# Guide to PhidgetSBC and OpenCV

Written by Christopher McGirr, Winter 2014

## 1 Installing OpenCV on Linux

First, before we start it is useful to install OpenCV on your personal computer for testing the image algorithms. The phidgetSBC board already has a older version of OpenCV which is not an issue as we will be using OpenCV primarily for Image Capturing. The robot is running OpenCV 2.4.2 so it may be a good idea to install this version on your personal computer as well to avoid differences between libraries.
Now this guide will look at installing OpenCV for Linux Systems.
You can also simply follow this link. It has a script for installing OpenCV on Ubuntu. If not just keeping following this guide.

`https://help.ubuntu.com/community/OpenCV`

First you will have to make sure that the following programs are installed. The guide for installation can be found at the following link.

`http://docs.opencv.org/doc/tutorials/introduction/linux_install/linux_install.html`

Or try this site for a comprehensive guide to installing.

`http://www.ozbotz.org/opencv-installation/`

Start with the following command to update the system.

```
sudo apt-get install build-essential
```

If this does not install everything you may need to install it manually. You can use the following program to see if the program is within you repository, but you will still need to install it, if it is not already.

```
apt-cache search program
sudo apt-get install program
```

Program Prerequisites

1. CMake 2.6 or Higher
2. Git
3. GTK+2.x or higher, including headers (libgtk2.0-dev)
4. pkgconfig
5. Python 2.6 or later and Numpy 1.5 or later with developer packages (python-dev, python-numpy)
6. ffmpeg or libav development packages: libavcodec-dev, libavformat-dev, libswscale-dev;
7. (optional) libdc1394 2.x
8. (optional) libjpeg-dev, libpng-dev, libtiff-dev, libjasper-dev.

Now we will be using CMake to install OpenCV. First we must download the appropriate source tar ball. Follow the link below to download.

`http://sourceforge.net/projects/opencvlibrary/`

Now head to the directory where you downloaded the source and extract the contents.

```
tar xvf opencv-2.4.8.tar.gz
```

Then enter the folder and make a build directory for OpenCV.

```
cd ~/OpenCV-2.4.8
mkdir release
cd release
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..
```

Then enter the release folder and run.

```
make
sudo make install
```

If this guide is not complete or not useful. Feel free to search online for tutorials for how to install OpenCV. There are plenty out there that may explain this in more detail if it does not work for you.

`http://miloq.blogspot.ca/2012/12/install-opencv-ubuntu-linux.html`

# 2 Installing OpenCV on Windows

First you will need to download the appropriate version of OpenCV for our code to run. We will be using Version 2.4.2 of OpenCV. This link provides where you can download it.

`http://sourceforge.net/projects/opencvlibrary/files/opencv-win/`

If you are have troubles installing please check out these links. They helped me to install OpenCV on my Windows 8 PC.

`https://www.youtube.com/watch?v=kVyVEHK4qfQ`
`http://kevinhughes.ca/tutorials/opencv-install-on-windows-with-codeblocks-and-mingw/`
`http://wiki.wxwidgets.org/Adding_an_Environment_Variable_under_Windows`
`http://stackoverflow.com/questions/10860352/getting-started-with-opencv-2-4-and-mingw-on-windows-7`
`http://docs.opencv.org/doc/tutorials/introduction/windows_visual_studio_Opencv/windows_visual_studio_Opencv.html#windows-visual-studio-how-to`

Now take the *.exe* you have just downloaded from Sourceforge and run it. Extract the contents to the root of your main disk, i.e. the C drive *C:*. Once this is done you will most likely need to install a C/C++ Compiler for you system. I used an open sourced compiler for windows called MinGW. Go to this link and download the GUI installer.

`http://sourceforge.net/projects/mingw/files/latest/download`

Run the MinGW GUI installer and make sure to install the GCC and G++ compilers. If asked make sure to install them in *C:*, it makes life simpler.
Now you should have two folders in the root of your main drive called *MinGW* and *OpenCV*. We will have to add the MinGW to the Path Envirnoment on windows. So open up your File Explorer and right-click on your PC, in Windows 8 its the icon that says *This PC* and hit properties. You should now see the specs of you computer and options on the left side. Click on Advanced System Settings and then click Environment Variables. Now another window should open up and you should see two fields, the User Variables and the System Variables. Head to system Variables and find the Variable named *PATH*. Edit *PATH* and add the location of MinGW. Simply add a semicolon to the end of the line and add the path to the bin folder of MinGW. In my case it is,

`C:\MinGW\bin`

If you don't get what I mean or how to do this on your version of windows, there are plenty of guides online on how to add locations to your System PATH. You can test to see if MinGW is correctly installed

by opening up a Command Prompt and entering *gcc*. It should return an error saying there is no input file meaning you have correctly installed it. If this does not happen, something is wrong with the PATH definition and windows can't find the *.exe* for gcc. Great, now we need to install CMake for windows so that we can compile OpenCV for you computer. So first head to this link and install CMake.

```
http://www.cmake.org/cmake/resources/software.html
```

Once you've installed CMake, open the GUI and enter the source code directory for OpenCV. This should be *C:\OpenCV* if you've extracted OpenCV2.4.2 to the main C drive. Now depending on you version of windows, 32bit or 64bit, the built binaries should be in the appropiate folders. In my case for my 64bit machine I told CMake to store the binaries in,

```
C:/OpenCV/build/x64/mingw
or for 32bit
C:/OpenCV/build/x86/mingw
```

Now once that is done, hit the *Configure* button and wait CMake to return with build options. Once that is done you can hit the *Generate* button which will generate the necessary files for install. Make sure to use the MinGW generator if asked.
Once that is done head to *C:/OpenCV/build/x64/mingw* in the file explorer, hit *Shift+Right-Click* and open a command prompt. Then run the following command.

```
mingw32-make
```

Once that is done, which can take a while run this command from the same prompt.

```
mingw32-make install
```

Hopefully, this will install everything correctly and no errors will be thrown during installation. The necessary built files, which we will be using, will be under the new location of

```
C:\OpenCV\build\x64\mingw\install\
```

The final step is to add the OpenCV libraries to the PATH environment on Windows. Simply follow the same steps as before for MinGW and add the location to the end of the line. In my case the location was,

```
C:\OpenCV\build\x64\mingw\install\bin
```

This should complete the installation of OpenCV. If you have any troubles this link, mentioned earlier explains clearly and with pictures how to go about what I just explained.

```
http://kevinhughes.ca/tutorials/opencv-install-on-windows-with-codeblocks-and-mingw/
```

To run one of our sample code and compile it simply head to the folder where our code is contained. It should have a file named *obdetect.c*.

```
...\C Code\PC Code\Optimized
```

And run the following command to compile it.

```
g++ -o obdetect obdetect.c -I"C:\OpenCV\build\x64\mingw\install\include"
 -L"C:\OpenCV\build\x64\mingw\install\lib" -llibopencv_core242
 -llibopencv_imgproc242 -llibopencv_highgui242
```

The commands *-I and -L* tell the compiler where the library files are located and the *-l* command tells the compiler which libraries to include. This command should return without error and now you can test the program by running *obdetect* from the command line.

# 3 Installing BlueZ for Linux

In order to interface with the robot over wireless you will need to have a Bluetooth interface as well as the Bluetooth stack protocol so that we can connect to the robot. You can simply install the programs with the following command.

```
sudo apt-get install bluez, bluez-lib, bluez-utils, bluez-sdp, bluez-fw, bluez-hcidump
```

Or alternatively you can download the source code and install using CMake. Below is a link to a tutorial that may help if you are having problems.

```
http://www.iub.edu.pk/Documents/Mamoona/BLUEZ_Installation_Configurations_on_Linux.pdf
```

You can use the following command to see if the bluetooth library is installed on your computer.

```
ldconfig -v | grep lbluetooth
or
ldconfig -v | grep bluetooth
```

# 4 Uploading Code to the Robot

Now the simplest way to connect to the robot is to directly connect to the robot over Ethernet. Simply take an network cable and plug the robot directly into your PC. Hopefully, the PC will automatically detect there is a valid connection on the Ethernet port. If not you may have to change some connection settings for your Ethernet port.

The PhidgetSBC has a webserver running on the robot. So once a valid connection is established you can login to the robot in a web browser. Simply enter this link.

```
phidgetsbc.local
```

You will be prompted to login. The login information is the following

```
Username : admin
Password : admin
```

The on board web server has many options and information about the robot. The main feature is the Projects tab which contains all the past projects on the robot. It's an easy way for uploading your code onto the robot.

Now if you want to compile code and run commands on the robot you can SSH into the robot. Simply use a program like *Putty* and enter the robot using the same address, *phidgetsbc.local*. When asked the login information is,

```
user: root
password : admin
```

Once you have logged in over SSH you can check out the directory where all the programs are store on the robot.

```
cd \usr\userapps
ls
```

The *cd* command changes directory to where the codes are stored. The command *ls* lists all the folders and files within the *userapps* directory. The motion detection code is stored in the following location.

```
cd \usr\userapps\image
```

# 5 Compiling the code

Now to compile code that contains an OpenCV library or Bluetooth Library we can use the following command on the PC if your running Linux (Ubuntu).

```
g++ -o objectdetection objectdetection.c -lbluetooth ‘pkg-config --libs opencv --cflags opencv‘
```

Now for compiling on the robot we use the same command except if we are including motor control. So we have to call the phidget library when compiling.

```
g++ -o objectdetection objectdetection.c -lbluetooth -lphidget21 ‘pkg-config --libs opencv --cflags ope
```

# 6 Description of Image Code

Note some of this code requires that you have a bluetooth interface so that the program can communicate with the robot.

Code listed in the directory *\usr\userapps\image*

1. image.c[1] = Bluetooth image sending code that sends a black and white image of size 640*480
2. image2.c[2] = Bluetooth image sending code that sends a black and white image of size 640*240
3. image3.c[3] = Bluetooth image sending code that sends a black and white image of size 640*120
4. objectdetection.c = main motion detection algorithm for the robot. It needs debugRobot.c to be running on a PC with bluetooth. the program waits for a connection from a PC.

PC code listed under *\C Code\PC Code* and *\C Code\Final Code*

1. debugRobot.c = debug code for the robot. receives info from the robot such as the image differences and where the robot thinks motion is occurring.
2. obdetect.c & ”object detection.c” = PC versions of the motion algorithm. optimized version and none optimized version respectively.

[1]You will need to run the companion code on a PC called Image.c to receive
[2]You will need to run the companion code on a PC called Image2.c to receive
[3]You will need to run the companion code on a PC called Image3.c to receive