1. Giriş

Bu dokümanda, Qt Creator IDE'si ile nasıl DDS uygulamaları geliştirebileceğimizi irdeleyeceğiz. Yazılan DDS uygulamaları, C++ dilinde olacak ve Qt Creator'ın C++ diline odaklı geliştirdiği IDE'si sayesinde problemli kaynak kodları kolayca 'debug' (hata ayıklama) işlemleri yapabileceğiz. Eğer daha önce Qt Creator ile çalışılmamış ise basit bir 'HelloWorld' C++ projesi Qt Creator'da nasıl derleneceğine dair herhangi bir 'online tutorial' izlenmesi tavsiye edilir. Böylece temel menu fonksiyonlarına hakimiyet kazanılmış olacaktır.

Tutorial: https://www.youtube.com/playlist?list=PL2D1942A4688E9D63

Ayrıca, DDS uygulamaları tasarlanırken Prismtech firmanın geliştirdiği Opensplice DDS Community Edition kullanılmıştır.

2. Qt Creator'ın Yüklenmesi

Aşağıdaki linkteki tutorialı katip ederek, Qt5 kütüphanelerini ve Qt Creator'ı yükleyebilirsiniz

http://www.bogotobogo.com/Qt/Qt5_Install_Ubuntu_14_64bit.php

3. Opensplice DDS'in Yüklenmesi

Aşağıdaki linkten, DDS Community Edition Version 6.x for Linux kernel 3 and up (64-bit) Host and Target, gcc 4.6 compiler, x86 chipset (Ubuntu 12) versiyonunu indirin.

http://www.prismtech.com/dds-community/software-downloads

Not 1: Linkten indirdiğiniz DDS'i, PC'deki Downloads klasörüne indirdiğinizi farz edilerek aşağıdaki işlemleri gerçekleştirilmiştir.

Not 2: '<xxx>' ile belirtilen yerler Ubuntu'daki hostname'e aittir.

Not 3: Eğer hostname'nizi bilmiyorsanız '`$ hostname`' komutu ile öğrenebilirsiniz.

```
$ cd ~/
$ mkdir opt
$ mv ~/Downloads/HDE ~/opt
$ cd ~/opt/HDE/x86_64.linux
$ sudo chmod +x release.com
$ gedit release.com
```

```
echo "<<< OpenSplice HDE Release V6.4.140407OSS For x86_64.linux,
Date 2014-04-15 >>>"
if [ "${SPLICE_ORB:=}" = "" ]
then
    SPLICE_ORB=DDS_OpenFusion_2
    export SPLICE_ORB
fi
if [ "${SPLICE_JDK:=}" = "" ]
then
    SPLICE_JDK=jdk
    export SPLICE_JDK
fi
BOOST_ROOT="/usr/include/boost"
OSPL_HOME="/home/<xxx>/opt/HDE/x86_64.linux"
PATH=$OSPL_HOME/bin:$PATH
LD_LIBRARY_PATH=$OSPL_HOME/lib${LD_LIBRARY_PATH:+:}$LD_LIBRARY_PATH
CPATH=$OSPL_HOME/include:$OSPL_HOME/include/sys:${CPATH:=}
OSPL_URI=file://$OSPL_HOME/etc/config/ospl_unicast.xml
OSPL_TMPL_PATH=$OSPL_HOME/etc/idlpp
. $OSPL_HOME/etc/java/defs.$SPLICE_JDK
export OSPL_HOME PATH LD_LIBRARY_PATH CPATH OSPL_TMPL_PATH OSPL_URI
BOOST_ROOT
```

```
# Eğer yukarıdaki source işlemini doğru yapıp yapmadığınızını kontrol
# etmek istiyorsanız, aşağıdaki komutu deneyin. Eğer çıktısı
# release.com'da tanımladığınız "/home/<xxx>/opt/HDE/x86_64.linux" ise
# işlem doğru gerçekleşmiş demektir.

$ source ~/opt/HDE/x86_64.linux/release.com
$ echo $OSPL_HOME
```

Terminal 1: DDS'e Ait 'release.com' Dosyasının Düzenlenmesi

4. Derleme Öncesi Gerekli Paketlerin Yüklenmesi

```
$ sudo apt-get install gcc g++
$ sudo apt-get install autoconf
$ sudo apt-get install automake
$ sudo apt-get install build-essential
$ sudo apt-get install libboost-all-dev
```

5. DDS'e Ait Environment Değişkenlerin Tanımlanması

```
$ sudo gedit /etc/environment

# Açılan dosyaya aşağıdaki metni yapıştırın.

    PATH=/home/bolatu/opt/HDE/x86_64.linux/etc/idlpp:/home/bolatu/opt
    /HDE/x86_64.linux/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/u
    sr/bin:/sbin:/bin:/usr/games:/usr/local/games
    OSPL_HOME=/home/bolatu/opt/HDE/x86_64.linux
    OSPL_TARGET=x86_64.linux
    LD_LIBRARY_PATH=/home/bolatu/opt/HDE/x86_64.linux/lib
    CPATH=/home/bolatu/opt/HDE/x86_64.linux/include://home/bolatu/opt
    /HDE/x86_64.linux/include/sys:/home/bolatu/opt/HDE/x86_64.linux/i
    nclude/dcps/C+
    +/SACPP:/home/bolatu/opt/HDE/x86_64.linux/include/dcps/C+
    +/CCPP:/home/bolatu/opt/HDE/x86_64.linux/include/dcps/C++/isocpp
    OSPL_URI=file://home/bolatu/opt/HDE/x86_64.linux/etc/config/ospl.
    xml
    OSPL_TMPL_PATH=/home/bolatu/opt/HDE/x86_64.linux/etc/idlpp
    QTDIR=/home/bolatu/opt/Qt5.5/5.5/gcc_64/
    CLASSPATH=/home/bolatu/opt/HDE

# Sistem yeniden başlatılıyor!

$ sudo reboot
```

Terminal 2: DDS İçin Gerekli Evironment Değişkenlerinin Tanımlanması

6. DDS Source (Kaynak) Kodları ve IDL Dosyası

Bu bölümde, bir DDS projesi için gerekli source kodları kendi yazdığımız bir IDL dosyasından nasıl üreteceğimizi inceleyeceğiz.

DDS'nin çalışma mekanizması kısaca hatırlatılmak istenirse; dağıtık sistemlerde gerçek-zamanlı bir haberleşmeyi hedefleyen ve bu haberleşme sırasında Publisher/Subscriber mesajlama modelini

kullanan ara katman yazılımıdır. Publisher (yayımlayıcı) tarafından yayımlanan herhangi bir mesaja ister lokalde çalışan bir DDS uygulamasına ait bir Subscriber (aboneci) tarafıdan, isterse mesaj kaynağından uzaktaki bir DDS uygulamasına ait Subscriber tarafından abone olunabilir. Gönderilen bu mesajın Publisher ve Subscriber tarafından belirli bir standarda uygun bir şekilde okuma ve yazma yapması elzemdir. Bunun için de DDS, IDL veri tipi standardını kullanmaktadır.

IDL hakkında kaynak: http://www.omg.org/gettingstarted/omg_idl.htm

RFKON projesi için geliştirilen DDS uygulamalarında kullanılan IDL dosya içeriği aşağıdaki gibidir.

```
module KonSensData
{
   struct Msg
   {
      string devID;
       string hostName;
      long dbm;

   };
   #pragma keylist Msg devID

   typedef sequence<Msg> MesSeq;

   struct WifiSeq
   {
            long timestamp[2][2];
            long userID;
            long messageID;
            MesSeq messages;
   };
   #pragma keylist WifiSeq userID

   struct BtSeq
   {
                long timestamp[2][2];
                long userID;
            long messageID;
                MesSeq messages;
   };
   #pragma keylist BtSeq userID

   struct ServerReq
   {
      string request;
      long requestID;

   };
   #pragma keylist ServerReq requestID
};
```

Kod 1: KonSensData IDL Veri Tipi Formatı

Şekil 1'deki kod bloğu, DDS Topic'lerin arasında gerçekleşecek olan veri transferinde kullanılacak olan IDL formatındaki veri tipini ifade etmektedir. Oluşturduğumuz bu IDL dosyası, Opensplice DDS'nin içerisinde bulunan 'idlpp' executable (yürütülebilir) dosyası ile gerekli C++ source kodları üretiminde kullanılacaktır.

```
$ cd Downloads
$ git clone https://github.com/inomuh/rfkon
$ mkdir DDS_Project_with_Qt
$ cd DDS_Project_with_Qt
$ cp ~/Downloads/rfkon/Server_Requester/KonSensData.idl
~/DDS_Project_with_Qt
$ source ~/opt/HDE/x86_64.linux/release.com
$ idlpp -S -l cpp KonSensData.idl

# İşlem sonrasında yaratılan source kodları görüntülemek için aşağıdaki
# komutu kullanabilirsiniz.
$ ls
```

Terminal 3: IDL Dosyasında C++ Source Dosyaları Üretilmesi

Terminal 3'de gerçekleştirilen komutlar sonrasında;

- ccpp_KonSensData.h
- KonSensDataDcps.cpp
- KonSensDataDcps_impl.cpp
- KonSensData.h
- KonSensDataSplDcps.cpp
- KonSensData.cpp
- KonSensDataDcps.h
- KonSensDataDcps_impl.h
- KonSensDataSplDcps.h

C++ dilinde yaratılmış source kodları elde etmiş oluruz. Bu source kodlar; DDS Domain, DDS Topicleri, DDS Publisher/Subscriber, DDS DataWriter/DataReader vb. nesnelerini yaratmada kullanılacak sınıflardır.

7. KonSens İçin Geliştirilmiş DDS Projeleri

Aşağıdaki komut isteğe bağlıdır. Eğer, yukarıda tasarladığımız KonSensData.idl formatında mesaj alışverişi yapılmak isteniyorsa aşağıdaki adımlar izlenebilir. Fakat, KonSensData.idl'den farklı bir DDS uygulaması için yukarıdaki adımlar, yeni yaratılmış idl dosyası için yapılmalı ve aşağıdaki 'ServerRequester.cpp' source dosyası yerine, yeni geliştirilen veri tipine uygun DDS projeleri tasarlanmalıdır. Kısaca, aşağıdaki adımlar sadece 'ServerRequester' fonksiyonunu kullanabilmek için yapılmıştır.

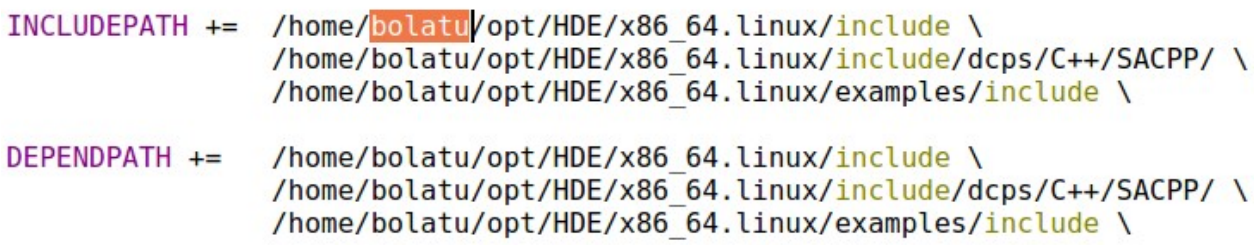
```
# Not: Aşağıdaki kopyalama işlemleri her proje içeriğine göre
# değişecektir. Bu yüzden Terminal 3'de 'idlpp' komutu ile oluşturduğumuz
# ccpp_KonSensData.h, KonSensDataDcps.cpp, KonSensDataDcps_impl.cpp,
# KonSensData.h, KonSensDataSplDcps.cpp, KonSensData.cpp,
# KonSensDataDcps.h, KonSensDataDcps_impl.h, KonSensDataSplDcps.h
# dosyaları haricindeki tüm dosyalar kopyalanmalıdır.

# Yapılan kopyalama işlemleri Server_Requester projesi için geçerlidir!
$ cp ~/Downloads/rfkon/Server_Requester/DDSEntityManagerKonSens*
~/DDS_Project_with_Qt
$ cp ~/Downloads/rfkon/Server_Requester/CheckStatus* ~/DDS_Project_with_Qt
$ cp ~/Downloads/rfkon/Server_Requester/Timeout* ~/DDS_Project_with_Qt
$ cp ~/Downloads/rfkon/Server_Requester/tcp* ~/DDS_Project_with_Qt
$ cp ~/Downloads/rfkon/Server_Requester/ServerRequester.cpp
~/DDS_Project_with_Qt
$ cp ~/Downloads/rfkon/Server_Requester/Server_Requester.pro
~/DDS_Project_with_Qt
```

Terminal 4: ServerRequester ve Diğer Gerekli Source Dosyalarının Kopyalanması

8. Qt Creator'da DDS Projesinin Yaratılması ve Derlenmesi

Geriye, DDS_Project_with_Qt klasörüne topladığımız tüm dosyları derleme işlemi kaldı. Qt Creator'da, 'Open Project→DDS_Project_with_Qt→Server_Requester.pro' yolunu izeleyerek ServerRequester projesini açabilirsiniz.

Daha sonra derleme işlemine geçmeden öncede Server_Requester.pro dosyasının içerisindeki DDS kütüphanelerinin çoktan yazıldığını göreceksiniz. Burada size düşen tanımlanan hostnameler yerine kendinizinki ile değiştirmektir.



Şekil 1: Server_Requester.pro Dosyasındaki Hostname Değişikliği

Eğer hala anlaşılmayan bir nokta var ise dokümana ait tutorial videosunu izleyin.

Dokümana ait tutorial video linki: https://www.youtube.com/watch?v=3y8ua0ahqho