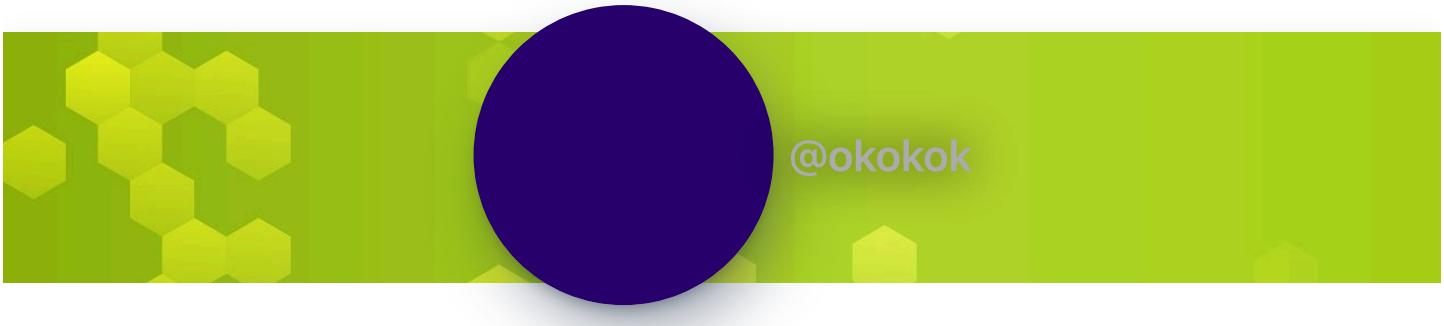
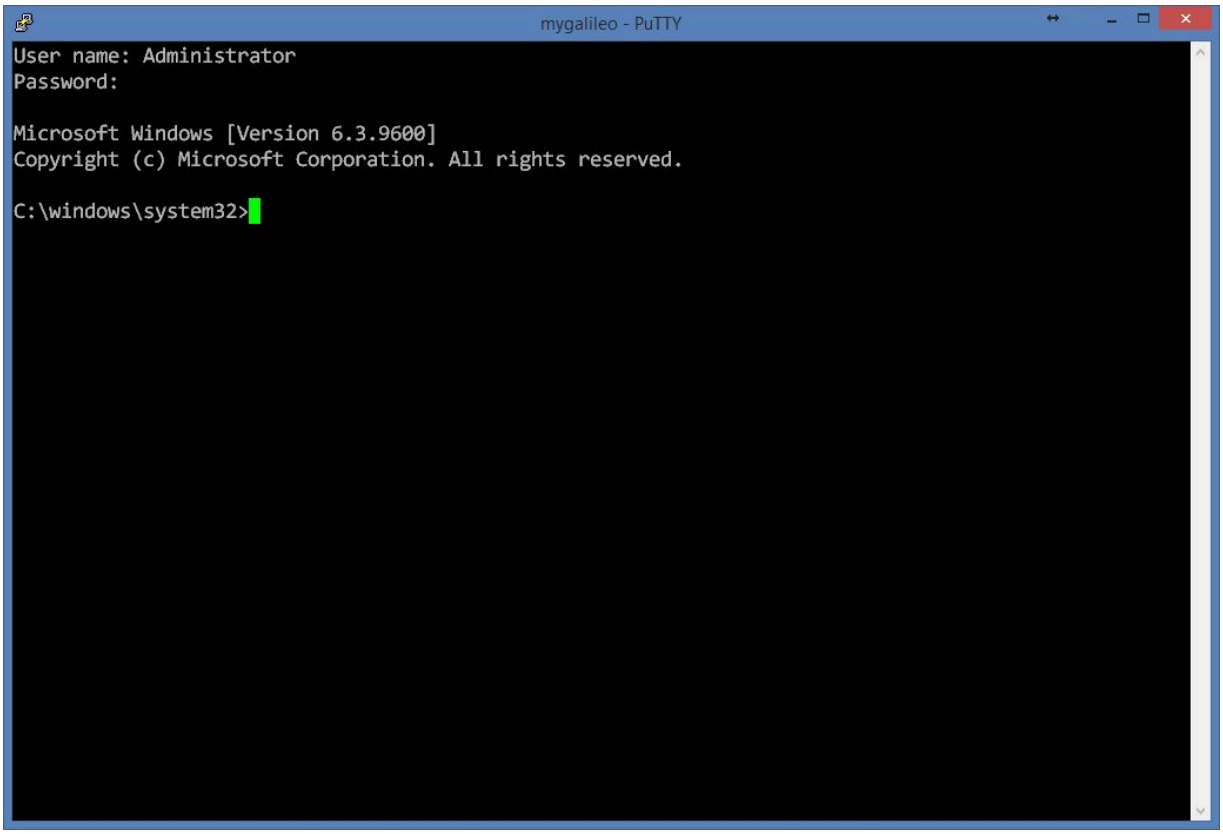


MinWin – Windows dla Internetu Rzeczy

18 sierpnia 2014, 03:49

UDOSTĘPNIJ

Cofnijmy się do czasów, w których Windows XP był najnowszym systemem Microsoftu, a sam Microsoft już od wielu lat pracował nad jego następcą, znanym wtedy pod nazwą kodową Windows Longhorn. Z siedziby producenta, w Redmond, wyciekały kolejne buildy systemu, które, dzięki znajomemu prowadzącemu sklep komputerowy, miałem okazję wtedy testować. Przez Longhorna przewinęło się mnóstwo ciekawych rozwiązań, które niestety nie doczekały się miejsca w finalnej edycji Windows Vista. Prace Microsoftu skupione były głównie nad interfejsem nowego systemu. Był to wciąż Iksepek. Może trochę upudrowany i świetnie wyglądający w nowej skórce - Slate, a wcześniej Plex, jednak wciąż Ikspek.



Działo się tak aż do momentu wydarzenia znanego później jako Longhorn Reset. Microsoft porzucił efekty dotychczasowych prac nad Longhornem i rozpoczął wszystko na nowo. Tym razem skupiając się na przerobieniu jądra systemu. To wtedy powstała szósta, aktualna do dziś, gałąź wydań rodziny systemów NT. Od tego czasu, w kolejnych buildach nie pojawiały się jednak żadne interesujące funkcje. Z zewnątrz wyglądało na to, że rozwój Windowsa stanął w miejscu.

Jednym z celów które postawiono sobie, w pracach nad nowym kernelem, był jego podział na warstwy. Microsoft bardzo zazdrościł Linuksowi możliwości budowy małej dystrybucji, którą można by uruchomić choćby na routerze, wrzucając tam jedynie to co jest w danej chwili potrzebne do pracy takiego urządzenia. W jądrze Windowsa wiele rzeczy polegało jedna na drugiej. Nie można było wtedy tak po prostu usunąć z niego interfejsu graficznego, bo jakiś mały, niskopoziomowy element mógł polegać na funkcji udostępnianej przez coś odpowiedzialnego głównie za rysowanie GUI.

Pierwszym etapem prac nad nowym jądrem - MinWin - było przypisanie do odpowiednich jego elementów numerów warstw, kolejnym sprawdzanie czy elementy danej warstwy nie polegają na elementach warstw wyższych i eliminowanie tego zjawiska. Dopiero wtedy stało się możliwe okrojenie Windowsa i przystosowanie go do małych urządzeń.

Dużo później oglądałem nagranie jednej z prezentacji Microsoftu, przedstawiającej efekty tych prac - system operacyjny zbudowany jedynie z najniższych warstw nowego jądra MinWin, składający się z około 100 plików i ważący niecałe 25 MB. Był na nim uruchomiony prosty serwer HTTP udostępniający trzy "podstrony" - listę procesów, listę tych 200 plików oraz statystyki zajętości pamięci.

Patrząc na konsolowe instalacje Windows Server - Server Core, wydawać by się mogło że ten prototypowy system nie doczekał się żadnej kontynuacji. Byłem tego praktycznie pewien. W końcu, na serwerze z założenia bez GUI instalowany był ciągle GUI, usunięto jedynie Explorer i parę graficznych aplikacji. Zdanie zmieniłem zaraz po rozpakowaniu Galileo :-)

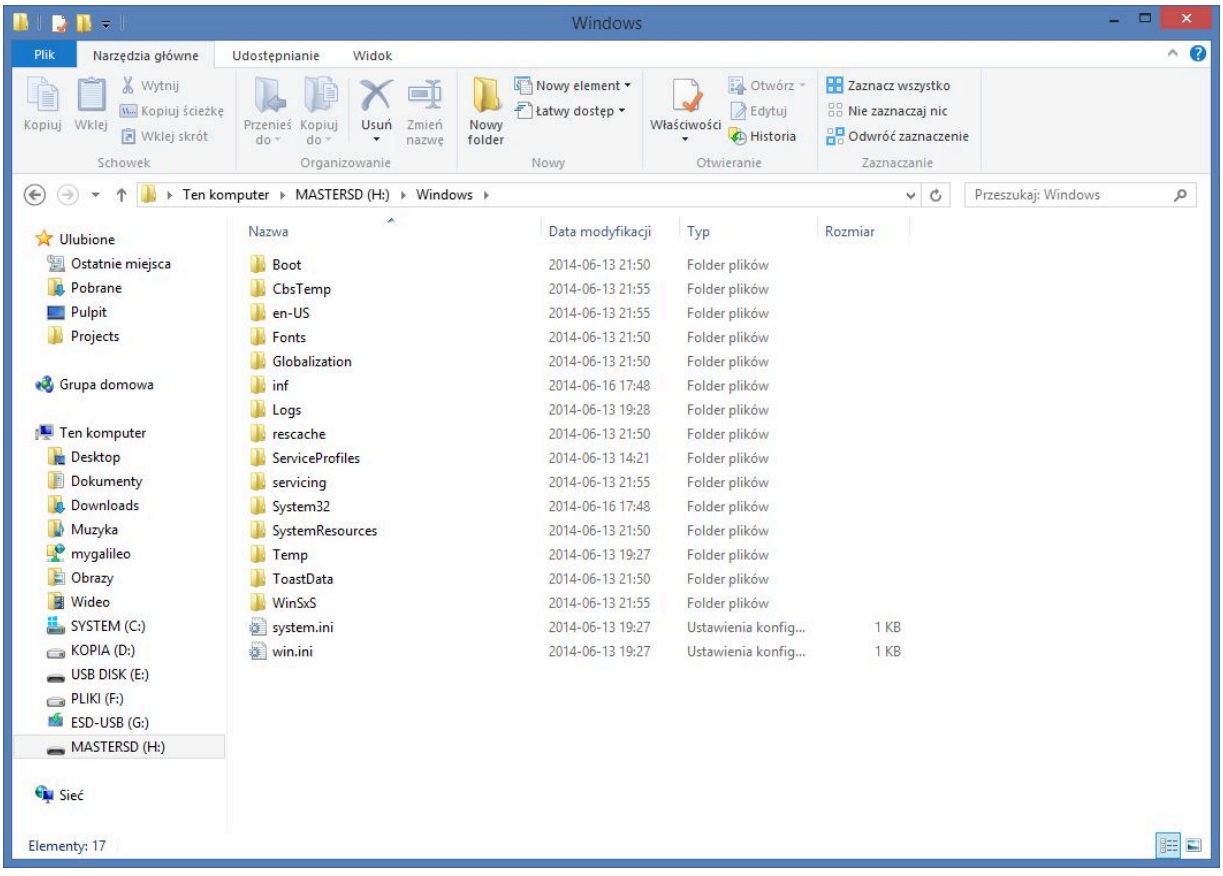
Windows dla IOT

Intel Galileo nie posiada wbudowanej karty graficznej. Dostęp przez RDP (zdalny pulpit) również nie jest możliwy. Jest to więc bardzo podobny system do tego przedstawionego na prezentacji z przed lat. Nie usunięto tutaj po prostu Explorera, jak ma to miejsce w wydaniach serwerowych - to pierwsze wydanie Windowsa bez jakiegokolwiek trybu graficznego!

Windows, nawet bez okienek, to jednak dalej Windows. Mimo ich braku, mamy tutaj dostęp do pełnych środowisk Win32 oraz .NET. Możemy skopiować na Galileo niemal każdego 32-bitowego konsolowego exe-ka i bez problemu uruchomić. Ciężkim zadaniem może jednak okazać się znalezienie takiego programu. Interfejs graficzny przez lata stał się nieodzownym elementem Windowsa. Dziś nawet oprogramowanie serwerowe, przeznaczone dla tej platformy, posiada okienkowe, graficzne interfejsy umożliwiające start wybranych usług lub podgląd logów.

Windows dla IOT legitymuje się wersją NT 6.3.9600, a więc jest to aktualne wydanie 8.1. Znajdziemy tutaj także najnowszą wersję środowiska .NET Framework - 4.5.

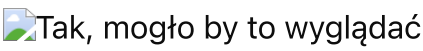
Pliki wykonywalne znajdujące się w folderze System32 to głównie polecenie konsolowe. Jest ich tylko 83. Windows bez plików pagefile.sys oraz swapfile.sys zajmuje tylko trochę ponad 700 MB. Microsoft praktycznie nie pozostawił niczego niepotrzebnego! Pamiętajmy że jeszcze w Windows XP SP2 znajdował się Program Manager - poprzedniki Explorera, ostatni raz domyślnie wykorzystywany w Windows for Workgroups 3.11 z 1993 roku.



Skoro nie ma okienek to jak się tam dostać?

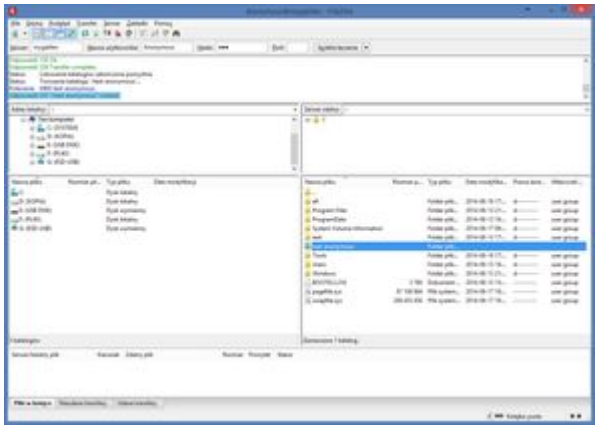
Microsoft przewidział aż cztery metody dostępu, z pośród których najważniejszą jest Telnet... Tak, ten nieszyfrowany i obecnie nie wykorzystywany już praktycznie protokół.

Poprzez Telnet, po zalogowaniu jako Administrator z hasłem admin, uzyskujemy dostęp do dobrze znanego cmd.exe. Aż dziwne że Gigant z Redmond (tak musiałem użyć tego określenia :P) nie udostępniła tutaj zdalnego PowerShella, a zdecydował się właśnie na cmd.exe oraz Telnet. Brzmi to prawie tak śmiesznie jak wysyłanie exploitów, emacsem, przez sendmaila. Ma jednak jedną zaletę - jest to bardzo lekkie rozwiązanie. Wystarczające do uruchomienia exe-ka pełniącego jakąś ściśle określoną funkcję np. sterowania elektroniką. Brakuje niestety szyfrowania :-(

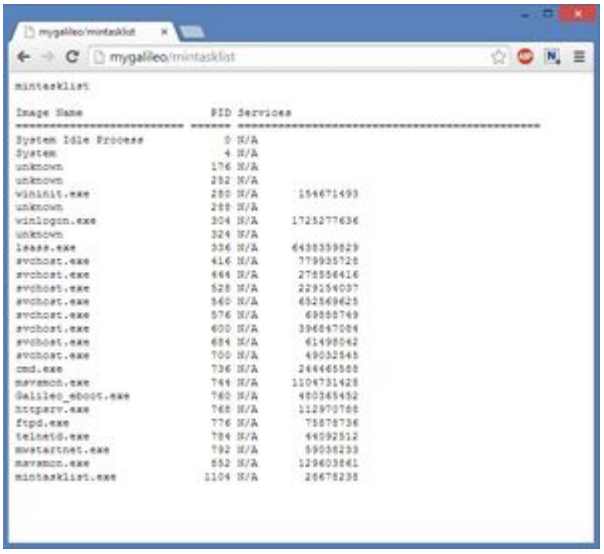

Tak, mogło by to wyglądać

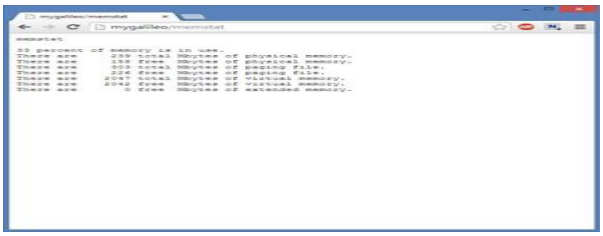
Tak, mogło by to wyglądać

Kolejną metodą dostępu jest FTP. I działa on tak, że aż nie mogę w to uwierzyć... Wbudowany serwer wpuszcza wszystkich użytkowników z dowolnymi loginami i hasłami. Przyznaje im też wszelkie prawa do modyfikacji zawartości systemu plików! Nie wspiera niektórych typowych poleceń, jak na przykład, zmiana nazwy pliku, a podczas kopiowania dużej ilości plików, serwer potrafi się wysypać i konieczny jest wtedy jego restart.


118496

Kolejną uruchomioną na Galileo usługą jest serwer HTTP udostępniający trzy podstawowe polecenia - listę procesów, niedziałającą listę plików oraz zajętość pamięci. To właśnie ta sama strona o której wspominałem na początku wpisu. I to właśnie dzięki niej przypomniałem sobie o MinWin i prezentacji z przed lat.

118497


118499 

Jak widać, bez żadnego firewalla np. w postaci Raspberry, wpiętego przed Galileo, nie możemy na razie wypuścić go do Internetu. Zdalny dostęp to porażka. Odnoszę wrażenie że Microsoft potraktował wbudowaną kartę sieciową jak port komunikacyjny z komputerem, a nie interfejs sieciowy. Serwery FTP i HTTP zostały wprost przeniesione z dawnego buildu MinWin. Nie poprawiono w nich żadnych błędów. Ba, nie wygląda nawet na to żeby ktokolwiek przetestował działanie tych narzędzi z nowym Windowsem. Funkcja listująca wszystkie pliki działała z ich małą ilością na pokazie. Tutaj nie ma na to szans... Pamiętajmy jednak że to build testowy. Przed Microsoftem pozostało jeszcze sporo pracy, aby uczynić go użytecznym nie tylko na biurku z elektroniką. Zgodnie z EULĄ, w ciągu kilku miesięcy możemy spodziewać się kolejnego wydania.

Pierwszy projekt

Przejdźmy teraz do tego, co działa prawidłowo :-)

Jak pewnie zauważyliście, wraz z płytką, w paczce, znajdowała się również karta sieciowa Fast Ethernet na USB oraz patchcord UTP. Galileo posiada już kartę sieciową i Windows nie ma problemów z jej obsługą. Adapter USB -> Ethernet dołączony został, aby przy jego pomocy podłączyć płytkę do komputera. Windows nie obsługuje wbudowanego portu micro USB client.

Zaraz po podłączeniu karty do komputera i wykryciu jej przez system, Windows, dzięki domyślnie włączonej opcji autokonfiguracji i braku serwera DHCP w sieci, losuje sobie adres IP z puli APIPA 169.254.0.0/16. Identyczny proces zachodzi w systemie uruchomionym na Galileo. Po chwili więc możliwa jest już pełna komunikacja, bez jakiekolwiek konfiguracji. Domyślny hostname Galileo to mygalileo i wpisując tę nazwę, zamiast adresu IP, możemy się z nim połączyć. Przyznam że to bardzo sprytne rozwiązanie.

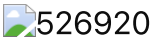
W przedstawionej przez Microsoft instrukcji, następnym krokiem po podłączeniu urządzenia do komputera, jest instalacja Visual Studio 2012 lub 2013, a dalej paczki *.msi zawierającej przykładowy projekt języka Visual C++ z dodatkowymi bibliotekami do obsługi portów GPIO (min. arduino.h) oraz domyślnie skonfigurowanym zdalnym wdrażaniem i debugowaniem przy pomocy Windows Remote Debuggera uruchomionego na Galileo. Funkcje dostarczane przez arduino.h są identyczne jak te dostępne na platformie Arduino. Język jest również ten sam. Czyni to Galileo z Windowsem w stu procentach kompatybilnym z Arduino. Uruchomiłem dzięki temu kilka przykładów ze strony http://arduino.cc/en/Tutorial/HomePage.

526917

Platforma .NET

Przykładowy projekt w C++ to jedyne co znajduje się w paczce *.msi. Nie znajdziemy tam żadnych innych projektów, ani choćby bibliotek, przygotowanych z myślą o pozostałych, wspieranych przez Visual Studio, językach programowania. Brakuje mi tutaj zwłaszcza out-of-the-boxowej obsługi GPIO w środowisku .NET. Wydawać by się mogło że Windows dla

Galileo został stworzony właśnie do tego celu. Z C++ możemy, w końcu, korzystać na samym Arduino. Można je jednak kupić za ułamek ceny płytki Galileo, albo korzystając z chipów ATMEGA, wykonać samemu. C++, a także inne języki dostępne są też na Linuksowej wersji płytki. Moim zdaniem to duży błąd Microsoftu.



Nic, na szczęście, nie jest stracone :-) Platforma .NET pozwala na import funkcji z bibliotek *.dll. Przerobiłem więc przykładowy projekt na taką bibliotekę i wyeksportowałem z niego kilka funkcji. Dalej zaimportowałem je, korzystając z DllImportera, i dodałem funkcję startującą projekt - DuinoRun(). Udostępniłem w ten sposób tylko część najważniejszych funkcji:

- PinMode() służącą do ustawienia pinu w tryb wejściowy lub wyjściowy,

- DigitalRead() umożliwiającą odczyt napięcia (LOW/HIGH) z pinów cyfrowych,
- DigitalWrite() pozwalającą ustawić napięcie 0V lub 5V na pinach cyfrowych,
- AnalogRead() odczytującą napięcie z pinów analogowych (A0-A5),
- oraz AnalogWrite() umożliwiającą skorzystanie ze sprzętowej funkcji PWM o której pisałem w poprzedniej części.

Import pozostałych funkcji jest równie prosty co powyższych i można go dokonać w identyczny sposób.

A oto Microsoftowy przykład "Hello Blinky!" przepisany na C#:

```
# C#

static void Main(string[] args)
{
        DuinoRun();
}

static void Setup()
{
        PinMode(13, OUTPUT);
}

static void Loop()
{
        DigitalWrite(13, HIGH);
        Thread.Sleep(200);

        DigitalWrite(13, LOW);
        Thread.Sleep(200);
}
```

***

Ja na razie zabieram się za naukę elektroniki, a sporo mam do nadrobienia. Zamówiłem już w Chinach przelotkę mirco USB -> USB potrzebną do mojego projektu. Postaram się też jakoś podmienić wbudowane usługi pochodzące z dawnego buildu na coś bezpieczniejszego i wypuścić Galileo w świat.
Co wy na małe hackme?

Netografia:

- http://pl.wikipedia.org/wiki/Microsoft_Longhorn - dość ciekawy opis kolejnych buildów Longhorna aż do resetu prac
- http://en.wikipedia.org/wiki/MinWin - anglojęzyczny opis prac nad jądrem MinWin

Sprzęt | Windows | Programowanie

Wybrane dla Ciebie