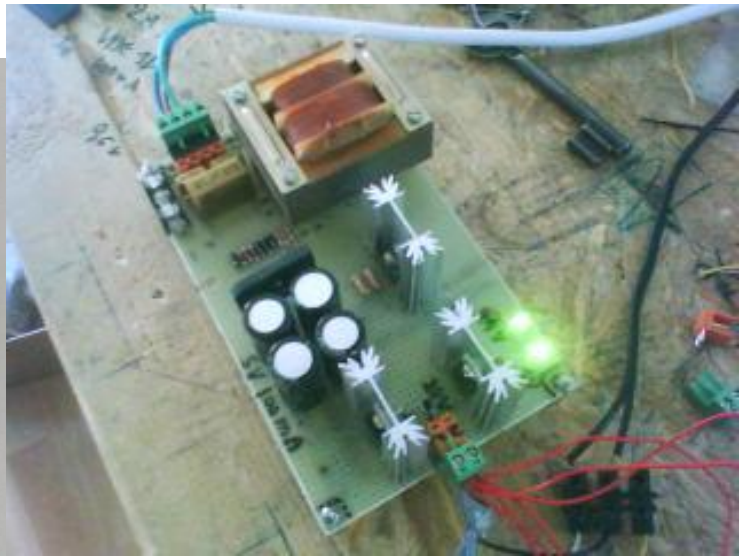
Interface USB d'une matrice analogique



« Clap-circuit »



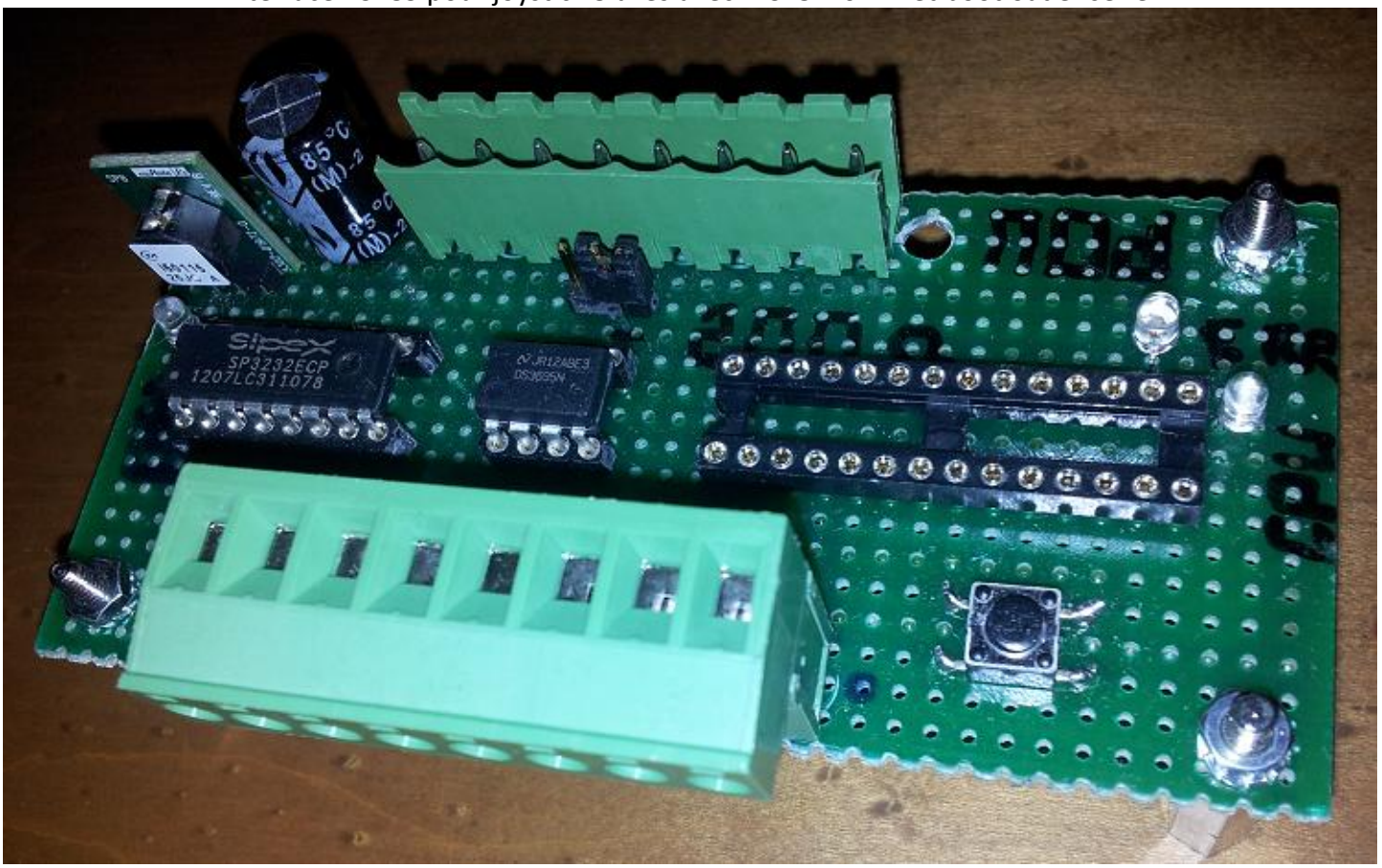
Alimentation +15V et +5V



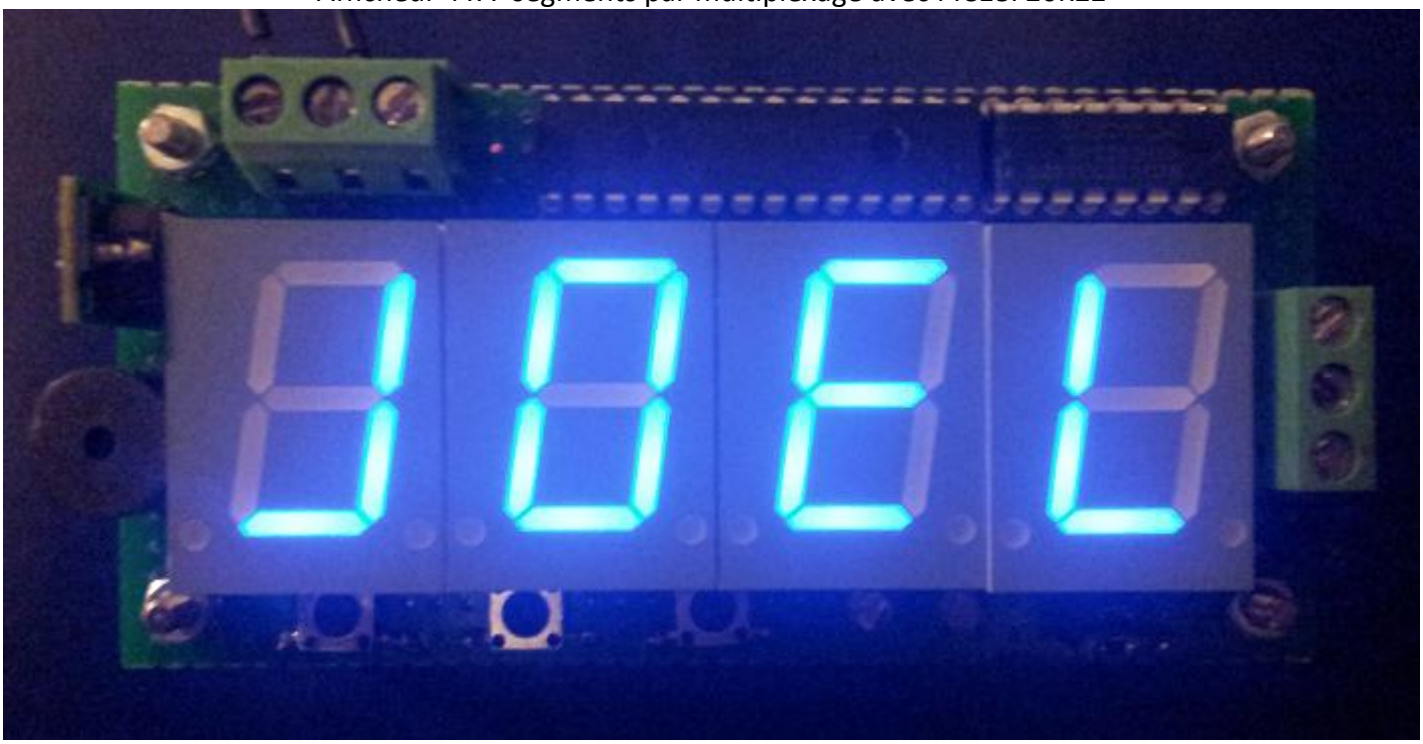
Interface Ethernet pour 2 lampes halogènes 12V

Driver PWM pour LED tricolore

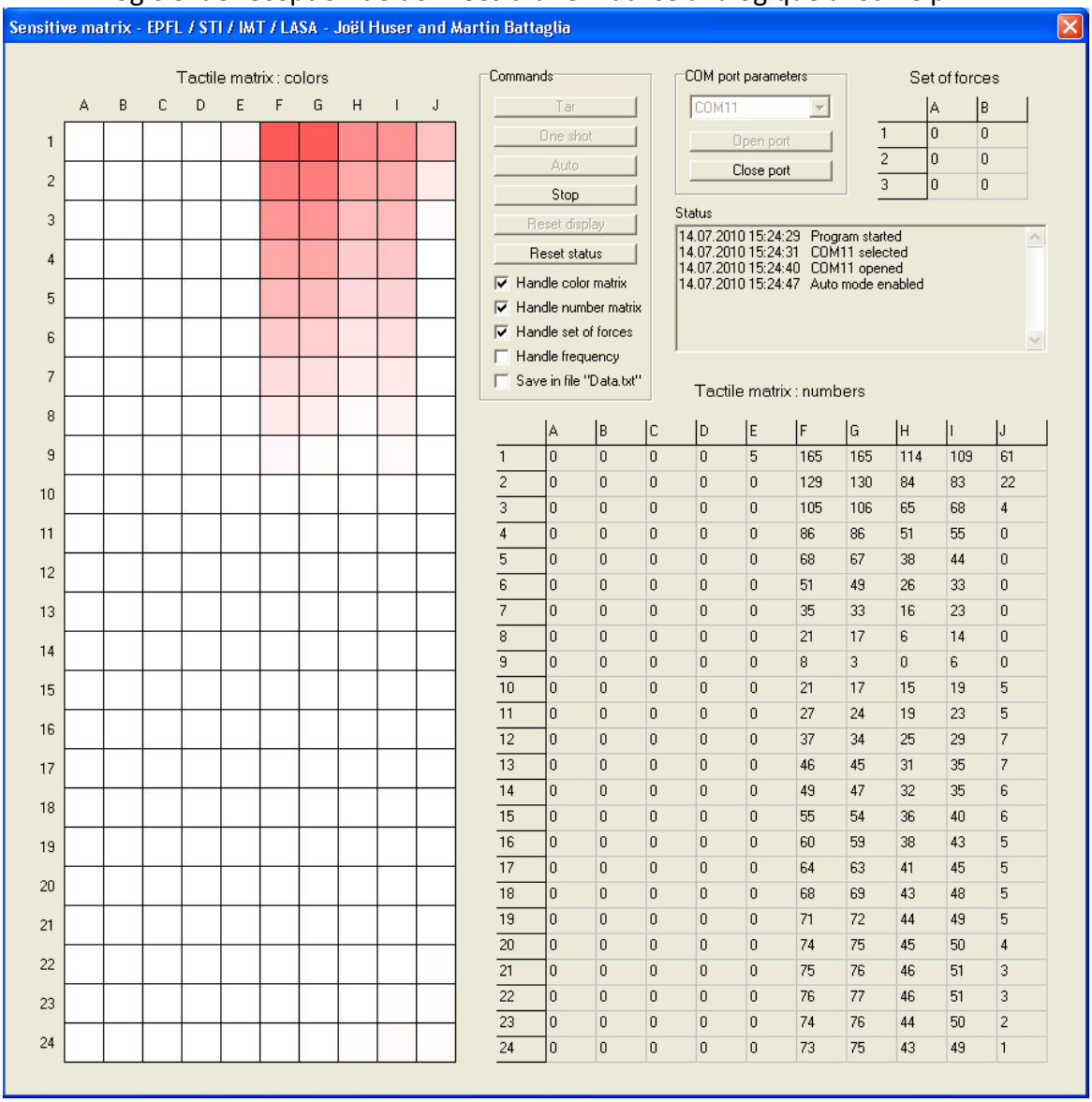
Interface RS485 pour joystick 3 axes avec PIC18F26K22 et bootloader seriel



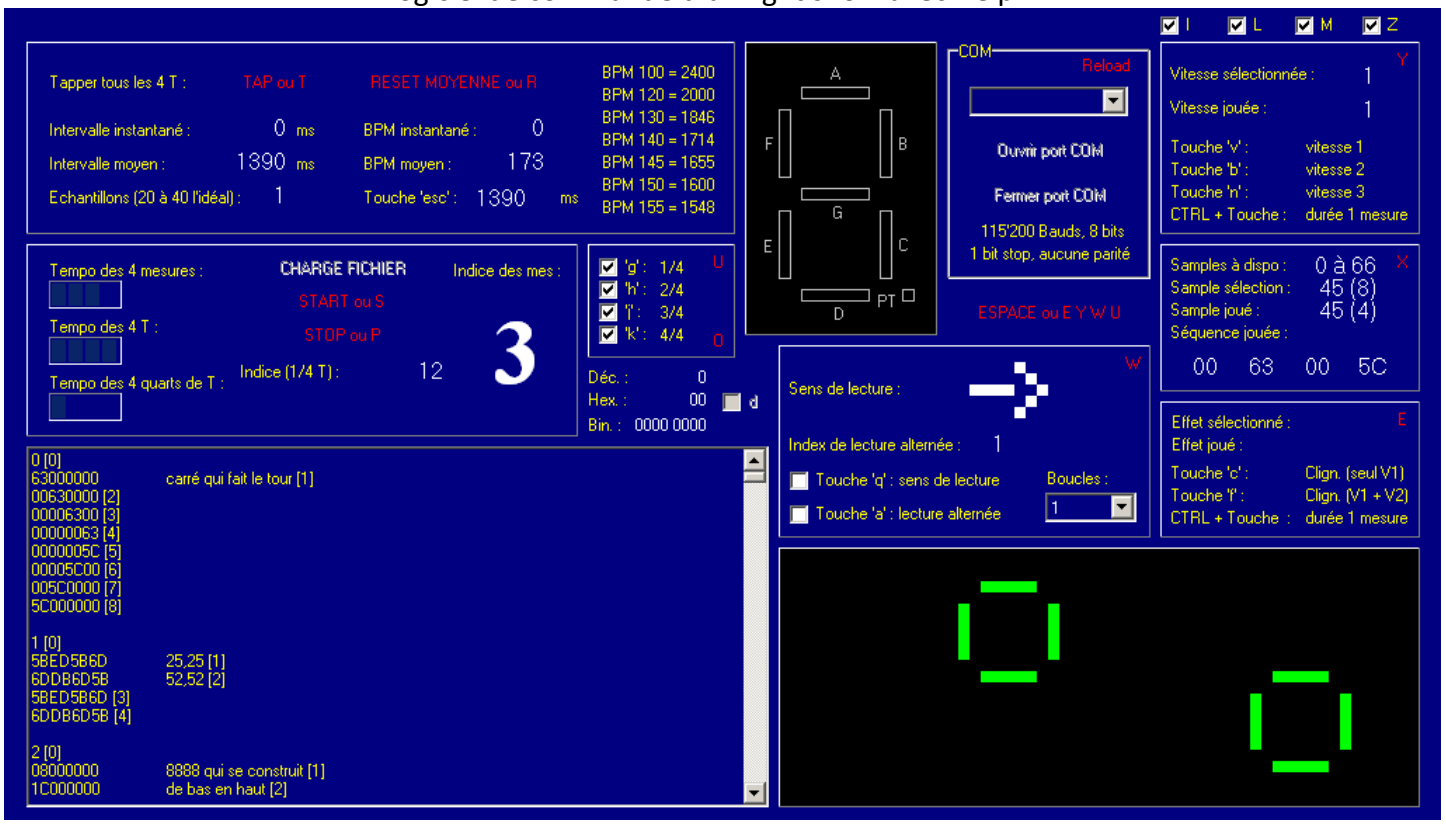
Afficheur 4 x 7 segments par multiplexage avec PIC18F26K22



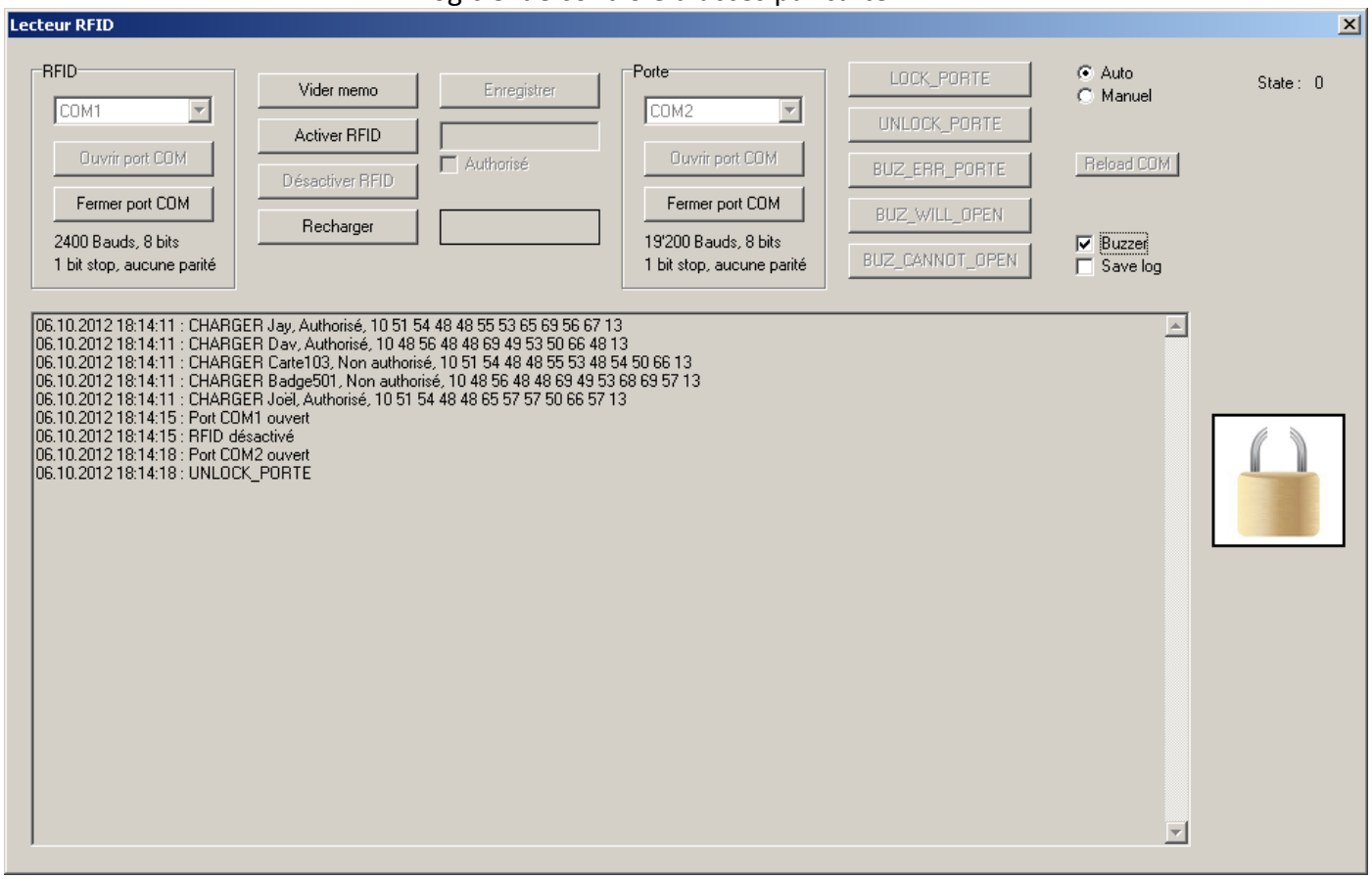
Logiciel de réception de données d'une matrice analogique avec Delphi 7



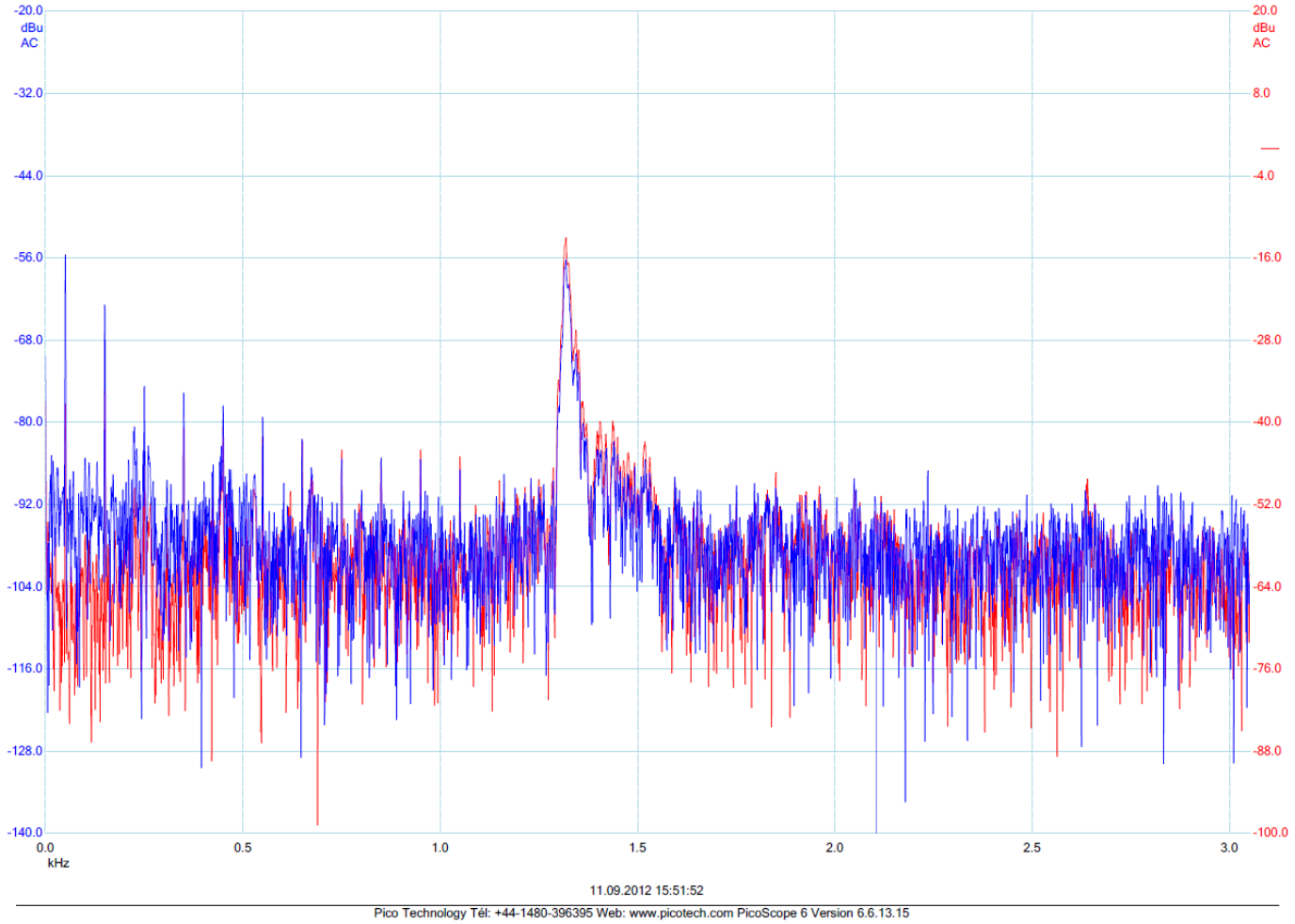
Logiciel de commande d'un light show avec Delphi 7



Logiciel de contrôle d'accès par carte RFID



Transformée de fourrier avec un picoscope 2205



Exemple de code C pour un PIC 18F26K22 avec MPLab

```
373:                                        // it's for me
374:                if(RxUart2 == SlaveRS485Adress) {
375:                    ItsForMe = 1;
376:                    AddRxByte(RxUart2);
377:                }
378:            }
379:            else {                      // we are not @ the start of a frame
380:                if(ItsForMe > 0) {
381:                    if((AddRxByte(RxUart2)) > 0) {
382:                                        // Buffer full => handle buffer
383:                        j = ModbusCRC16(&RxBuffer[0], NRxBytes - 2);
384:                                        // compute CRC
385:                        if((GetLow8(j) == RxBuffer[NRxBytes - 2]) && (GetHigh8(j) == Rx
386:                                        // check CRC ok
387:                            switch(RxBuffer[1]) {
388:                                        // function
389:                                case 8 :    // ping => send back the same data
390:                                            // wait before send back data
391:                                            Delay10KTCYx(Delay10K);
392:                                            SendUart2(&RxBuffer[0], NRxBytes);
393:                                            break;
394:                                case 5 :    // reset => send back the same data and res
395:                                            // wait before send back data
396:                                            Delay10KTCYx(Delay10K);
397:                                            SendUart2(&RxBuffer[0], NRxBytes);
398:                                            Reset();
399:                                            break;
400:                                case 3 :    // want the data
401:                                            TxBuffer[0] = SlaveRS485Adress;
402:                                            TxBuffer[1] = 3;            // function
403:                                            TxBuffer[2] = 6;            // number of by
404:
405:                                            ADCON0bits.CHS = 0;         // channel 0
406:                                            ADCON0bits.GO_DONE = 1;     // start an AD
407:                                            while(ADCON0bits.GO_DONE);  // wait convers.
408:                                            TxBuffer[3] = ADRESH;       // ADC result H
409:                                            TxBuffer[4] = ADRESL;       // ADC result L
410:
411:                                            ADCON0bits.CHS = 1;         // channel 1
412:                                            ADCON0bits.GO_DONE = 1;     // start an AD
413:                                            while(ADCON0bits.GO_DONE);  // wait convers.
414:                                            TxBuffer[5] = ADRESH;       // ADC result H
415:                                            TxBuffer[6] = ADRESL;       // ADC result L
416:
417:                                            ADCON0bits.CHS = 2;         // channel 2
418:                                            ADCON0bits.GO_DONE = 1;     // start an AD
419:                                            while(ADCON0bits.GO_DONE);  // wait convers.
420:                                            TxBuffer[7] = ADRESH;       // ADC result H
421:                                            TxBuffer[8] = ADRESL;       // ADC result L
422:
423:                                            j = ModbusCRC16(&TxBuffer[0], NTxBytes - 2)
424:                                            // compute CRC
425:                                            TxBuffer[NTxBytes - 2] = GetLow8(j);
426:                                            TxBuffer[NTxBytes - 1] = GetHigh8(j);
427:
428:                                            // wait before send back data
429:                                            Delay10KTCYx(Delay10K);
430:                                            SendUart2(&TxBuffer[0], NTxBytes);
431:                                            break;
432:                            }
433:                        }
434:                        else {              // CRC not ok
```