Bartosz Guździoł I2 106144

Hubert Mizera I2 106080

Wyświetlanie danych, pobranych ze strony filmweb.pl na temat wybranego filmu na ekranie wyświetlacza mikroprocesora.

Cele projektu

Celem projektu jest wyświetlanie danych na temat filmu, którego nazwę wpisuje użytkownik w konsoli na komputerze. Następnie dane zostają parsowane ze strony internetowej filmweb.pl (jest to internetowa baza danych na temat filmów itp.). Później, odpowiednio obrobiony strumień danych jest wysyłany przy pomocy USB na mikroprocesor. Mikroprocesor STM32F4 Discovery jest odpowiedzialny za inicjalizację wyświetlacza oraz kanału komunikacyjnego (USB). Mikroprocesor zajmuje się także obrobieniem przesłanych do niego danych z komputera oraz odpowiednie wysłanie ich na wyświetlacz, by zostały wyświetlone. Program na komputerze umożliwia wielokrotne wyszukiwanie danych na temat kolejnych filmów lub zamknięcie programu.

Implementacja kodu

Pobieranie danych z Internetu

```
if(curl)
	{
	curl_easy_setopt(curl, CURLOPT_URL, URL);
	curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteCallback);
	curl_easy_setopt(curl, CURLOPT_WRITEDATA, &readBuffer);
	res = curl_easy_perform(curl);
	/* Check for errors */
	if(res != CURLE_OK)
		fprintf(stderr, "curl_easy_perform() failed: %s\n",
				curl_easy_strerror(res));
	/* always cleanup */
	curl_easy_cleanup(curl);

/.../
curl = curl_easy_init();
	if(curl)
	{
	curl_easy_setopt(curl, CURLOPT_URL, URL2);
	curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteCallback);
	curl_easy_setopt(curl, CURLOPT_WRITEDATA, &readBuffer);
	res = curl_easy_perform(curl);
	/* Check for errors */
	if(res != CURLE_OK)
		fprintf(stderr, "curl_easy_perform() failed: %s\n",
				curl_easy_strerror(res));
	/* always cleanup */
	curl_easy_cleanup(curl);
```

Kontrola wątków

```
HANDLE  watek;
	 while(1)
	 {
			std::cout<<"Wpisz \"exit\" aby zakonczyc lub"<<std::endl<<"Podaj nazwe
filmu: ";
			getline(std::cin, nazwa);
			std::cout<<std::endl;
			if(watek!=NULL)
			{
				stopBool=1;
```

```
                        WaitForMultipleObjects(1, &watek, TRUE, INFINITE);
                        CloseHandle(watek);
                    }
                    if(!nazwa.compare("exit"))
                        break;
                    nazwa = std::tr1::regex_replace(nazwa, std::tr1::regex("[ ]+"),
std::string("+"));
                    std::vector<std::string> *NaPlytke = new
std::vector<std::string>(Szukaj(nazwa));
                    stopBool=0;
                    watek =
CreateThread( NULL,0,MyThreadFunction,reinterpret_cast<LPVOID>(NaPlytke),0,0);
                    if (watek == NULL)
                    {
                       ErrorHandler(TEXT("CreateThread"));
                       ExitProcess(3);
                    }
                    nazwa.erase();
         }
```

Wysyłanie danych do mikroprocesora

```
static void Main(String^ portName, std::vector<std::string> data)
    {
        String^ message;
        StringComparer^ stringComparer = StringComparer::OrdinalIgnoreCase;
        Thread^ readThread = gcnew Thread(gcnew ThreadStart(PortChat::Read));

        // Create a new SerialPort object with default settings.
        _serialPort = gcnew SerialPort();

        // Allow the user to set the appropriate properties.
        _serialPort->PortName = portName;
        _serialPort->BaudRate = SetPortBaudRate(_serialPort->BaudRate);
        _serialPort->Parity = SetPortParity(_serialPort->Parity);
        _serialPort->DataBits = SetPortDataBits(_serialPort->DataBits);
        _serialPort->StopBits = SetPortStopBits(_serialPort->StopBits);
        _serialPort->Handshake = SetPortHandshake(_serialPort->Handshake);

        // Set the read/write timeouts
        //_serialPort->ReadTimeout = 500;
        _serialPort->WriteTimeout = 500;

        _serialPort->Open();
        _continue = true;
        //readThread->Start();
              System::String^ startBit = gcnew String("{");
              System::String^ newLineBit = gcnew String("~");
              System::String^ stopBit = gcnew String("}");
              System::String^ clr = gcnew String("]");
              int licznik=0;
              while(!stopBool)
              {
                     std::vector<std::string>::iterator it;
                     _serialPort->Write(startBit);Sleep(15);
                     for(it=data.begin(); it!=data.end(); ++it)
                     {
                            if((*it).length()>21)
                            {
                                   int off = (*it).length() - 21;
                                   if(licznik<off)
                                   {
```

```
                                        System::String^ tmp = gcnew
String((*it).substr(licznik,7).c_str());
                                        System::String^ tmp2 = gcnew
String((*it).substr((licznik+7),7).c_str());
                                        System::String^ tmp3 = gcnew
String((*it).substr((licznik+14),7).c_str());
                                        _serialPort->Write(tmp);Sleep(15);
                                        _serialPort->Write(tmp2);Sleep(15);

                                        _serialPort->Write(tmp3);Sleep(15);
                                }
                                else
                                {
                                        System::String^ tmp = gcnew
String((*it).substr(licznik%off,7).c_str());
                                        System::String^ tmp2 = gcnew
String((*it).substr(((licznik%off)+7),7).c_str());
                                        System::String^ tmp3 = gcnew
String((*it).substr(((licznik%off)+14),7).c_str());
                                        _serialPort->Write(tmp);Sleep(15);
                                        _serialPort->Write(tmp2);Sleep(15);
                                        _serialPort->Write(tmp3);Sleep(15);
                                }
                                _serialPort->Write(newLineBit);Sleep(15);
                        }
                        else
                        {
                                if(licznik == 0)
                                {
                                        int off = 21-(*it).length();
                                        for(int i=0; i<off; i++)
                                        (*it).push_back(' ');
                                                System::String^ tmp = gcnew
String((*it).substr(0,7).c_str());
                                                System::String^ tmp2 = gcnew
String((*it).substr(7,7).c_str());
                                                System::String^ tmp3 = gcnew
String((*it).substr(14,7).c_str());
                                                _serialPort->Write(tmp);Sleep(15);
                                                _serialPort->Write(tmp2);Sleep(15);
                                                _serialPort->Write(tmp3);Sleep(15);

                                }
                                        _serialPort->Write(newLineBit);Sleep(15);
                        }
                }
                licznik++;
                Sleep(40);
                _serialPort->Write(stopBit);Sleep(15);
            }
            _serialPort->Write(clr);Sleep(15);
        //readThread->Join();
        _serialPort->Close();Sleep(15);
    }
```

Odbieranie danych w mikroprocesorze oraz wysyłanie danych na wyświetlacz

```
      int odbior = 0, czysc=0, linia=0, n=0;
      while (1){
            if(usb_cdc_kbhit())
            {
                  char c;
```

```
                c = usb_cdc_getc();
                switch(c){
                        case '{':
                                odbior = 1;
                                break;
                        case '~':
                                linia++;
                                n=0;
                                break;
                        case '}':
                                czysc = 1;
                                break;
                        case ']':
                                display_refreshAll();
                                break;
                        default:
                                if(odbior)
                                {
                                                ST7565_drawchar(n, linia, c);
                                                n+=6;
                                }
                                break;
                }
                if(czysc)
                {
                        odbior=0;
                        linia=0;
                        czysc=0;
                        n=0;
                        ST7565_display();
                }
        }
}
```