Kiss Zoltán - Endrich Bauelemente Vertriebs GmbH

Okoskijelzők használata - ember–gép interfész (HMI) megvalósítása DLOGIC modulokkal 2 rész

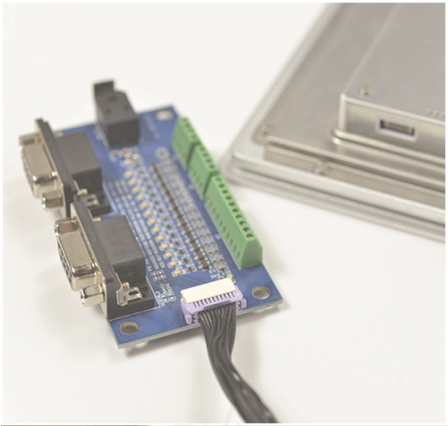
Cikksorozatunk első részében áttekintettük a DLOGIC okoskijelzőkkel megvalósítható ember-gép interfész (HMI – human–machine interface) technológiát. A hagyományos érintőképernyős TFT kijelzőkkel megvalósított megoldások esetén a hardver illesztés tervezésére és a rejtett hibák keresésére és kiküszöbölésére fordított idő hosszú lehet. Érdemes megfontolni kész, robosztus, megbízható és elfogadható árú okos kijelző modulok használatát, melyekkel a fejlesztési idő, ezáltal a termék piacra dobási ideje is jelentősen csökkenthető. Az előző részben ez utóbbi technológia előnyeit mutattuk be, a második részében a HMI konkrét megvalósításával foglalkozunk, áttekintjük a cross-platform fejlesztés folyamatát, és néhány példán keresztül a perifériák használatát is.

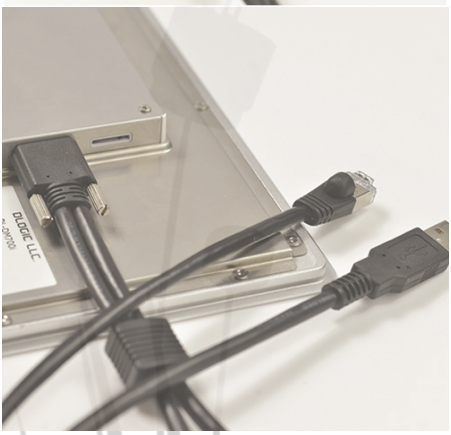
Az okoskijelző kereszt-platformos programozása

A HMI prototípusának kialakításához egy komplett DLOGIC fejlesztő készletre van szükség, mivel ez tartalmazza a kijelző modul mellett a fejlesztéshez szükséges többi komponenst is, mint pl. a beépítő keretet, tápegységet, kábeleket és egy a kommunikációs portok fizikai csatlakoztatásához kialakított panelt is. A gyártó értékesítési filozófiája szerint a sorozatgyártáshoz vásárolt termék dobozában csak az előretelepített okoskijelző kap helyet, hogy a lehető legolcsóbb legyen a megoldás. A szabványos csatlakozófelületről a megfelelő kábelezés úgyis egyedi igényekhez és megvalósítási tervhez igazodik, csak a használni kívánt sztenderd portok csatlakoztatására van szükség. A beépítés is egyedi módon valósul meg a gyakorlatban, míg a tápellátás kérdésében is előnyösebb a széles határok közti egyenáramú táplálhatóság biztosítása (9-36V), mint egy kötött 220V-os tápegység hozzáadása.

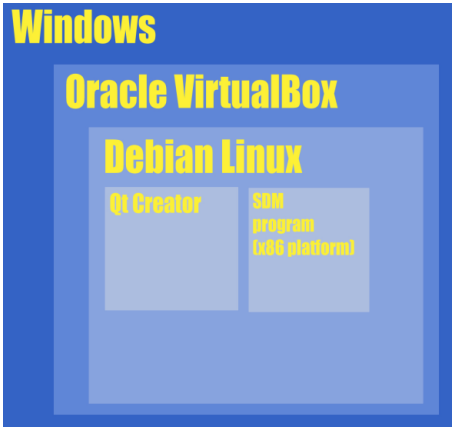
Az előzőleg megvásárolt fejlesztői csomaghoz jár természetesen a szoftver támogatás is, a kijelző modul előretelepített BSP-vel érkezik a cross-platform szoftver fejlesztői környezetet

endrich

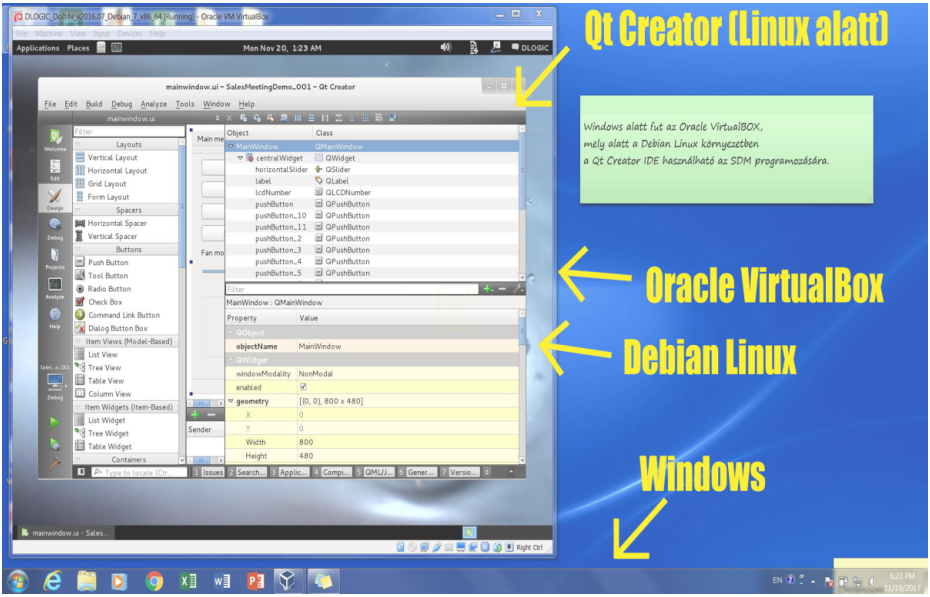




pedig a készülék Mac címének regisztrálása után a gyártótól egyedileg kialakított virtuális gép image fileként kapjuk. Az x86 alapú fejlesztői (jellemzően Windows operációs rendszert futtató) számítógépünkre valamilyen virtuális gép programot, például az ingyenes Oracle VirtualBox-ot telepítve néhány perces munkával elkészíthetjük a lokális Linux fejlesztőrendszert.

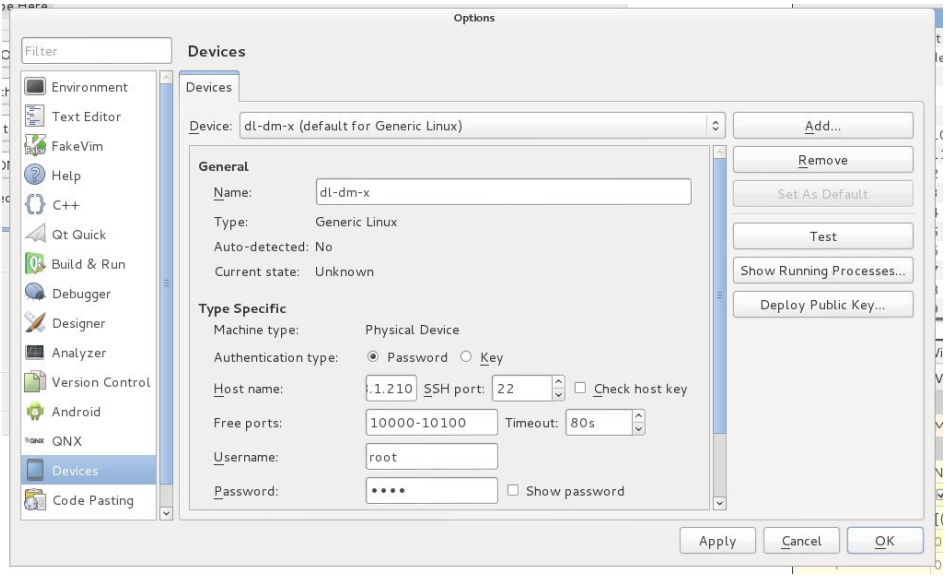


A Debian Linux disztribúció és a QT grafikai környezet ideális szoftveres alapot biztosít a megbízható működésre, a grafikus könyvtár és a hozzá tartozó eszközök gyors alkalmazásfejlesztést tesznek lehetővé, ezáltal a mai kornak megfelelő grafikus felhasználói felületek alakíthatók ki az érintőképernyőt igénylő alkalmazások számára. A kereszt-platformos fejlesztés lényege, hogy a fejlesztőmérnök saját, általában x86 alapú számítógépén Windows vagy MacOS alatt futtatja egy előre telepített virtuális gépen a DLOGIC által biztosított Debian alapú előre konfigurált grafikus fejlesztőkörnyezetet, majd az elkészült, kipróbált programot ARM kódra konvertálva az SDM-en futtatja.



Az x86 PC és az SDM ugyanazon Ethernet hálózathoz kapcsolódik, így biztosított a TCP/IP alapú kommunikáció akár terminál emulátoron keresztül, akár SFTP kapcsolattal a két rendszer között.

A virtuális gép telepítése után a fejlesztőkörnyezet a vásárolt SDM típusához igazítva mindenféle további paraméterezés nélkül azonnal rendelkezésre áll, egyetlenegy beállításra van csak szükség, meg kell mondanunk a rendszer számára, hogy a kijelző modul milyen IP címen érhető el a fenti ábrán látható mini lokális hálózaton. Ehhez a bekapcsolt SDM-re telepített hálózati konfigurációs segédprogramot kell

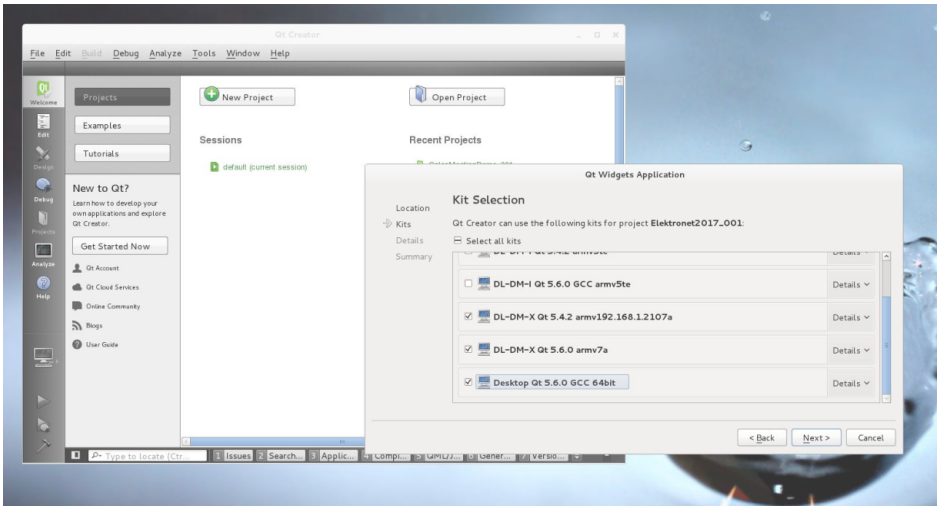


elindítani, ahol beállítható statikus és dinamikus IP cím is. Az utóbbi esetben a routerben be kell kapcsolni a DHCP opciót. A kijelzőn megjelenő IP címet kell a Qt Creator Tools/Option/Devices menüjén keresztül az előretelepített kit „Host Name” változójába beírni. Ezután indulhat is az okos-kijelző programozása.

Qt Creator projekt indítása

A Qt creator keresztplatformos alkalmazásfejlesztő keretrendszer segítségével többféle típusú applikáció készíthető. Ezek némelyike az 5.6 változattól fogva csak grafikus processzor megléte esetén működik, így „i” sorozatú (Freescale Freescale Arm 9, iMX257 alapú), olcsóbb DLOGIC SDM-eken csak a „Widget” applikációk fejleszthetők, emiatt példáinkban is ilyen projektet mutatunk be. Megjegyzendő, hogy korábbi verziószámú Qt platform GPU hiánya esetén támogja a szoftveres grafikus renderelést, így egyszerűbb Quick és Canvas 3D projektek is használhatóak az „i” sorozatú SDM-eken, ha az erőforrásigény nem nagy. A Qt fejlesztésre elsősorban a C++ nyelvet támogatja, de más nyelvekre is elérhető, valamint rendelkezik saját leíró nyelvvel (Qt Quick)

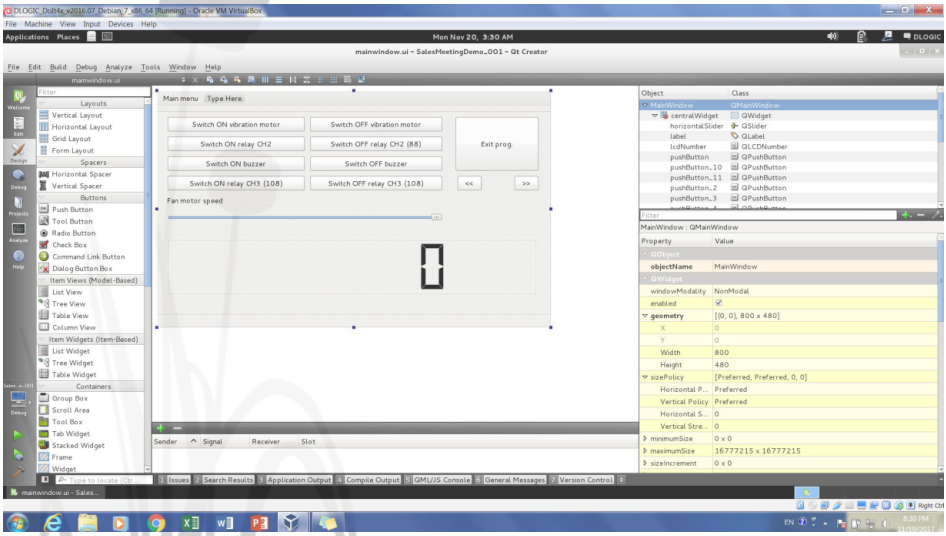
Az általunk választott alkalmazás típusban rengeteg előre definiált vezérlőelem (Widget = Windows Gadget), mint például nyomógomb, vízszintes és függőleges görgetősáv, checkbox, radiobutton áll rendelkezésre korszerű grafikus felhasználói felület létrehozásához. Ezek szabadon elhelyezhetők a tervezendő képernyőn. A grafikus felület objektumorientáltan épül fel, a vezérlők tulajdonságai részletesen



beállíthatók. A program által kezelendő interakciókat, mint például a gomb megnyomása, dupla kattintás, sáv görgetése eseménykezelő szubrutinok megírásával végezzük (C++). Ezekre a későbbiekben mutatunk be példákat is.

Az új projekt létrehozásakor ki kell választanunk az előre felépített „kit”-készletből, hogy mely célhardveren fogunk dolgozni. Természetesen egyszerre több ilyen is választható, de a projekten belül csak azok, amelyeket létrehozáskor beállítottunk. Célszerű mindenképpen kiválasztani az SDM-et és a lokális gépet, mint elérhető célt.

A projekt indításakor a „MainWindow.ui” kiválasztásával grafikusan megtervezhetjük a képernyőt, amelyen elhelyezhetők az egyes



vezérlők, melyeket szeretnénk használni az ember-gép interfész megvalósításakor. Példánkban néhány nyomógombot, vízszintes csúszkát és 7 szegmenses kijelzőt helyeztünk el. A nyomógombokkal a DLOGIC SDM egyes I/O portjainak ki- és bekapcsolását, a vízszintes csúszkával egy PWM kimenet kitöltési tényezőjének állítását, a szegmens-kijelzővel pedig a választott opció kiíratását végezzük. Példánkban az egyszerűséget és az átláthatóságot tartottuk szem előtt, természetesen összetettebb grafikus elemek használatával korszerű és trendi felhasználói interfész alakítható ki.
Mielőtt rátérünk az eseménykezelők megírására, szeretnénk bemutatni az egyes hardver interfészek, mint például a GPIO portok, illetve a PWM kimenetek használatát LINUX alatt, hogy könnyen megérhető legyen ezek programozása.

GPIO portok kezelése

A DLOGIC SDM Linux operációs rendszere a fizikai GPIO portok kezelését hozzárendelt állományok manipulálásával teszi lehetővé. Az elérhető portok mindegyikéhez létezik egy könyvtár, amely az I/O port tulajdonságait tartalmazó állományokat tartalmazza (pld a 88-as I/O porthoz: **`/sys/class/gpio/gpio88`**).

Az egyes állományok a GPIO port jellemzőit tartalmazzák és ezek egyszerű fájlműveletekkel megváltoztathatók.
Az irányultság megváltoztatásához (bemenetté tétel) az „in” értéket kell a direction állományba írni au **`echo in > direction`** paranccsal, vagy kimenetté a **`echo out > direction`** paranccsal tehető ugyanez a port.

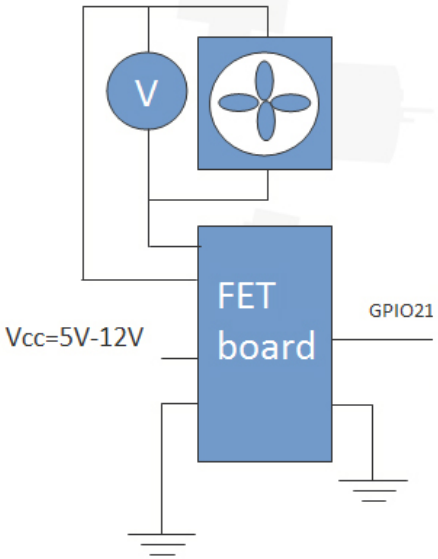
A port értéke a **`value`** állomány írásával állítható be. Magas logikai szintre az **`echo 1> value`** paranccsal, míg logikai „0” szintre az **`echo 0 > value`** paranccsal állítható. Ha a HMI egyik digitális kimenetéhez egy FET, vagy relémodul kapcsolódik, akkor ezzel az egyszerű állományművelettel ki-, vagy bekapcsolhatunk egy akár 220V-os elektromos fogyasztót. (Szeretném megjegyezni, hogy az SDM ARM processzorának 3.3V-os tápfeszültsége korlátozza a GPIO kimenetek feszültségszintjét is, ezért szükséges lehet egy fizikai szintillesztés, ha például 5V-os relémodult használunk.)

```
root@dlogic-dm:/sys/class/gpio# cd gpio88
root@dlogic-dm:/sys/class/gpio/gpio88# ls
active_low  direction   power       uevent
device      edge        subsystem   value
```

PWM eszköz kezelése

Az impulzusszélesség modulált kimenetek egyenfeszültségű táplálás effektív feszültségértékének csökkentéséhez használatosak oly módon, hogy periódusonként bizonyos időre megszakítják a táplálást, például egy FET be- és kikapcsolásával. A bekapcsolt és kikapcsolt állapot egymáshoz képesti időtartama (kitöltési tényező) határozza meg a feszültség effektív értékét. Az ábrán egy 5...12V ventilátor sebességének szabályzását végezzük közvetlenül a GPIO21 PWM kimeneten keresztül egy FET modul közbeiktatásával. Az 50%-os kitöltési tényező beállításával a tápfeszültség fele jut csak a kapcsokra, így a forgási sebesség lecsökken.

Az **`mxc_pwm.0`** eszközhöz tartozó könyvtárban lévő **`periodns`** állományba a periódusidő értékét, a **`dutyns`** állományba pedig a duty cycle [nsec] értékét kell beírni.



Kiemelt szerepe van a DLOGIC SDM PWM-2 eszközének, melyhez a TFT háttérvilágítás vezérlése van kötve, azaz ennek az eszköznek a manipulálásával a felhasználó állíthatja be a kijelző fényerejét.

```
root@dlogic-dm:/sys/devices/platform/mxc_pwm.0# cat dutyns
0
root@dlogic-dm:/sys/devices/platform/mxc_pwm.0# echo 100000 > dutyns
root@dlogic-dm:/sys/devices/platform/mxc_pwm.0# cat dutyns
100000
```

```
root@dlogic-dm:/sys/class/backlight/pwm-backlight.1# cat max_brightness
500
root@dlogic-dm:~# cd /sys/class/backlight/pwm-backlight.1
root@dlogic-dm:/sys/class/backlight/pwm-backlight.1# echo 100 > brightness
root@dlogic-dm:/sys/class/backlight/pwm-backlight.1# cat brightness
100
```

Példa egy HMI megvalósítására

A példában bemutatott egyszerű felhasználói felület néhány eszköz GPIO portokon keresztüli ki- és bekapcsolásához, valamint egy ventilátor sebességének impulzusszélesség modulációval való szabályzásához rendelt vezérlőt tartalmaz. A projekt megnyitása után a program „.pro” állományához meg kell adnunk a futtatási környezetet a mellékelt kód hozzáfűzésével, ellenkező esetben az ARM bináris állomány SDM-en való futtatásakor hibaüzenet jelenhet meg, ha a környezeti változók esetleg nem lettek beállítva. Érdemes ezt a programban megadni.

A GPIO portok kezeléséhez a vonatkozó nyomógombok megfelelő esemény-vezérlő rutinjait kell megírnunk. A kezelendő interakció a gomb megnyomása, így a „clicked()” eseményt szükséges leprogramozni. A port ki- és bekapcsolása a fentiekben leírt állományművelettel valósítható meg. Ehhez a mellékelt rutinokat kell megírnunk. A GPIO 88 port bekapcsolását és kikapcsolását a megfelelő „value” állományba „1” és „0” értékek beírásával vezéreljük. Az lcdNumber vezérlő a 7 szegmenses kijelzőnk, értékének beírásával kijelezhetjük a HMI-n utoljára kiadott parancsot.

A ventilátor sebességének szabályzásához egy vízszintes csúszkát

```
#-------------------------------------------------
#
# This is to define runtime environment at target
# device kit (embedded panelPC)
#-------------------------------------------------

unix:!android {
    isEmpty(target.path) {
        qnx {
            target.path = /tmp/$${TARGET}/bin
        } else {
            target.path = /opt/$${TARGET}/bin
        }
        export(target.path)
    }
    INSTALLS += target
}

export(INSTALLS)
```

```
void MainWindow::on_pushButton_clicked()
{
    QString filename = "/sys/class/gpio/gpio88/value";
    QFile file(filename);
    if (file.open(QIODevice::ReadWrite )) {
        QTextStream out(&file);
        out << "1";
        ui->lcdNumber->display("088    ");
    }
}

void MainWindow::on_pushButton_2_clicked()
{
    QString filename = "/sys/class/gpio/gpio88/value";
    QFile file(filename);
    if (file.open(QIODevice::ReadWrite )) {
        QTextStream out(&file);
        out << "0";
        ui->lcdNumber->display("088 OFF");
    }
}
```

használunk. A beállított PWM frekvencia 25 kHz, tehát a periódusidő 40000 nsec. Ezt kell a megfelelő **periodns** állományba írnunk a program inicializálásakor. Ehhez a MainWindow eszköz létrejöttekor generált eseményt használhatjuk.

A duty cycle (kitöltési tényező) értékét a vízszintes csúszkával szeretnénk beállítani, melynek értéke 0...100

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    QString filename = "/sys/devices/platform/mxc_pwm0/periodns";
    QFile file(filename);
    if (file.open(QIODevice::ReadWrite )) {
        QTextStream out(&file);
        out << "40000";

    }
ui->lcdNumber->display("SET");

}
```

tartományban változik. A 100 értékhez tartozik a 40000 ns (100%) érték, így lineárisan interpolálva az aktuális értékhez a value*400 [ns] duty cycle tartozik. Ezt a csúszka értékének változását jelentő esemény kezelő rutinjában írjuk be.

```
void MainWindow::on_horizontalSlider_sliderMoved(int position)
{
}

void MainWindow::on_horizontalSlider_valueChanged(int value)
{

    QString filename = "/sys/devices/platform/mxc_pwm0/dutyns";
    QFile file(filename);
    if (file.open(QIODevice::ReadWrite )) {
        QTextStream out(&file);
        out << value*800;
    }
    ui->lcdNumber->display(round(value*400));
}
```

A fentieket megismételve minden használni kívánt vezérlő eseménykezelőjének megírásához lassan eljutottunk oda, hogy a kész programot tesztelhetjük is. A Qt grafikai környezetben beállítható, hogy az elkészített alkalmazás fordítás után hol fusson, a lokális x86 gépen, vagy exportálni szeretnénk az elkészült ARM alapú bináris futtatható állományt az SDM-re. Először érdemes a „Desktop PC” opciót választani és azonnal elindul a program a fejlesztéshez használt számítógépen. Itt csak a megjelenítés és a hardver független funkciók tesztelésére van mód, hiszen az asztali számítógépen nincsen összerendelve a fájlrendszer a GPIO portokkal, PWM kimenetekkel, mert azok az SDM sajátjai. Természetesen ezeket az állományokat kézzel létrehozva azok változása nyomon követhető. Igazi funkcionális teszt majd az ARM bináris kód exportálása után végezhető magán a kijelző modulon.

A keresztplatformos fejlesztés előnye, hogy amint a Qt Creatorban megváltoztatjuk a céleszközt az ARM alapú SDM-re, a lefordított bináris kód azonnal felkerül arra és elindul a futása.

A fent ismertetett módszerrel tehát szoftveres úton egyszerűen alakítható ki vagy változtatható meg a kijelző felépítése, működése és egyszerű vezérlési funkciók is integrálhatóak.