Tworzenie oprogramowania dla ARM'ów pod Linuxem

Darmowe, zintegrowane środowisko programistyczne dla mikrokontrolerów ARM

Na internetowym forum EP, w dziale poświęconym dyskusjom i komentarzom na temat magazynu EP, jakiś czas temu pojawił się wątek w którym czytelnicy zasugerowali, aby w jednym z wydań czasopisma pojawił się artykuł o programowaniu mikrokontrolerów pod Linuksem. Ponieważ to forum zostało stworzone, aby utworzyć pewnego rodzaju sprzężenie zwrotne celem lepszego, ciągłego dostosowywania EP do potrzeb czytelników, artykuł ten jest odpowiedzią na prośbę forumowiczów i przedstawia w jaki sposób zainstalować i skonfigurować zintegrowane środowisko programistyczne dla mikrokontrolerów ARM pod systemem Linux.

W numerach 2-3/2011 EP opublikowano artykuły przedstawiające instalację tytułowego środowiska dla systemu operacyjnego Windows. Jednak z pewnych względów nie wszystkie dane oraz linki tam zawarte są nadal aktualne. Dodatkowo instalacja nowych aplikacji i środowisk bywa często kłopotliwa zarówno dla nowych jak i obytych z systemem Linux osób. Jednym z bodźców do napisania tego artykułu była chęć minimalizacji czasu jaki elektronik będzie musiał poświęcić na instalację opisanego pakietu.

Który Linux?

Dystrybucji Linuksa jest wiele. Nie można obiektywnie stwierdzić, która jest najlepsza – każda ma swoje wady i zalety. Artykuł ten opisuje instalację i konfigurację zintegrowanego środowiska do tworzenia oprogramowania dla mikrokontrolerów ARM oraz jego wczytywania i debugowania w *Ubuntu 11.10 (64-bit)*. Jednak, aby nie ograniczać się tylko do jednej dystrybucji zostaną również podane komendy dzięki którym instalacja poszczególnych pakietów będzie możliwa poprzez konsolę. Dzięki temu artykuł może być bardziej pomocny osobom korzystającym z innych dystrybucji Linuxa.

Eclipse

Sprawdzonym i bardzo dobrze spisującym się pod Linuksem środowiskiem programistycznym jest *Eclipse*, który ma możliwość integracji z innymi aplikacjami. Pobieramy go wchodząc na stronę internetową www.eclipse.org, następnie klikamy na zakładkę *Downloads* i wybieramy pozycję *„Eclipse IDE for C/C++ Developers (includes Incubating components)”* w wersji Linux 32- lub 64-bitowej (w chwili pisania artykułu jest to wersja *INDIGO* (3.7.0) „ważąca” 107 MB). Następnie wybieramy serwer, z którego ma być pobrany plik i zapisujemy go na dysku (domyślnie w folderze *~/Downloads*). Należy pamiętać, że wszystkie pliki internetowe można pobrać również za pośrednictwem konsolowej przeglądarki internetowej, na przykład *Lynx*. Archiwum rozpakowujemy w trybie graficznym lub poprzez konsolę poleceniem (nazwa pliku może być inna w zależności od pobranej wersji):

```
tar zxvf eclipse-cpp-indigo-SR2-incubation-linux-gtk-x86_64.tar.gz
```

W trybie graficznym w katalogu domowym tworzymy folder *development* i przenosimy do niego folder *eclipse*. Używając konsoli wpisujemy:

```
mkdir ~/development
mv -f ~/Downloads/eclipse ~/development/
```

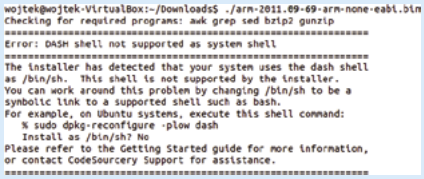
Typ sposobem *Eclipse* jest gotowy do pracy. Można go uruchamiać poprzez dwukrotne kliknięcie na ikonie eclipse lub w razie potrzeby z konsoli poprzez wpisanie komendy:

```
~/development/eclipse/eclipse
```

Używając konsoli, będąc już w folderze *eclipse* należy wpisać (na komputerze należy mieć zainstalowaną maszynę wirtualną *Javy*)

Sourcery CodeBench

W poprzednich artykułach opisujących zintegrowane środowisko programistyczne dla mikrokontrolerów ARM pod Windowsa, jako *toolchain* użyty został pakiet firmy CodeSourcery – Sourcery G++ Lite Edition. Jednak w ubiegłym roku firma Mentor Graphics przejęła oprogramowanie *CodeSourcery* i je rozwija pod inną nazwą: *Sourcery CodeBench*. Ponieważ nadal dostępna jest darmowa wersja tego *toolchain'a*, która dodatkowo tworzona jest w wersji linuksowej, została ona użyta w opisywanym środowisku dla mikrokontrolerów *ARM*. W celu pobrania go, otwieramy stronę internetową www.mentor.com, a następnie wybieramy *Products –> Embedded Software –> Sourcery CodeBench*. Na nowej stronie klikamy przycisk *Editions* i wybieramy darmową wersję oprogramowania „*Sourcery CodeBench Lite Edition”*. Z zestawienia znajdującego się na dole strony wybieramy „*Download the EABI Release”* dla procesorów *ARM*. Tworzymy własne konto lub jeżeli już je posiadamy – logujemy się w serwisie. W mailu otrzymamy link do oprogramowania. Na stronie klikamy na link „*Download CodeBench Lite 2011.09-69*” (lub nowszy, jeżeli będzie dostępny), a następnie „*IA32 GNU/Linux Installer*” i zapisujemy go na dysku (domyślnie w folderze *~/Downloads*). W chwili pisania artykułu jest to wersja *Sourcery G++ Lite 2011.09-69* ważąca 108 MB). Pobrany plik jest w postaci binarnej. Będąc w folderze *~/Downloads* można go uruchomić z poziomu konsoli poleceniem:



Rysunek 1. Zamiana shell'a

Rysunek 2. Listing konfiguracyjny shell'a

```
./arm-2011.09-69-arm-none-eabi.bin
```

W przypadku wystąpienia komunikatu:

```
bash:  ./arm-2011.09-69-arm-none-eabi.bin: Permission denied
```

należy nadać odpowiednie uprawnienia do uruchomienia pliku poprzez:

```
chmod a+x arm-2011.09-69-arm-none-eabi.bin
```

Podczas uruchamiania pliku binarnego może się okazać, że aktualnie używany *system shell* (w przypadku *Ubuntu* domyślnie ustawiony jest *DASH*) nie jest wspierany przez instalator (**rysunek 1**). Zmieniamy go poleceniem:

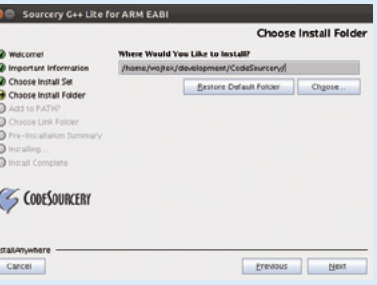
```
sudo dpkg-reconfigure -plow dash
```

a następnie wybieramy *No*. W konsoli powinniśmy otrzymać listing jak na **rysunku 2**. Po instalacji, jeżeli będziemy chcieli przywrócić domyślny *system shell*, należy po uruchomieniu polecenia rekonfiguracji wybrać opcję *Yes*.

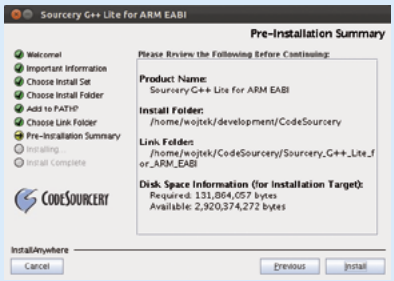
Po uruchomieniu pliku binarnego, otworzy się graficzny instalator. Klikamy *Next*, akceptujemy licencję, *Next,* znów *Next*, wybieramy opcję *Typical* i klikamy *Next.* Następnie należy wybrać ścieżkę instalacji. Jeżeli chcemy zainstalować to oprogramowanie dla różnych użytkowników systemu, to musimy wybrać odpowiedni folder i nadać plikom odpowiednie uprawnienia. Jednak na potrzeby tego kursu, pakiet zostanie zainstalowany w domowym folderze użytkownika – w moim przypadku nazwa użytkownika to *wojtek* (**rysunek 3**): */home/wojtek/development/CodeSourcery/* Klikamy na *Next*, następnie wybieramy opcję „*Modify PATH for current user*", *Next*, w opcji *Other* wpisujemy lokalizację */home/wojtek/development/CodeSourcery/links* i klikamy *Next*. Jeżeli wszystko się zgadza na wyświetlonym monicie (**rysunek 4**), możemy kliknąć na *Install*. Po zakończonej instalacji odznaczamy opcję „*View Getting Started guide?*", *Next* i *Done*. Istnieje również możliwość instalacji pakietu tylko z poziomu konsoli, bez okienkowego trybu graficznego. Taki rodzaj instalacji urachamiamy poprzez wpisanie w konsoli:

```
./arm-2011.09-69-arm-none-eabi.bin -i console
```

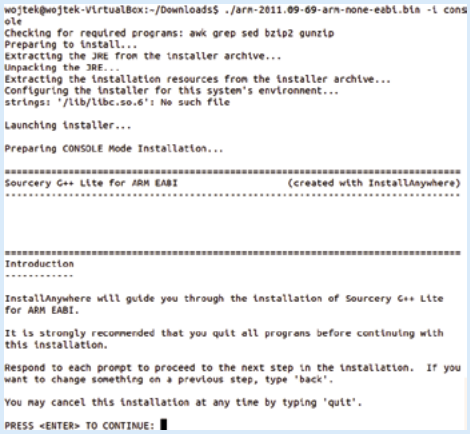
Początek instalacji konsolowej został przedstawiony na **rysunku 5**. Instalator poprowadzi nas przez te same kroki co w trybie graficznym. Różnica polega na tym, że wszystkie opcję musimy zatwierdzać z klawiatury (uwagę należy zwrócić, że aby zatwierdzić umowę licencyjną znakiem ‚*Y*', należy ją pierwsze całą przeglądnąć, wielokrotnie naciskając klawisz *Enter*). Instalacje przerwać można w dowolnym momencie wpisując komendę *quit* w konsoli. Po ukończonej instalacji jednym z wyżej opisanych sposobów, można rozpocząć integrację środowiska dla mikrokontrolerów *ARM*.



Rysunek 3. Graficzny wybór katalogu instalacyjnego



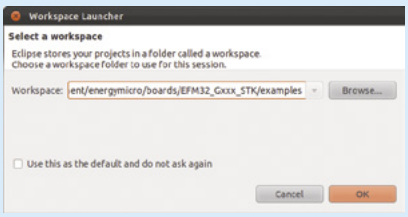
Rysunek 4. Graficzne potwierdzenie wyboru konfiguracji instalacji



Rysunek 5. Instalacja kompilatora z poziomu konsoli

Pierwszy projekt

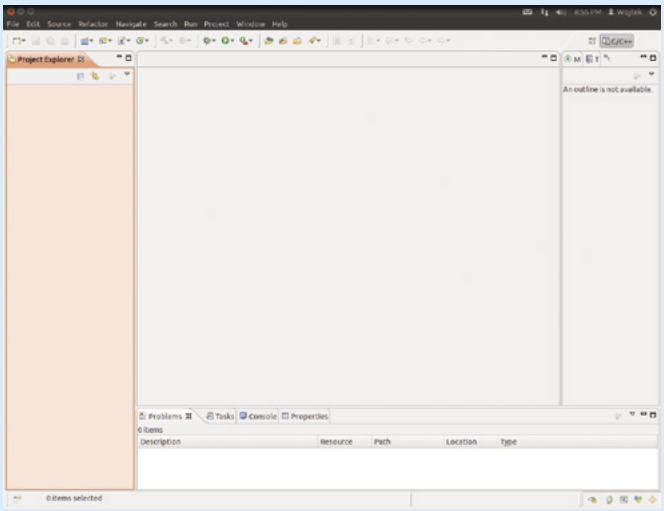
Jako kod do kompilacji posłuży przykładowy projekt testowy dla mikrokontrolerów *EFM32 Gecko* (*ARM Cortex-M3*) firmy *Energy Micro*. W celu zachowania hierarchii plików i katalogów, a zarazem ścieżek do potrzebnych bibliotek oraz ułatwienia, katalog *energymicro* (utworzony po instalacji aplikacji *Simplicity Studio* ze strony *www.energymicro.com*) wraz z zawartością zostanie skopiowany z windowsowej partycji do lokalizacji */home/wojtek/development/* – można tego dokonać w trybie graficznym lub poprzez konsolowe polecenie *cp*. Mając przykładowy, prosty projekt dla danego mikrokontrolera, dużo łatwiej jest zaczynać tworzyć oprogramowanie dla *ARM'ów*. Zatem zachęcam, aby na forum *EP* dzielić się projektami (szczególnie opartymi na pliku *makefile*) dla układów *ARM – ATMEL, NXP*, etc.



Rysunek 6. Wybór ścieżki z przykładowymi projektami (Eclipse)

Będąc w lokalizacji */home/wojtek/development/eclipse* uruchamiamy środowisko programistyczne poprzez plik *eclipse*. W przypadku pierwszego uruchomienia środowiska zostaniemy poproszeni o wskazanie ścieżki do obszaru roboczego w którym chcemy obecnie pracować (**rysunek 6**). Po wpisaniu ścieżki, można również zaznaczyć opcję, aby ten obszar roboczy był od teraz do-

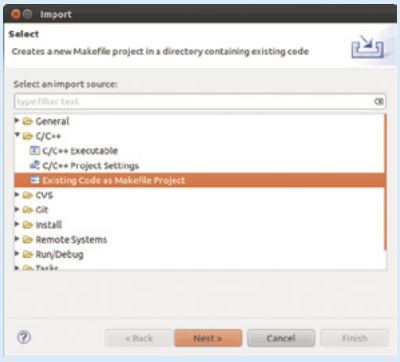
Rysunek 7. Ekran startowy (Eclipse)



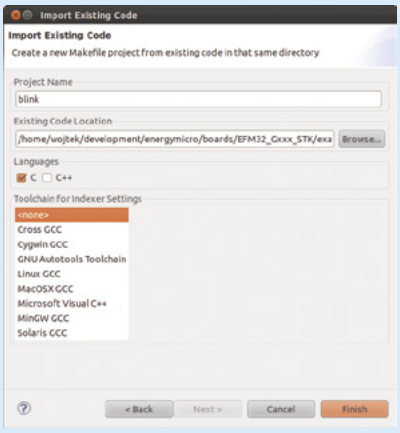
Rysunek 8. Podstawowy widok środowiska (Eclipse)

myślnym obszarem roboczym (w przyszłości można go zmienić klikając na *File* > *Switch Workspace* > *Other*). W chwili obecnej wskazujemy na ścieżkę w której znajdują się przykładowe projekty dla mikrokontrolerów *EFM32*: */home/wojtek/development/energymicro/boards/EFM32_Gxxx_STK/examples* i klikamy *OK*. Następnie klikamy na ikonę *Workbench* (**rysunek 7**) i naszym oczom powinien się ukazać podstawowy wygląd okien *Eclipse'a* (**rysunek 8**).

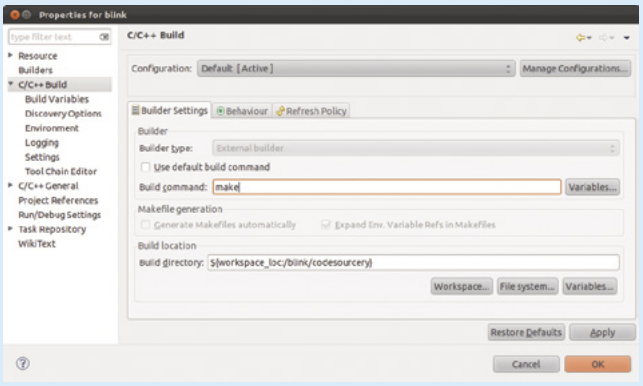
Po lewej stronie, w oknie zakładki *Project Explorer* klikamy prawym przyciskiem myszy, a następnie wybieramy opcję *Import...* Rozwijamy pozycję *C/C++*, wybieramy *Existing Code as Makefile Project* (**rysunek 9**) i klikamy *Next*. Dla pola *Existing Code Location* za pomocą przycisku *Browse* lub wpisując z klawiatury wybieramy ścieżkę do projektu mrugających diod: */home/wojtek/development/energymicro/boards/EFM32_Gxxx_STK/examples/blink* oraz odznaczamy pole języka *C++*, a jako *Toolchain* wybieramy *<none>* (**rysunek 10**). Klikamy *Finish*. Zaleca się zrobienie kopii zapasowej folderu *blink* w naszej lokalizacji, w razie gdybyśmy chcieli jeszcze z niego kiedyś skorzystać w oryginalnej wersji. Teraz niezbędne jest ustawienie naszego projektu tak, aby był on możliwy do kompilacji przez *Sourcery CodeBench*. W oknie *Project Explorer* klikamy prawym przyciskiem myszy na nasz projekt *blink* i wybieramy *Properties*. Po lewej stronie wybieramy pozycję *C/C++ Build*. W zakładce *Builder Settings* odznaczamy opcję *Use default build command* i upewniamy



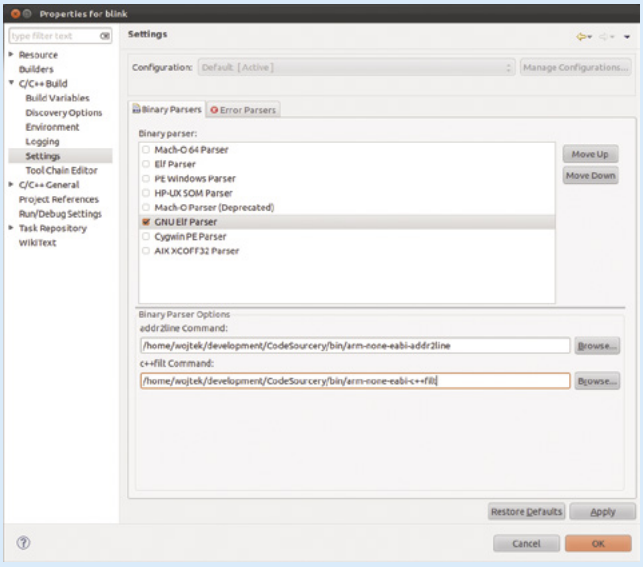
Rysunek 9. Importowanie przykładowego projektu



Rysunek 10. Konfiguracja projektu

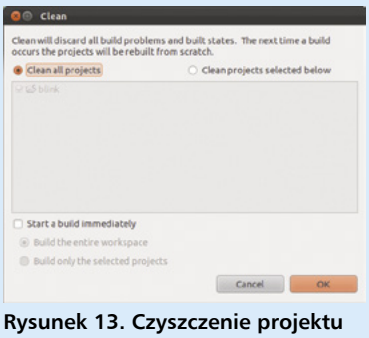


Rysunek 11. Wybór kompilatora



Rysunek 12. Wybór kompilatora c.d.

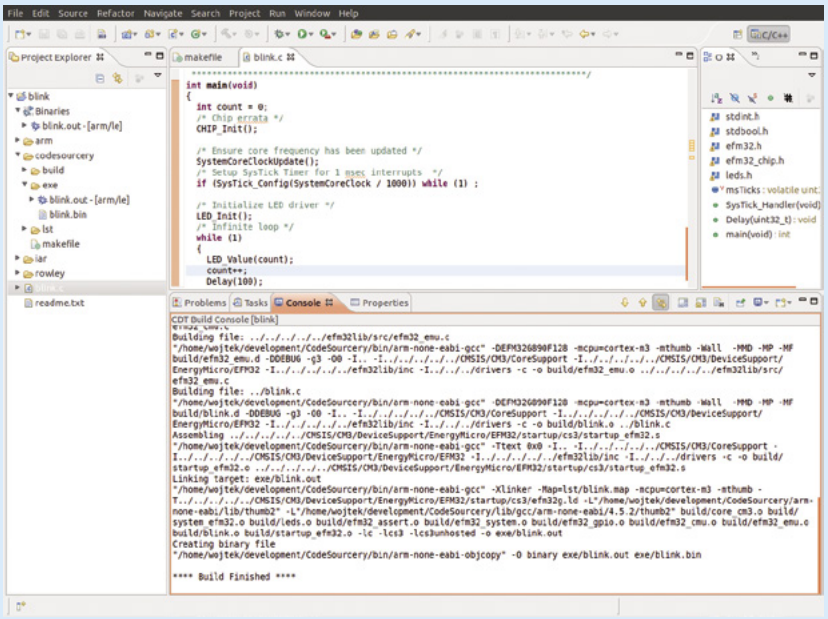
się, że w polu *Build command* znajduje się *make*. Następnie w dolnym obszarze okna klikamy przycisk *Workspace* i wybieramy z naszego projektu folder *codesourcery* jako lokalizację pliku *makefile* potrzebnego do zbudowania tego projektu. Klikamy *OK*. Teraz okno powinno wyglądać jak na **rysunku 11**. Klikamy przycisk *Apply*. Teraz przechodzimy po lewej stronie do pozycji *Settings* w *C/C++ Build*. W zakładce *Binary Parsers* zaznaczamy opcję *GNU Elf Parser*, a następnie w dwóch dolnych polach wybieramy pliki: */home/wojtek/development/CodeSourcery/bin/arm-none-eabi-addr2line* oraz */home/wojtek/development/CodeSourcery/bin/arm-none-eabi-c++filt* (**rysunek 12**), klikamy *Apply*, a następnie *OK*. Teraz odznaczamy w górnym menu opcję *Build Automatically*, aby za każdym razem automatycznie nie przebudowywał nam się projekt, co może być uciążliwe przy dużej ilości plików z kodem. Pozostaje jeszcze tylko edycja pliku *makefile*. W naszym projekcie w folderze *codesourcery* zmieniamy nazwę pliku *Makefile.blink* na *makefile* i edytujemy go. Ponieważ nie jest to artykuł o składni plików *makefile*, to zaznaczę tylko, że do poprawnej kompilacji naszego projektu należy zmienić linię: *LINUXCS = /cad/codesourcery/*



Rysunek 13. Czyszczenie projektu

arm-none-eabi/arm-2010q1 na *LINUXCS = /home/wojtek/development/CodeSourcery*

Wszytko jest już gotowe. Dla sprawdzenia poprawności kompilacji od samego początku wyczyśćmy jeszcze nas projekt. Klikamy w górnym menu na *Projekt > Clean*. Zaznaczamy opcję *Clean all projects*, a odznaczamy *Start a build immediately* (**rysunek 13**) i klikamy *OK*. Ostatecznie klikamy w menu *Project > Build All* lub naciskamy kombinację klawiszy *Ctrl+B*. Jeżeli nasz projekt zbuduje się bez błędów, to w logu okna *Console* powinien pojawić się pod koniec komunikat *Creating binary file*, a w oknie *Project Explorer* pliki wynikowe *blink.out* oraz *blink.bin* (**rysunek 14**), które są plikami wsadowymi do pamięci *flash* mikrokontrolera. Mogą służyć również do debugowania.



Rysunek 14. Listing po udanej kompilacji projektu

Jeżeli występują problemy z typem *uint32_t* w postaci błędu lub ostrzeżenia *Type ‚uint32_t' could not be resolved*, należy wejść w górne menu *Window > Preferences*, a następnie w opcję *Language Mappings* z listy *C/C++* i zmapować (za pomocą przycisku *Add*) *Content Type: C Header File* i *C Source File* z *Language: GNU C*, kliknąć *Apply*, następnie *OK* i zrestartować *Eclipse*.

Rysunek 15. Edycja projektu bez pomocy środowiska Eclipse

Podsumowanie

Posiadanie w swoim komputerowym zapleczu takiego darmowego, nieograniczonego, zintegrowanego środowiska programistycznego dla mikrokontrolerów *ARM* jest nieocenione w przypadku, gdy pracujemy na systemie operacyjnym Linux i nie mamy możliwości przełączenia się na Windowsa. W chwili obecnej wiele osób korzysta z netbooków, których ograniczone zasoby skłaniają do zastosowanie jednej z dystrybucji Linuxa zamiast Windowsa. Dzięki mobilności tych komputerów jesteśmy w stanie dokonywać poprawek kodu i kompilacji będąc „w terenie". Dodatkowo, niewątpliwym plusem jest możliwość edytowania i budowania projektu za pomocą linii komend w Linuxie. Choć tworzenie projektu może nie jest wygodne i efektywne za pomocą konsolowych edytorów, to w nagłych, awaryjnych przypadkach, o ile mamy zdalny dostęp (na przykład poprzez *SSH*) do komputera na którym jest zainstalowane linuxowe środowisko programistyczne, możemy za pomocą konsoli edytować pliki źródłowe (**rysunek 15**), a następnie będąc w katalogu *codesourcery* naszego projektu wpisać komendę *make* i skompilować projekt (lub na przykład *make clean* w celu wyczyszczenia projektu). Należy pamiętać, aby przy instalacji i konfiguracji środowiska podawać poprawne ścieżki dostępu, których składowe mogą się zmieniać, jak na przykład ścieżki zawierające folder domowy użytkownika czy lokalizacje projektów.

Wojciech Gelmuda
AGH Akademia Górniczo-Hutnicza