

******Draft******

Custom Raspbian Image with Lazarus FPC installed

=

image_2018-03-20-RaspbianUltibo.zip

Created with pi-gen

Testing Lazarus/FPC

Comparing number files needed

for VideoCore Apps

Raspbian & Ultibo

03/21/18

******Draft******

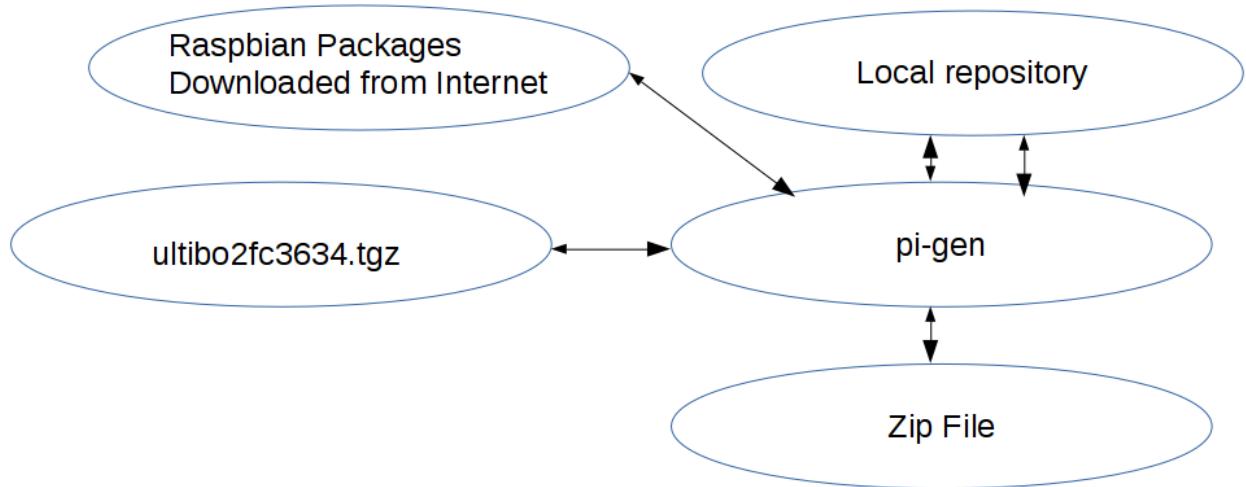
Introduction:

The overall goal is to create a bootable micro sd card ready to start developing Ultibo applications using Lazarus IDE (Ultibo Edition).with Free Pascal Compiler FPC. This was made possible with creation of the “*ultiboinstaller.sh*” found at

<https://github.com/ultibohub/Tools/blob/master/Installer/Core/Linux/ultiboinstaller.sh> by Gary from Ultibo, “*pi-gen*” and the “*Stretch Raspbain release*” make this possible. Ultibo released an installer script “*utiboinstaller.sh*” that installs Lazarus and Free Pascal Compiler FPC on the Raspberry Pi. This marks a milestone in the Ultibo Bare Metal development providing an Integrated Development Environment (IDE). The Stretch Raspbian release still needs several dependencies to install Lazarus IDE (Ultibo Edition).with Free Pascal Compiler FPC. In addition, other tools such as arm-none-eabi tools, telnet, tftp, gitk, diffuse, hexedit are desired.

This is why pi-gen was used to get the system up quickly.

The target system is Linux based since the Linux firmware plus the kernel7.img or kernel.img are what make ultibo run on the RPi hardware.
We currently have a Lazarus/FPC executable that runs on Windows System.
This still requires the binutils and arm-none-eabi compiler.



Tool used to create the raspberrypi.org Raspbian images. The Debian packages plus Lazarus IDE (Ultibo Edition) which was created with ***utiboinstaller.sh*** script and put in ***ultibo2fc3634.tgz***.

The file ***stage4/02-extrase/01-run.sh*** installs the ultibo tree in rootfs /home/pi.

#!/bin/bash -e

```

tar xfv files/ultibo2fc3634.tgz -C ${ROOTFS_DIR}/home/pi/
chown 1000:1000 ${ROOTFS_DIR}/home/pi/ultibo -Rv
cp files/ultibo.desktop ${ROOTFS_DIR}/home/pi/.local/share/applications
chown 1000:1000 ${ROOTFS_DIR}/home/pi/.local/share/applications/ultibo.desktop
cp ${ROOTFS_DIR}/etc/dphys-swapfile ${ROOTFS_DIR}/etc/dphys-swapfile.orig
sed s/100/1000/ < ${ROOTFS_DIR}/etc/dphys-swapfile.orig > ${ROOTFS_DIR}/etc/dphys-
swapfile

```

The file ***stage4/00-install-packages/02-packages***

```

libgtk2.0-dev
libcairo2-dev
libpango1.0-dev
libgdk-pixbuf2.0-dev
libatk1.0-dev

```

*libghc-x11-dev
binutils-arm-none-eabi
gcc-arm-none-eabi
diffuse
gitk
tcl-dev
telnet
tftp
hexedit
cmake
flex
bison*

The packages above provide the dependencies for Lazarus IDE (Ultibo Edition) and tools needed to connect with Ultibo bare metal systems and transfer kernel7.img or kernel.img.

Creating the boot & rootfs in zip file that can be transferred to the micro sd card:
This required several steps found in *Appendix A: Notes during pi-gen testing*

Testing creating multiple micro sd cards

```
date > t1.txt; gzip -dc image_2018-03-20-RaspbianUltibo.zip | dd bs=16M of=/dev/sdb; date >> t1.txt
gzip: image_2018-03-20-RaspbianUltibo.zip: invalid compressed data--length error
gzip: image_2018-03-20-RaspbianUltibo.zip: invalid compressed data--length error
0+147301 records in
0+147301 records out
7730102272 bytes (7.7 GB, 7.2 GiB) copied, 1035.77 s, 7.5 MB/s
```

Thu Feb 15 17:37:09 MST 2018
Thu Feb 15 17:54:25 MST 2018

```
date > t2.txt; gzip -dc image_2018-03-20-RaspbianUltibo.zip | dd bs=16M of=/dev/sdc; date >> t2.txt
gzip: image_2018-03-20-RaspbianUltibo.zip: invalid compressed data--length error
0+130267 records in
0+130267 records out
7730102272 bytes (7.7 GB, 7.2 GiB) copied, 1189.3 s, 6.5 MB/s
```

Thu Feb 15 17:37:17 MST 2018
Thu Feb 15 17:57:06 MST 2018

```
date > t3.txt; gzip -dc image_2018-03-20-RaspbianUltibo.zip | dd bs=16M of=/dev/sdd; date >> t3.txt
gzip: image_2018-03-20-RaspbianUltibo.zip: invalid compressed data--length error
0+115177 records in
0+115177 records out
7730102272 bytes (7.7 GB, 7.2 GiB) copied, 1103.39 s, 7.0 MB/s
```

Thu Feb 15 17:37:27 MST 2018
Thu Feb 15 17:55:50 MST 2018
Three micro sd card were created in 18 mins..

Comparing number files needed for VideoCore Apps Raspbian & Ultibo.

Ultibo only requires a few files.

bootcode.bin

start.elf

fixup.dat

kernel.img RPi Zero or kernel7.img (RPi2B or RPi3B)

test.h264 needed for hello_videocube.bin

***Several thousand files are needed with an O/S
Below are the numbers of the main directories.***

/usr/bin

1 directory, 1411 files

/usr/lib

3632 directories, 33202 files

sudo su

First need to compile the libraries

cd /opt/vc/src/hello_pi/libs/ilclient

make

cd /opt/vc/src/hello_pi/libs/vgfont

make

Next compile and test the apps

cd ../../hello_videocube/

make

./hello_videocube.bin

cd ../../hello_tiger/

make

./hello_tiger.bin

The following 4 images were complied on Rpi2B for a Rpi Zero. The first 2 are 2D image of Rotating Tiger and 3rd & 4th are of a Rotating cube.

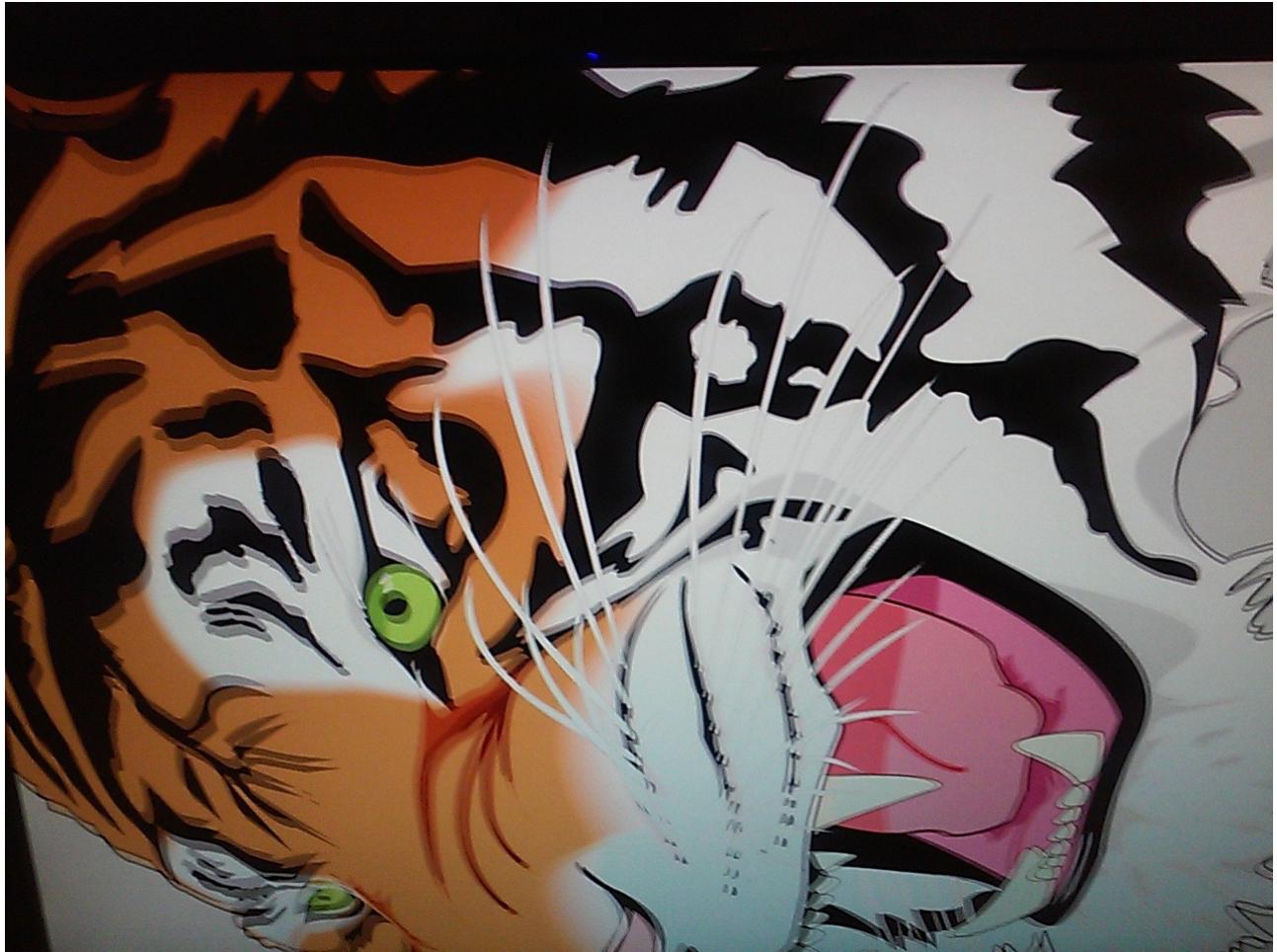


Image 1

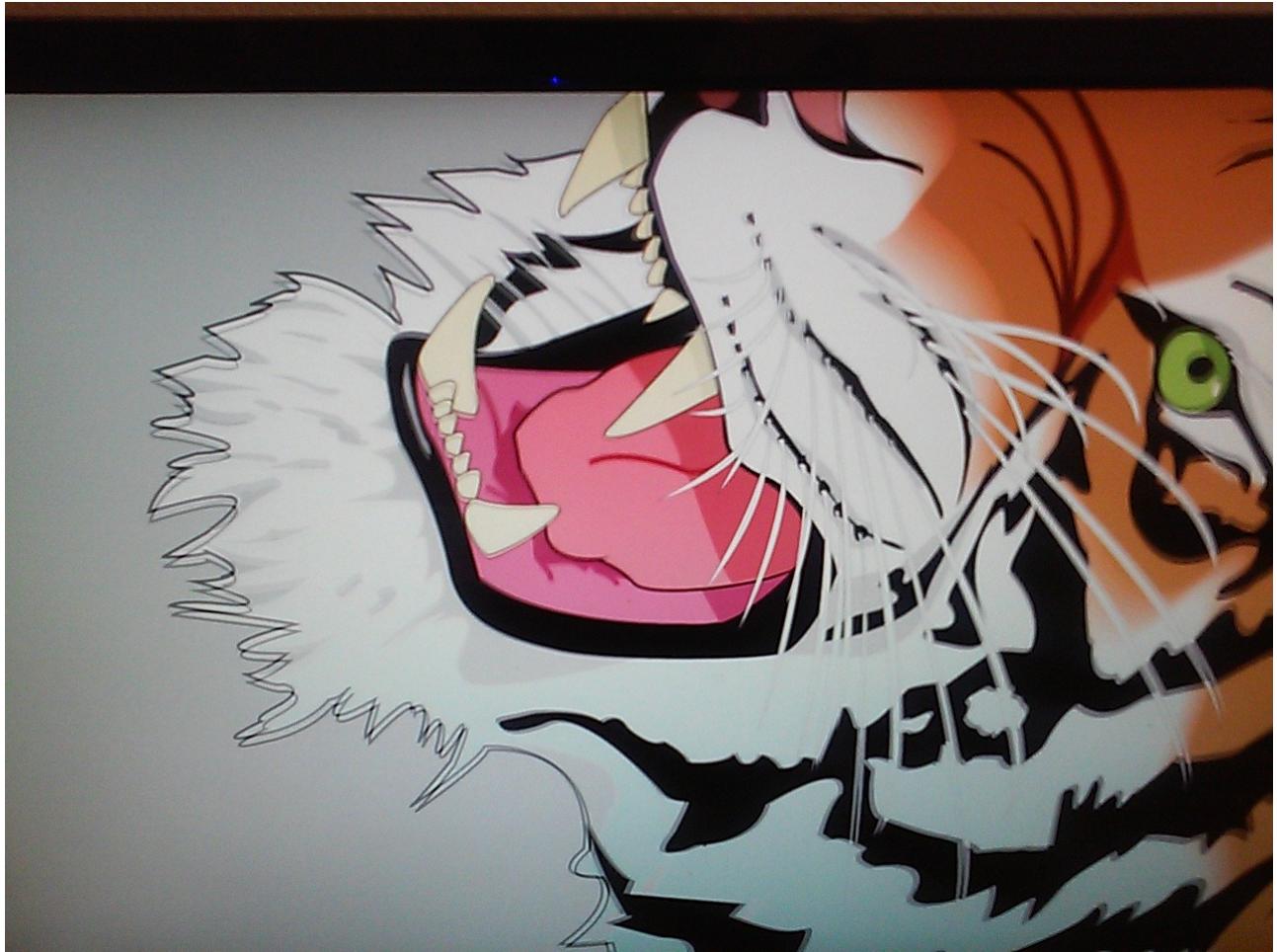


Image 2

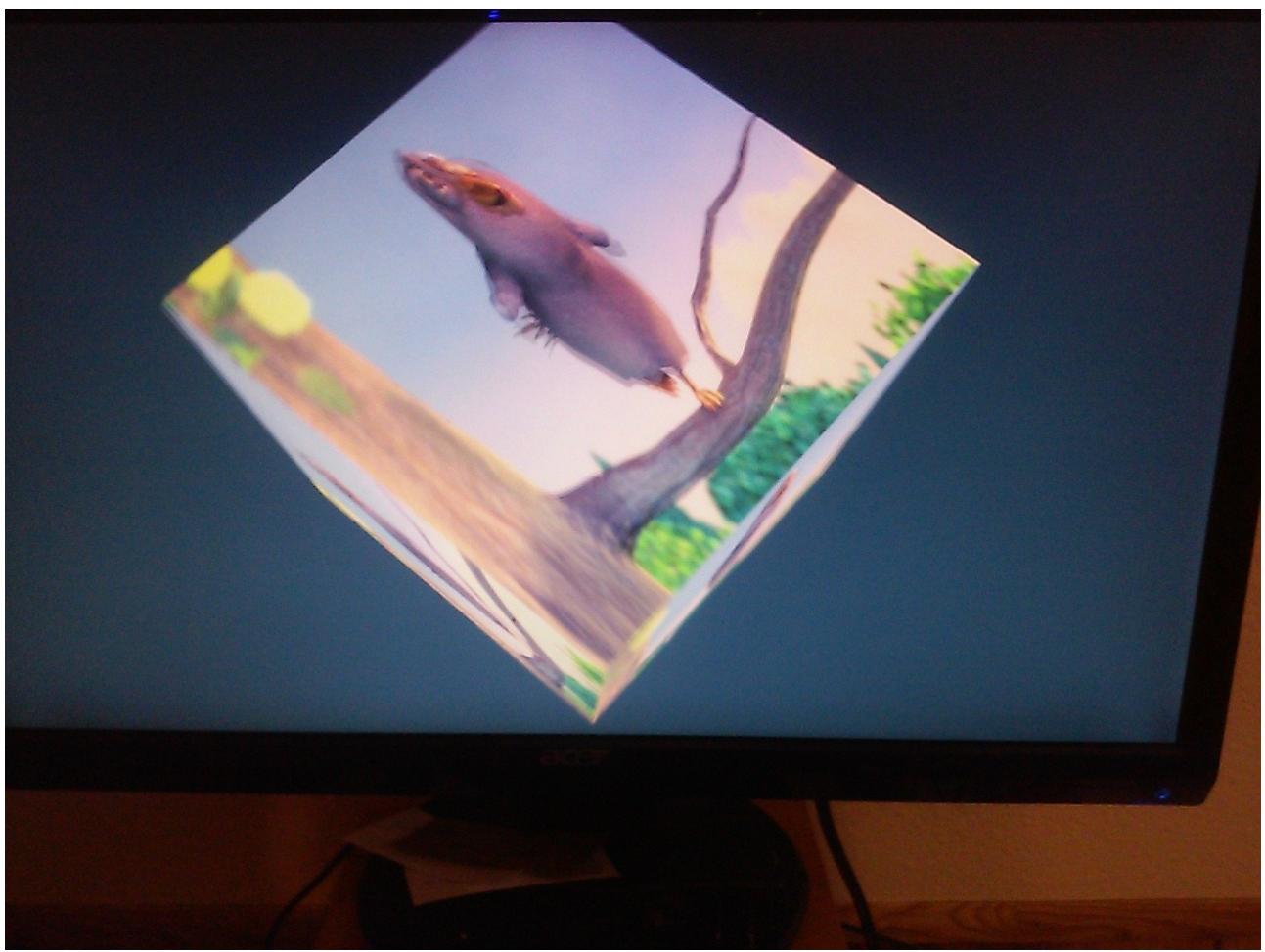


Image 3

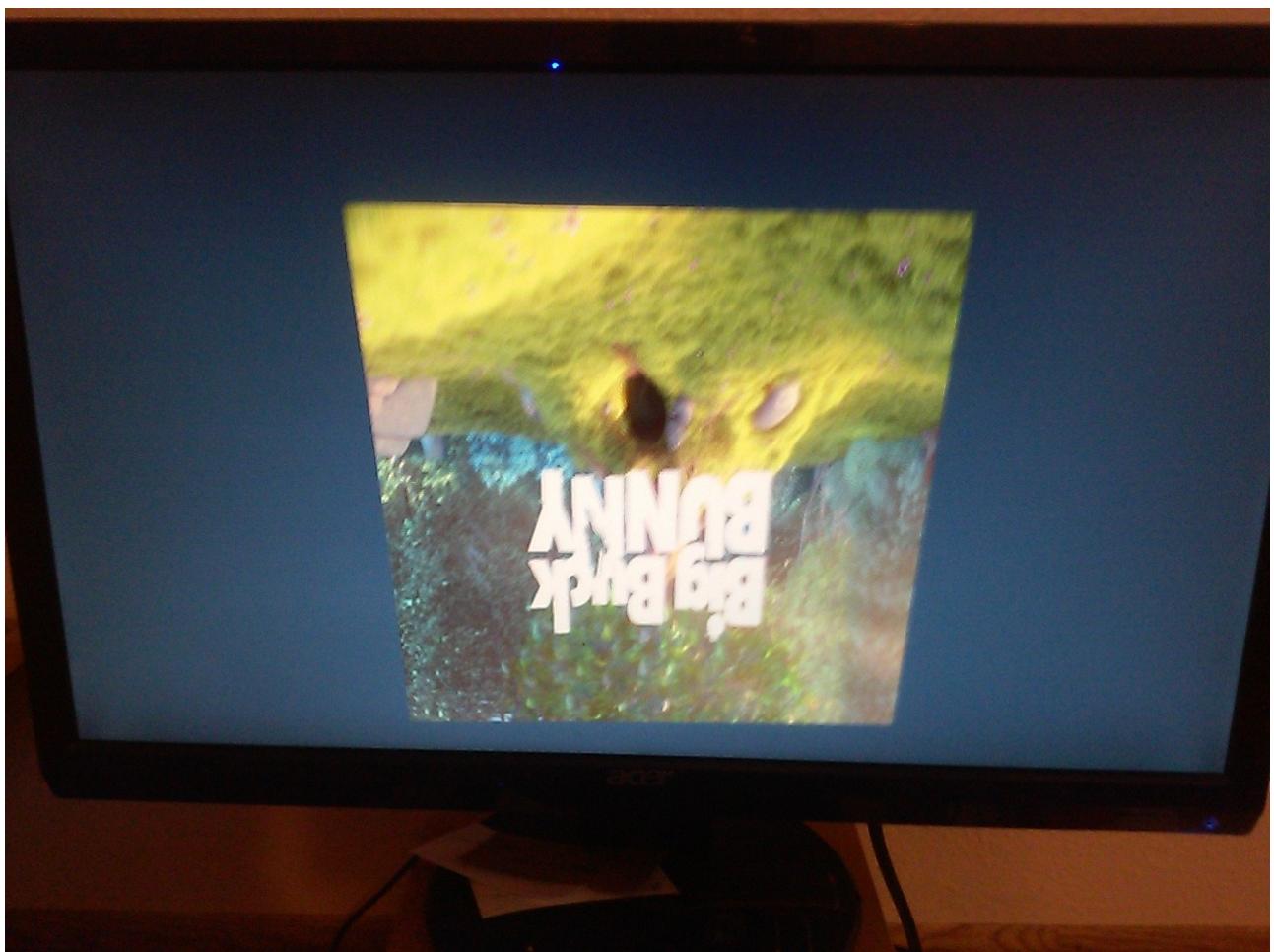


Image 4

Testing:

```
pi@raspberrypi:~ $ mkdir test_examples  
pi@raspberrypi:~ $ cd test_examples/  
pi@raspberrypi:~/test_examples $ git clone https://github.com/develone/Ultibo_PWM_TFTP.git
```

note

pi@raspberrypi:~ \$ **ultibo/core/startlazarus** was the command used to start the Lazarus IDE (Ultibo Edition).

This was required since a tar file **-rw-r--r-- 1 vidal vidal 329724698 Feb 7 17:19 ultibo.tgz** was used to install lazarus

```
pi@raspberrypi:~ $ time tar xfz ultibo.tgz
```

```
real 5m0.582s  
user 1m24.990s  
sys 0m44.640s  
ultibo  
2123 directories, 32499 files
```

From the Raspberry Pi/Programming/Lazarus IDE (Ultibo Edition)

If the PWM_TFTP_SERVO project was the current project Run/Compile

```
1 program PWM_TFTP_SERVO;  
.  
.  
{$mode objfpc}{$H+}  
.  
{ Advanced example - PWM_TFTP }  
.  
{ This example shows how to create webserver and TFTP server with remote shell }  
{ This version is for Raspberry Pi 2B and will also work on a 3B. }  
.  
{ After the first time that kernel7.img has been transferred to micro sd card }  
.  
{ tftp xx.xx.xx.xx < cmdstftp }  
.  
{ contents of cmdstftp }  
.  
{ binary }  
.  
{ put kernel7.img }  
.  
{ quit }  
.  
uses  
{InitUnit, {Include InitUnit to allow us to change the startup behaviour}  
RaspberryPi2, {Include RaspberryPi2 to make sure all standard functions are included}  
GlobalConfig,  
GlobalConst,  
GlobalTypes,  
Platform,  
Threads,  
Console,  
Classes,  
WebStatus,  
.  
.  
.  
uTFTP,  
Winsock2,  
{ needed to use ultibo-tftp }
```

1: 1 |INS |/home/pi/test_examples/Ultibo_PWM_TFTP/RPi2/PWM_TFTP_SERVO.lpr

Compile Project, OS: ultibo, Target: PWM_TFTP_SERVO: Success, Hints: 4

- ▶ PWM_TFTP_SERVO.lpr(44,2) Note: Local variable "MyPLoggingDevice" not used
- ▶ PWM_TFTP_SERVO.lpr(46,2) Note: Local variable "PWM1Device" not used
- ▶ PWM_TFTP_SERVO.lpr(49,2) Note: Local variable "Handle1" not used
- ▶ PWM_TFTP_SERVO.lpr(52,2) Note: Local variable "TCP" not used

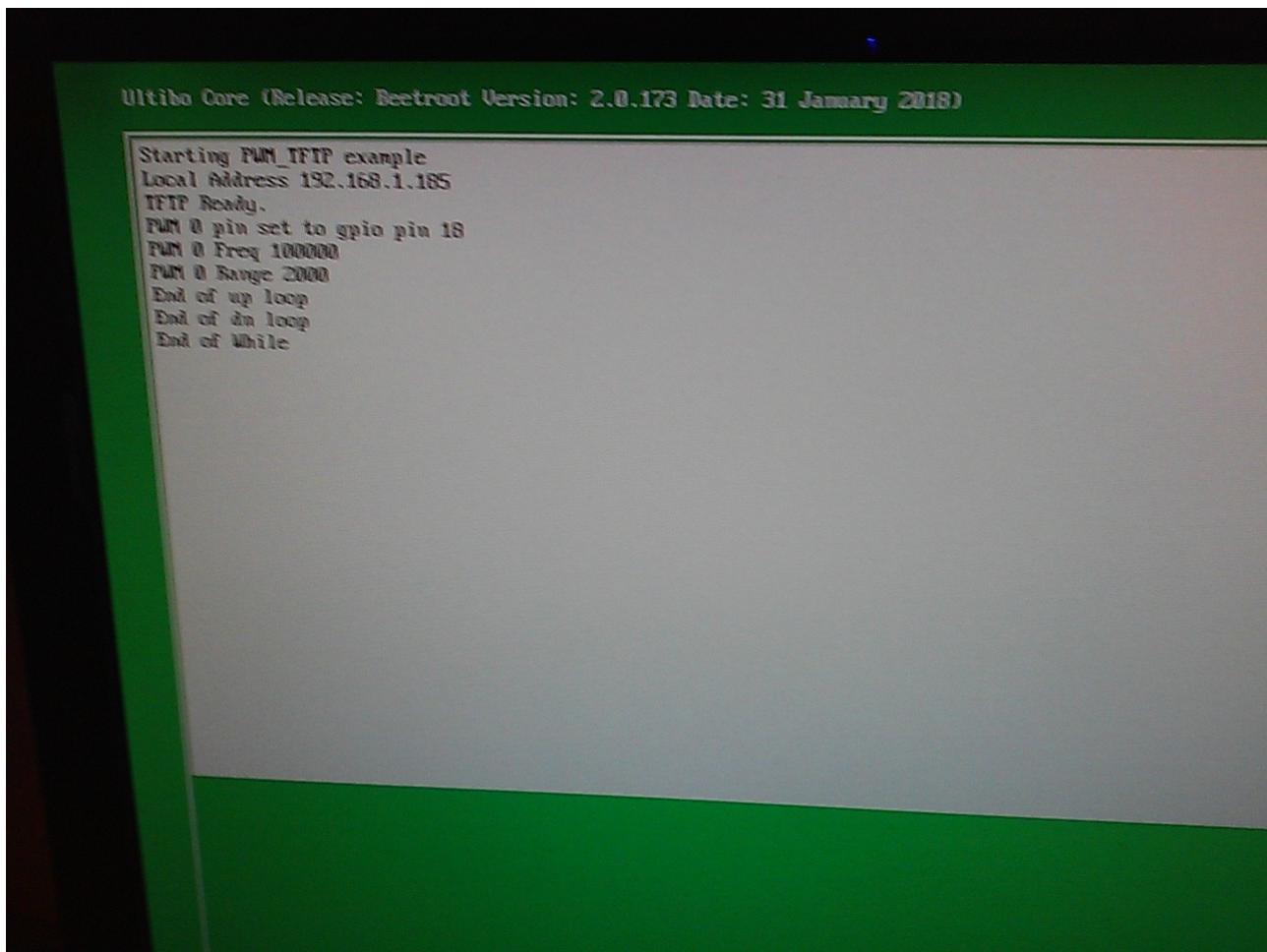
Success Compile

If the compile works without errors a green bar wil appear as seen above.

Transferring the kernel7.img and firmware files displays the following when booting.

The Core was updated

~~ULTIBO_RELEASE_DATE = '31 January 2018';~~
~~ULTIBO_RELEASE_NAME = 'Beetroot';~~
~~ULTIBO_RELEASE_VERSION = '2.0.173';~~
ULTIBO_RELEASE_DATE = '19 March 2018';
ULTIBO_RELEASE_NAME = 'Beetroot';
ULTIBO_RELEASE_VERSION = '2.0.235';
The recently built target matches the values found in ultibo/core/fpc/source/rtl/ultibo/core/globalconst.pas



pi@raspberrypi:~/test_examples/Ultibo_PWM_TFTP/RPi2 \$ cp kernel7.img /media/pi/6FCF-2BFD/

Rpi-Zero

prior to boot installed ultiboinstaller.sh and modified the dphys-swapfile

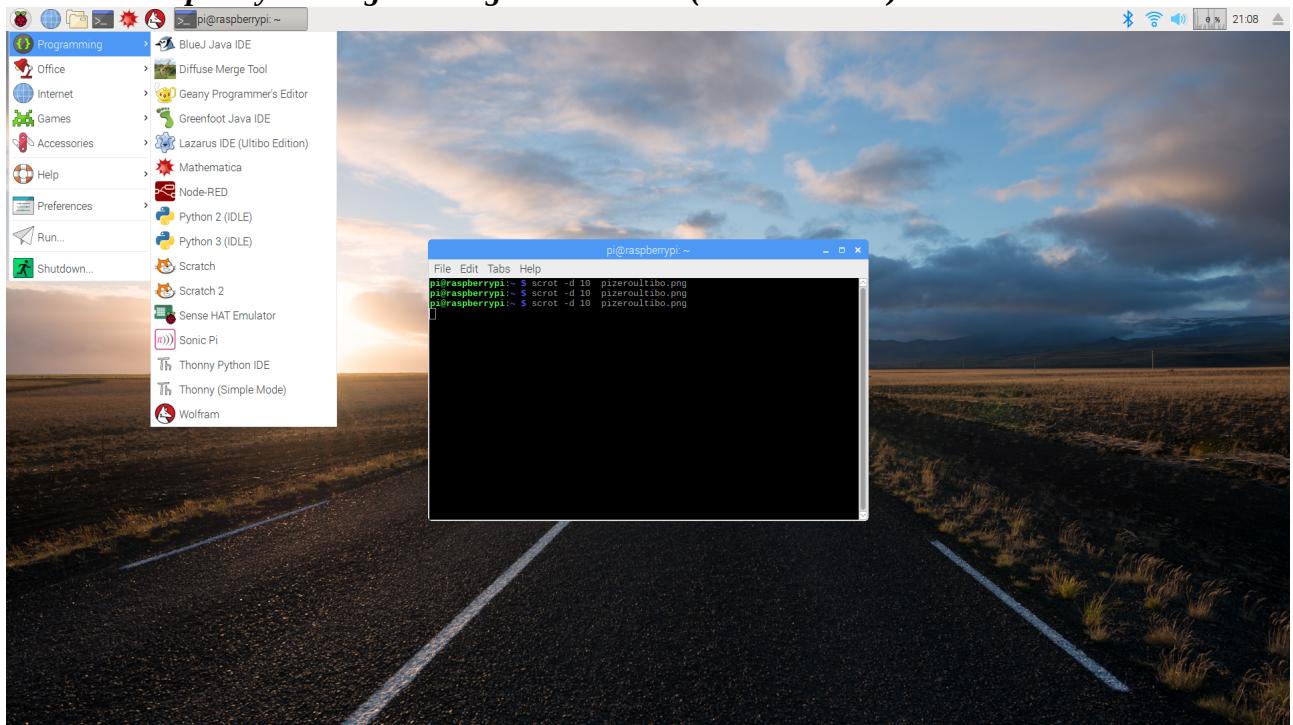
```
cd /media/vidal/rootfs/home/pi
cp /home/vidal/raspbian-pi-gen/ultiboinstaller.sh .
cd ./cd etc/
cp dphys-swapfile dphys-swapfile.orig
```

```
nano dphys-swapfile  
diff dphys-swapfile dphys-swapfile.orig  
16c16  
< CONF_SWAPSIZE=1000  
---  
> CONF_SWAPSIZE=100
```

See the first_boot document on booting the Raspbina Ultibo micro-sd-card..

Rpi-Zero Starting Lazarus

From the Raspberry Pi/Programming/Lazarus IDE (Ultibo Edition)



Starting Lazarus

```

1 program HelloTiger;
2   { (Mode objfpc)($H)
3
4   { VideoCore IV example - Hello Tiger
5   { A basic OpenVG example showing a rotating picture of a tiger, this example
6   { uses a picture embedded in the code.
7   { To compile the example select Run, Compile (or Run, Build) from the menu.
8   { Once compiled copy the kernel7.img file to an SD card along with the
9   { firmware files and use it to boot your Raspberry Pi.
10  { this version is for Raspberry Pi 2B and will also work on a 3B.
11
12  uses
13    RaspberryPi2, {Include RaspberryPi2 to make sure all standard functions are included}
14    GlobalConst,
15    GlobalTypes,
16    Threads,
17    Console,
18    Syscalls,
19    Syscalls, {Include the Syscalls unit to provide C library support}
20    Vc4; {Include the Vc4 unit to enable access to the GPU}
21
22  var
23    WindowHandle:TWindowHandle;
24
25  {Link our C library to include the original example}
26  {$linklib hello_tiger}
27
28  {Import the main function of the example so we can call it from Ultibo}
29  procedure hello_tiger; cdecl; external 'hello_tiger' name 'hello_tiger';
30
31 begin
32   {Create a console window as usual}
33
34 end.

```

Compile Project, OS: ultibo, Target: HelloTiger Success

Compiling HelloTiger on Rpi-Zero

Success Compile

If the compile works without errors a green bar wil appear as seen above.

pi@raspberrypi:~/ultibo/core/examples/VideoCoreIV>HelloPi>HelloTiger/RPi2 \$ cp kernel7.img /media/pi/6FCF-2BFD/

```

1 program HelloVideocube;
2   { (Mode objfpc)($H)
3
4   { VideoCore IV example - Hello Videocube
5   { Combining the 3D cube from hello triangle with video from the previous
6   { example, very nice demonstration.
7   { To compile the example select Run, Compile (or Run, Build) from the menu.
8   { Once compiled copy the kernel7.img file to an SD card along with the
9   { firmware files and use it to boot your Raspberry Pi.
10  { Make sure you also copy the test.h264 file from the Media folder.
11  { You also MUST create a config.txt file in the root directory of your SD card
12  { with at least the following setting:
13    { gpu_mem=128
14
15  { this version is for Raspberry PI 2B and will also work on a 3B.
16
17  uses
18    RaspberryPi2, {Include RaspberryPi2 to make sure all standard functions are included}
19    GlobalConst,
20    GlobalTypes,
21    Threads,
22    Console,
23    Sysutils,
24    Syscalls, {Include the Syscalls unit to provide C library support}
25    Vc4; {Include the Vc4 unit to enable access to the GPU}
26
27  var
28    WindowHandle:TWindowHandle;
29
30 end.

```

Compile Project, OS: ultibo, Target: HelloVideocube Success

Compiling HelloVideocube on Rpi-Zero

Success Compile

If the compile works without errors a green bar wil appear as seen above.

```
pi@raspberrypi:~/ultibo/core/examples/VideoCoreIV>HelloPi>HelloVideocube/RPi2 $ cp  
kernel7.img /media/pi/6FCF-2BFD/  
pi@raspberrypi:~/ultibo/core/examples/VideoCoreIV>HelloPi>Media $ cp test.h264  
/media/pi/6FCF-2BFD/
```

Both of examples created on the Rpi Zero ran correctly with on a few files running Ultibo.

Note:

Even on the RPI Zero the compile was very quick for the HelloTiger & HelloVideocube projects.

Appendix A: Notes during pi-gen testing

pi-gen Tool used to create the raspberrypi.org Raspbian images.

Forked pi-gen <https://github.com/RPi-Distro/pi-gen>
commit 066eb03d52868290661e813738a9a66eda263aa9
Author: Ben Pirt <ben@pirt.co.uk>
Date: Wed Jan 3 12:48:31 2018 +0000

Allow image building to be skipped for stages (#137)

pi-gen
GitHub - RPi-Distro/pi-gen: Tool used to create the raspberrypi.org ...
<https://github.com/RPi-Distro/pi-gen>

README.md. pi-gen. Tool used to create the raspberrypi.org Raspbian images.
Dependencies. pi-gen runs on Debian based operating systems. Currently it
is only supported on either Debian Stretch or Ubuntu Xenial and is known
to have issues building on earlier releases of these systems. To install
the required

On a Ubuntu 16.04 added pi-gen dependencies.

sudo apt-get install quilt parted realpath qemu-user-static debootstrap zerofree pxz zip dosfstools
bsdtar libcap2-bin grep rsync xz-utils

Linux sim2 4.4.0-112-generic #135-Ubuntu SMP Fri Jan 19 11:48:36 UTC 2018 x86_64 x86_64
x86_64 GNU/Linux
AMD FX(tm)-4130 Quad-Core Processor
cpu MHz : 1800.000

Steps create the pi-gen/work/2018-02-07-RaspbianUltibo tree.

- 1.0)
git clone <https://github.com/develone/pi-gen.git>
- 2.0)
cd pi-gen/
- 3.0)
cp ~/raspbian-pi-gen/config .
Added a file pi-gen/config

```
#!/bin/bash
export IMG_NAME=RaspbianUltibo
4.0)Appendix
cp ~/raspbian-pi-gen/stage4/00-install-packages/02-packages stage4/00-install-packages/
5.0)
cp ~/raspbian-pi-gen/ultibo_laz_fpc/ultibo94af3cb.tgz ~/pi-gen/stage4/02-extras/files
cp ~/raspbian-pi-gen/ultibo_laz_fpc/ultibo2fc3634.tgz ~/pi-gen/stage4/02-extras/files
```

**Note: The file raspbian-pi-gen/ultibo_laz_fpc/ultibo2fc3634.tgz is not
in the rasbian-pi-gen repository due to the size of the tar file.**

331297115 Feb 14 13:32 ultibo2fc3634.tgz

md5sum

e3b54b45689a14e67440f4ea428efc5a_ultibo94af3cb.tgz

a2b796d36aee271655e80ea088214ea2_ultibo2fc3634.tgz

**Adding the script in stage4/02-extras that un tar the ultibo tree which provides
lazarus. Uses the file ultibo2fc3634.tgz**

Adding the short cut to start Lazarus

6.0.)

```
cp ~/raspbian-pi-gen/ultibo.desktop stage4/02-extras/files/
```

7.0)

```
cp ~/raspbian-pi-gen/stage4/02-extras/01-run.sh ~/pi-gen/stage4/02-extras/
```

new for ver 03/20/18 the file /etc/dphys-swapfile

<CONF_SWAPSIZE=100

>CONF_SWAPSIZE=1000

8.0)

```
date > t1.txt; ./build.sh ; date >> t1.txt
```

Wed Feb 21 08:14:23 MST 2018

Wed Feb 21 11:30:34 MST 2018

Tue Mar 20 17:47:38 MDT 2018

Tue Mar 20 20:49:01 MDT 2018

Note: The 3 hours and 16 min could be reduced with the use of local repository.

du -s local-apt-repo/

1513164 local-apt-repo/

The download of required packages takes considerable time.

9.0)

```
cd deploy/
```

10.0)

```
fdisk /dev/sdb
```

11.)

```
date > t1.txt; gzip -dc image_2018-03-20-RaspbianUltibo.zip | dd bs=16M of=/dev/sdb; date >> t1.txt
```

0+165834 records in

0+165834 records out

7730102272 bytes (7.7 GB, 7.2 GiB) copied, 696.3 s, 11.1 MB/s

Wed Feb 21 11:40:20 MST 2018

Wed Feb 21 11:56:51 MST 2018

The stretch build has 1313 pkgs installed

~~The 2018-02-21-RaspbianUltibo build has 1427 pkgs installed~~

The 2018-03-20-RaspbianUltibo build has 1434 pkgs installed

ssh -Y pi@192.168.1.185

```
uname -a
Linux raspberrypi 4.9.59-v7+ #1047 SMP Sun Oct 29 12:19:23 GMT 2017 armv7l GNU/Linux
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/root	30596716	7078208	22182604	25%	/
devtmpfs	443792	0	443792	0%	/dev
tmpfs	448400	0	448400	0%	/dev/shm
tmpfs	448400	17236	431164	4%	/run
tmpfs	5120	4	5116	1%	/run/lock
tmpfs	448400	0	448400	0%	/sys/fs/cgroup
/dev/mmcblk0p1	41868	21328	20540	51%	/boot
tmpfs	89680	0	89680	0%	/run/user/1000

Testing new build with the dependencies need to install the Lazarus/FPC

Added a file pi-gen/stage4/00-install-packages/02-packages

```
libgtk2.0-dev
libcairo2-dev
libpango1.0-dev
libgdk-pixbuf2.0-dev
libatk1.0-dev
libghc-x11-dev
binutils-arm-none-eabi
gcc-arm-none-eabi
diffuse
gitk
tcl-dev
telnet
tftp
hexedit
cmake
flex
bison
```

./build.sh

Creates the pi-gen/work/2018-03-20-RaspbianUltibo tree.

```
vidal@sim2:~$ ls pi-gen/work/2018-03-20-
RaspbianUltibo/stage0/rootfs/var/cache/apt/archives/*.deb | wc
   220  220 15327
```

```
vidal@sim2:~$ ls pi-gen/work/2018-03-20-
RaspbianUltibo/stage1/rootfs/var/cache/apt/archives/*.deb | wc
    15   15 1613
```

```
vidal@sim2:~$ ls pi-gen/work/2018-03-20-
RaspbianUltibo/stage2/rootfs/var/cache/apt/archives/*.deb | wc
   214  214 15221
```

```
vidal@sim2:~$ ls pi-gen/work/2018-03-20-
RaspbianUltibo/stage3/rootfs/var/cache/apt/archives/*.deb | wc
   427  427 30630
```

```
vidal@sim2:~$ ls pi-gen/work/2018-03-20-
RaspbianUltibo/stage4/rootfs/var/cache/apt/archives/*.deb | wc
   536  536 38434
```

```
vidal@sim2:~$ ls pi-gen/work/2018-03-20-
RaspbianUltibo/stage5/rootfs/var/cache/apt/archives/*.deb | wc
```

9 9 664
1427 packages are required