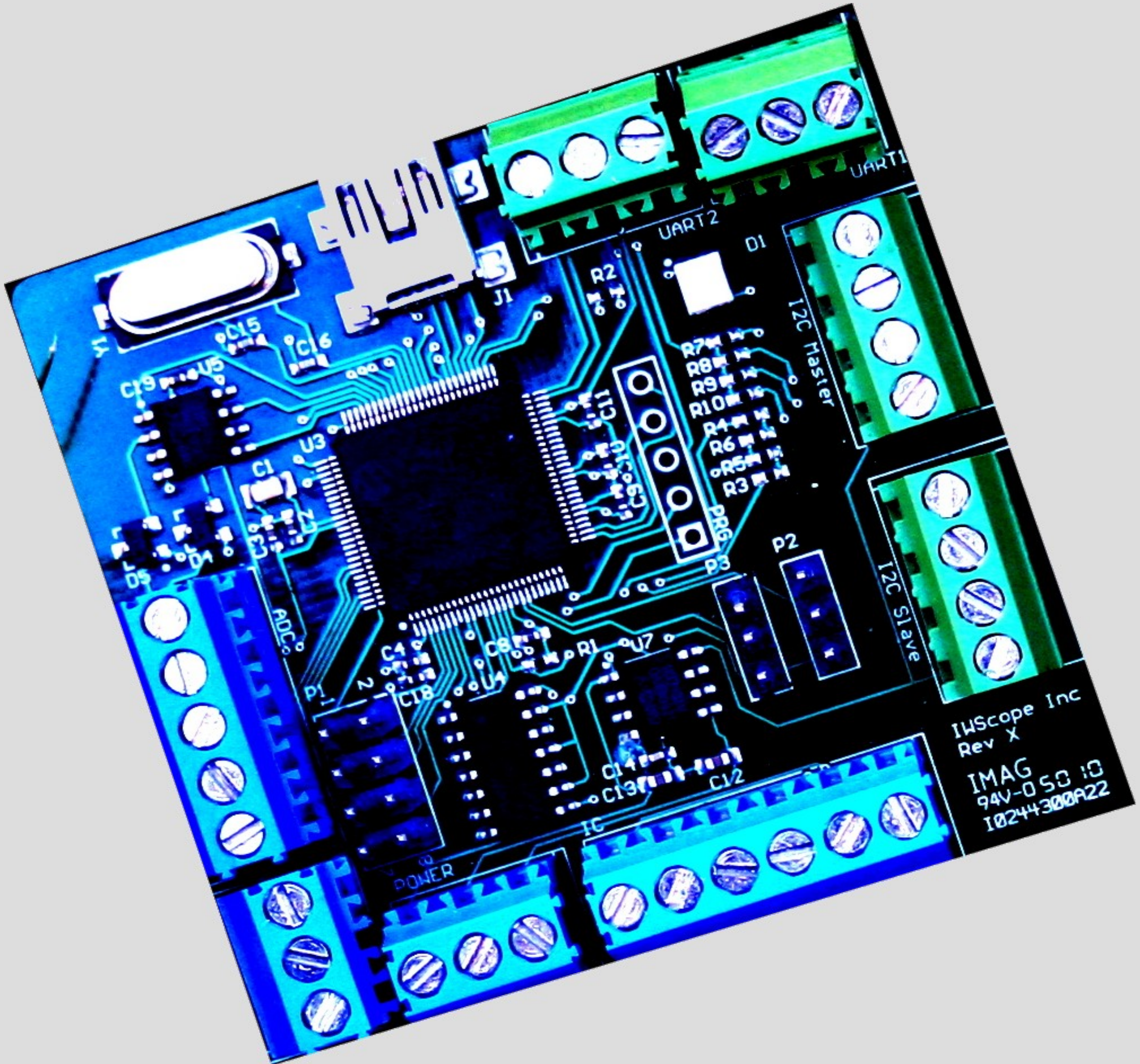
OScope V01
General Purpose Script Controlled Oscilloscope



Vijay Singh

Date Oct 25th 2015

Notice

Iwscope does not authorize any Iwscope product for use in life support systems and/or devices without the written approval of the Iwscope Corporation. Life support devices/systems are devices or systems which, a) are intended for surgical implantation into the body, or b) support or sustain life and whose failure to perform can be reasonably expected to result in injury. Iwscope products are not designed with the components required, and are not subject to the testing required to ensure a level of reliability suitable for the treatment and diagnosis of people.

Oscope Plugin Feature

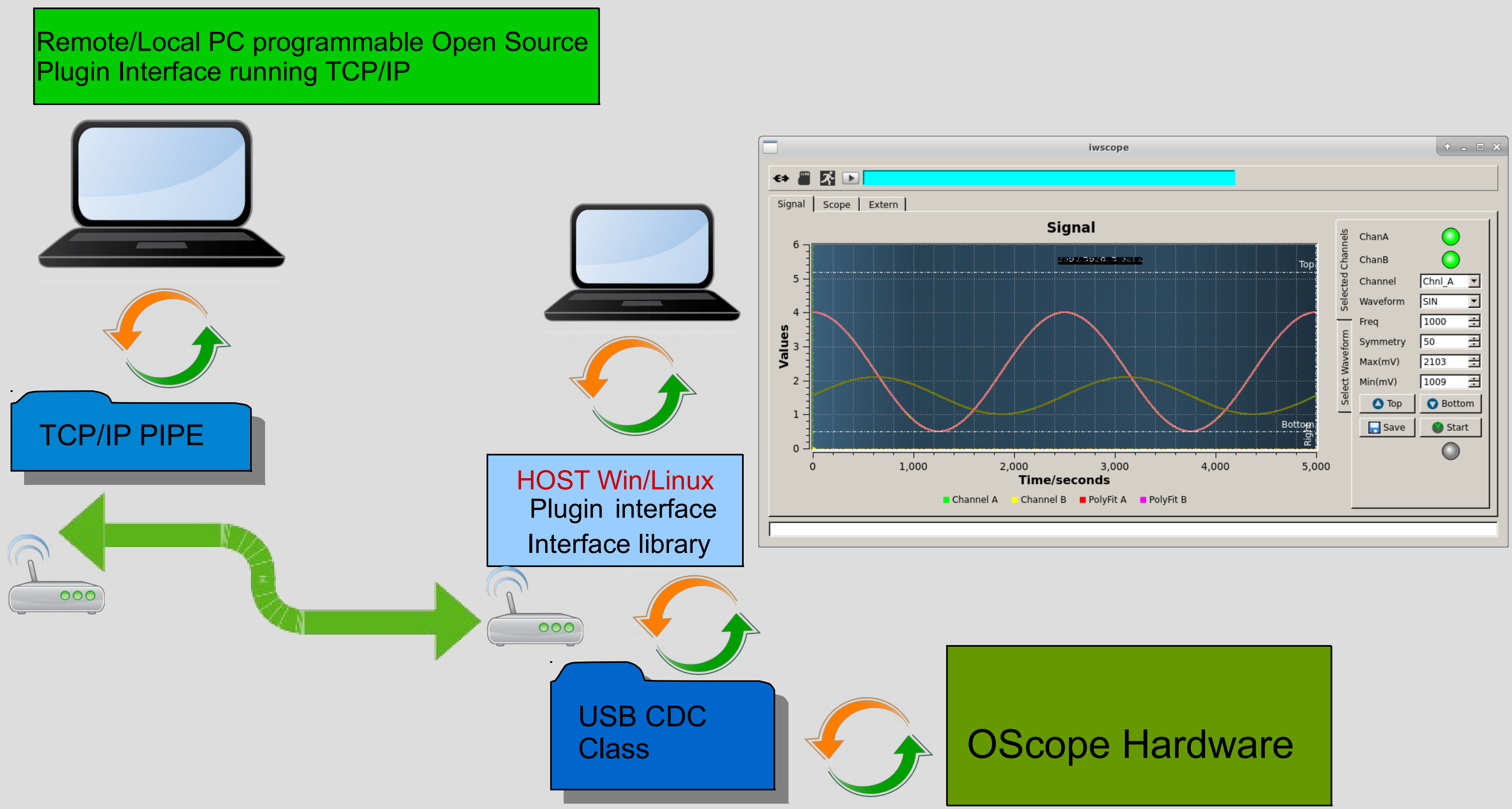
- Simple TCP/IP pipe based plugin interface which is written in “C”, plugin may operate on same host or it can be remote computer. All features of plugin interface, sample code has been provided.
- As plugin is simple client socket code which can be replaced with Python/C#/Labview API as needed.(At some point support for adding python2.x/3.0 and C# with example will be provided)
- Multiplatform (ARM,Linux,x86,Windows). Raspberry PI/Beagle bone Black/FreeScale/Windows/Intel Ubuntu and XP/7/8.
- Plugin may control all part of GUI with interface including message display/graph/signal with lot of feedback, which makes it useful for production floor calibration utility.
- Field firmware upgradeable without need of programmer.
- Flexible framework and powerful programmable intuitive and simple interface and best of all it works on most of operating systems

Use cases

- Handy tool which can perform multifunction dual channel signal generator or oscilloscope device, can be quickly customized for close loop system testing need.
- Up to 1 MHz when use other 10 bit ADC (Internal PIC32).
- Easily integrable with python script for doing complex task.
- Field firmware upgradeable without ICD/JTAG programmer device.
- Flexible framework and powerful programmable intuitive and simple interface and best of all it works on most of operating systems.
- Plugin feature enables user to communicate with array of devices.

OScope Topology

(Remote PC and Local PC may run on same host)



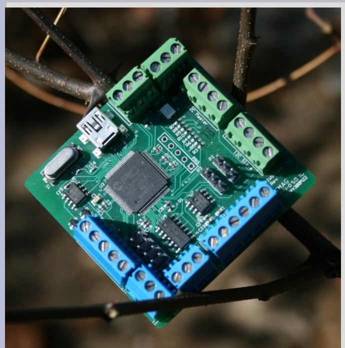
Features

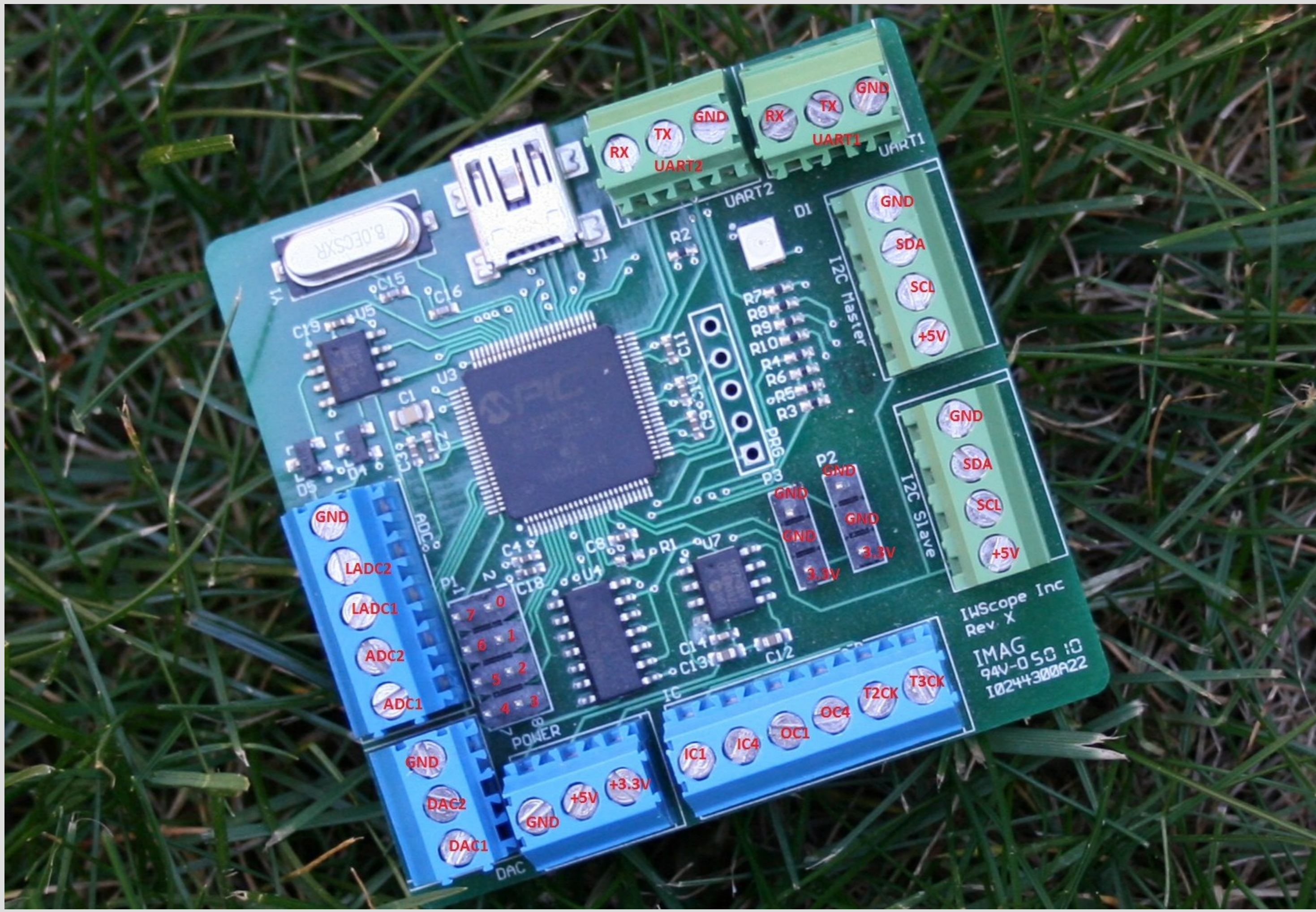
- Dual channel independent multi function generator
- Dual channel Oscilloscope may operates parallel to signal generator
- Twin two I2C Master which (7bit addressing/400KHz Clock).
- Read/Write 21 GPIO from plugin interface.
- Single channel 32bit PWM input capture for measuring pulse width modulation very accurately.
- Single channel output capture which can be controlled by plugin interface.
- Dual ADC/DAC can capture and generate continuous signal.
- Flexible communication library functions are well documented along with GUI/console sample code provided.
- Dual channel TTL UART may receive and transmit data up to 460800 baud.

Features

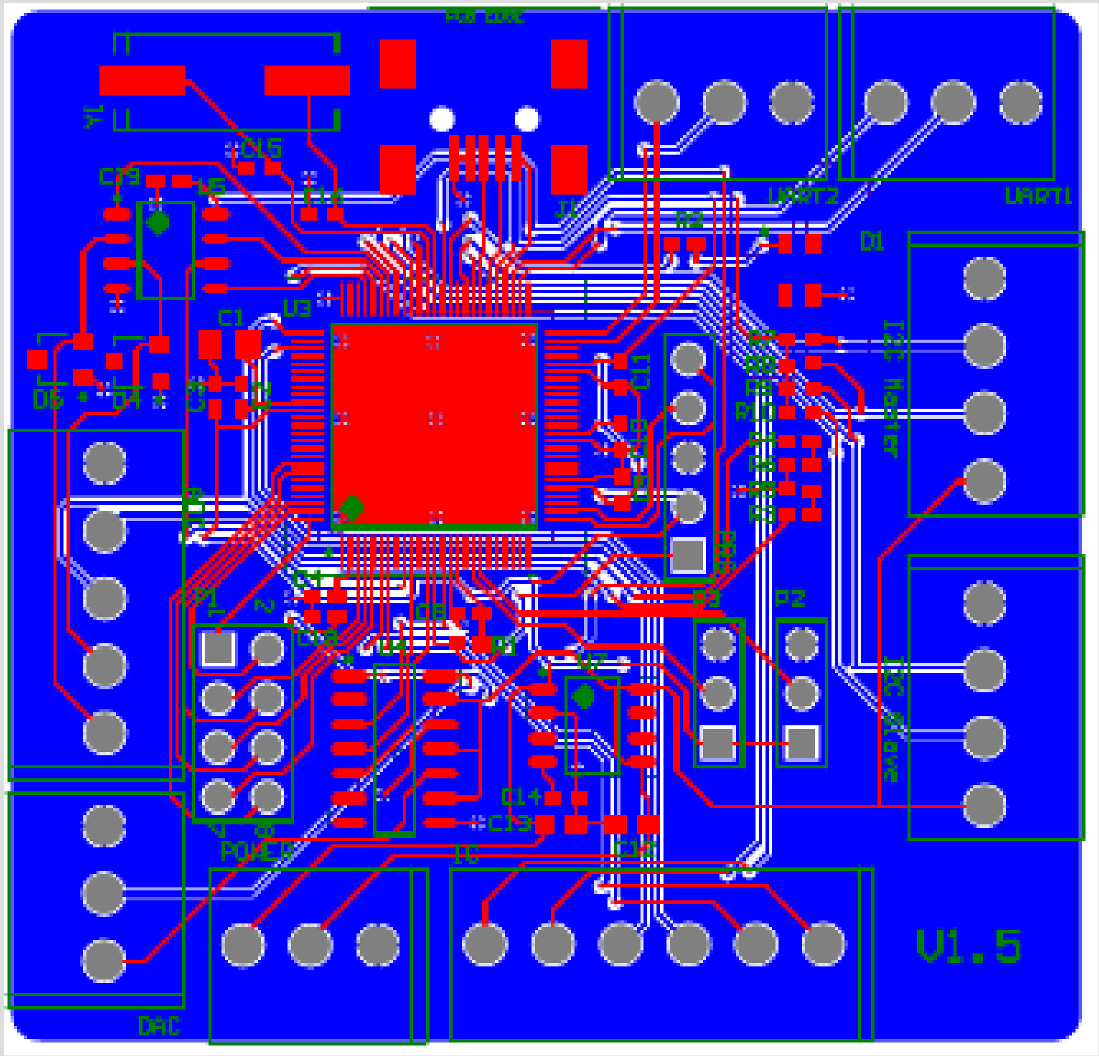
1. Oscilloscope (Dual Channel 12 bit 5V Max, Max 100 Khz)
2. Signal Generator(Dual Channel 12 bit 5V Max,Max 100 Khz)
3. I2C Master/Slave (5V, Max 400KHz Clock) Sample app included.
4. Input capture
5. Single Channel PWM Generation.
6. channel digital 3.3V IO
7. Output capture for creating PWM
8. TTL 3.3V UART (5V Tolerance,upto 460800baud)
9. Direct run time import signal from csv file via plugin/direct interface.

PIC32MX795F512L





PCB Layout



UART2
UART1
J1
Y1
C15
C19
C18
U5
U3
C1
D1
R2
D6
D4
C11
I2C Master
P1
P2
P3
C14
C13
POWER
DAC
I2C Slave
V1.5

Possible Use Cases

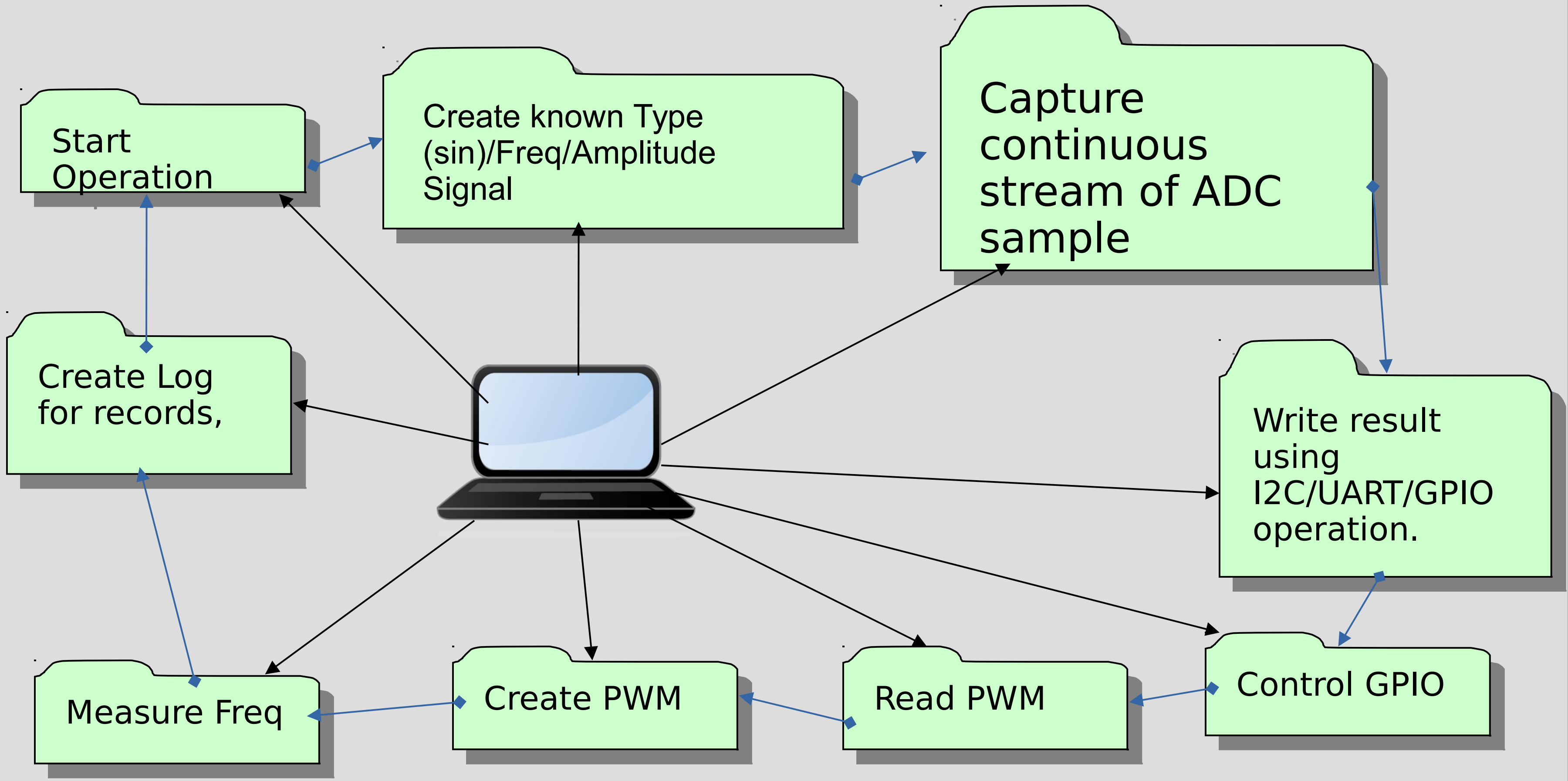
Powerful tool when implementing iterated real complex automated long sequence of calibration which requires read / write/ i2c/ uart/ gpio/ pwm where interaction between set of operation is heavily interdependent on previous or next.

For example user A who is calibrating a magnetic measurement in feedback loop. Calibration of such a system will require feeding known calibrated signal and measuring it system output in multiple steps and while varying temperature and other parameter, Calibrating such a system manually by using traditional signal generator and while controlling temperature is very tedious task with possibility of human error,

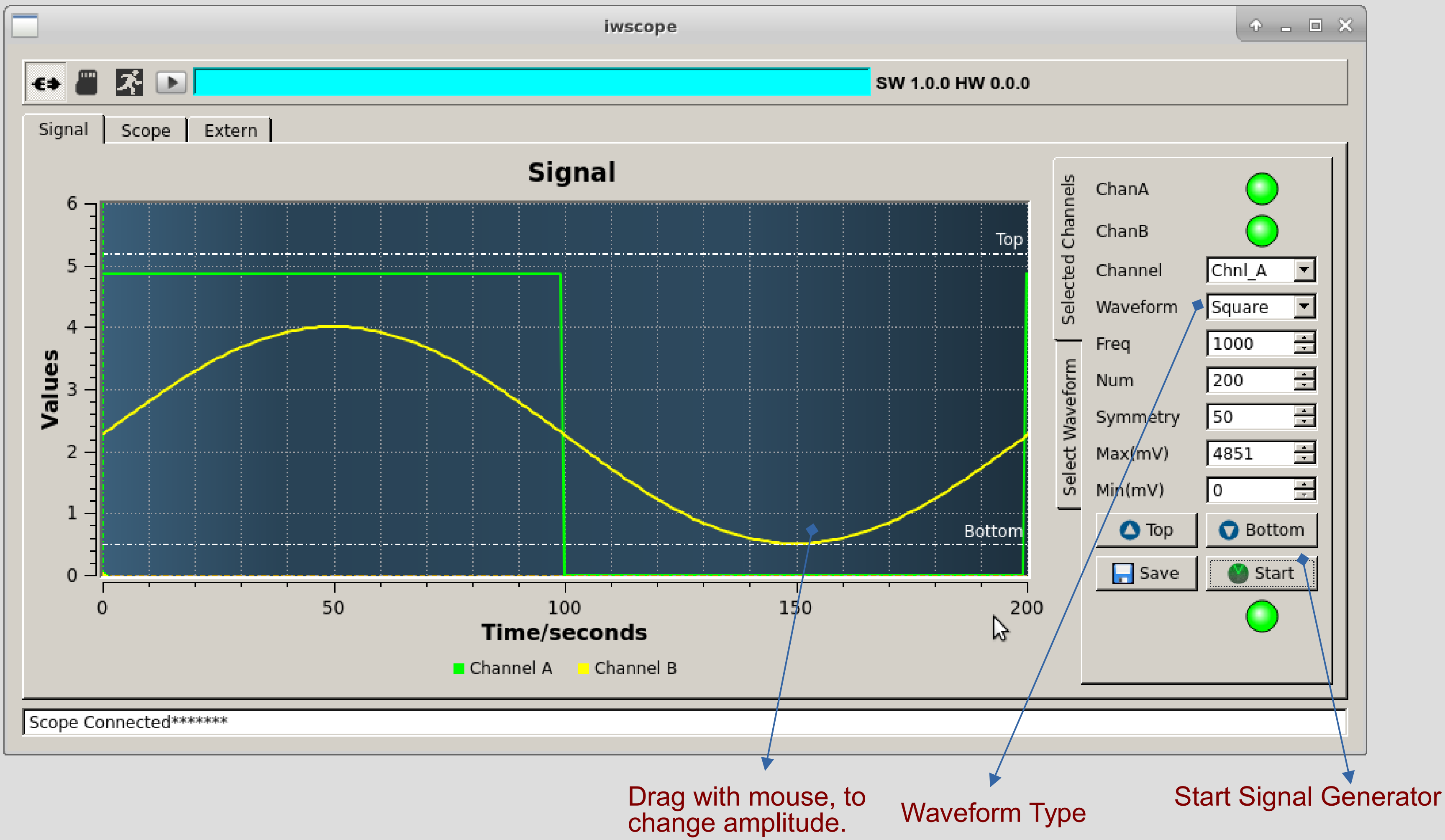
This tools enables you to generate variable signal and capturing output while controlling other sequence can be done directly using plugin interface. It also enables end user to create his own C++/C#/Qt GUI. Plugin feature has been covered heavily in next slides.

Once of feature of plugin lot of colorful feed back along with log file generation can be added as needed that reduces the chaces of error.

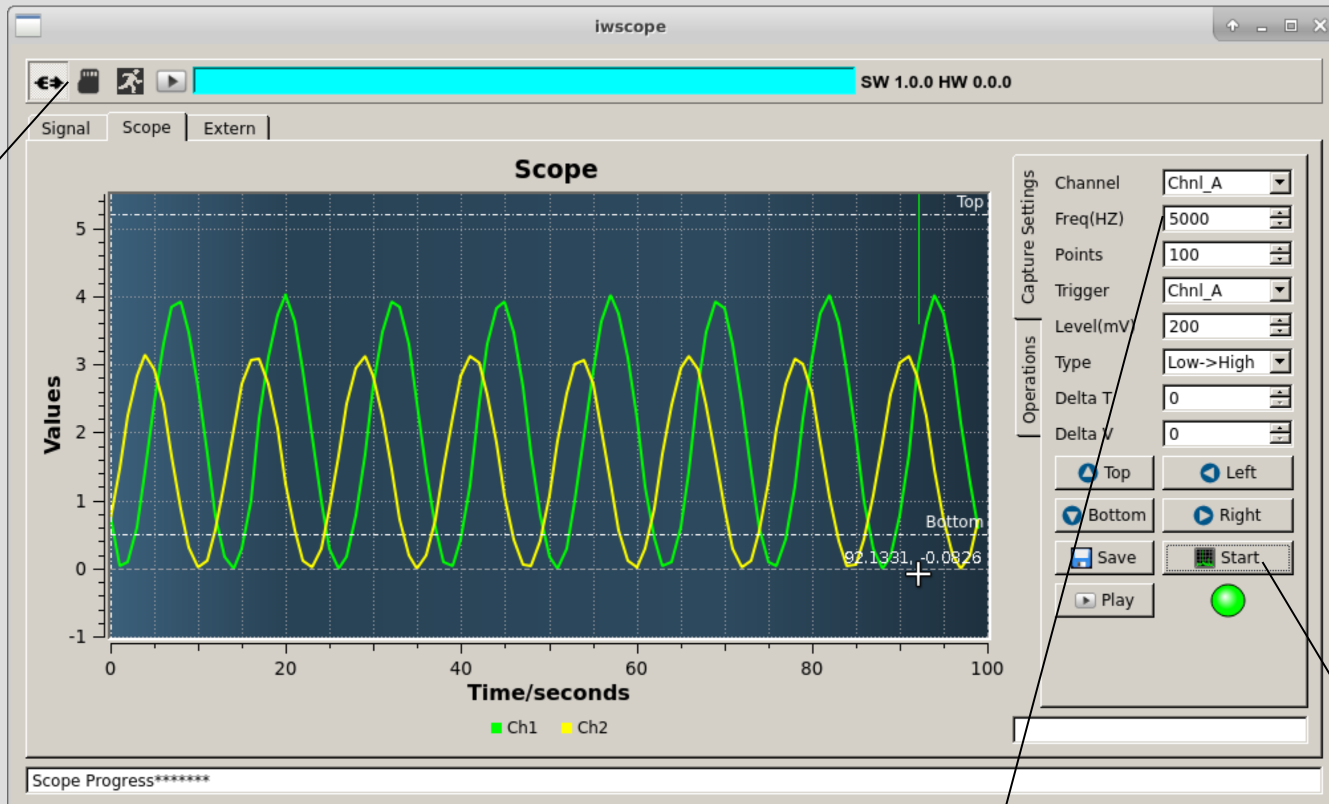
Example: Plugin



Signal Generator



Oscilloscope

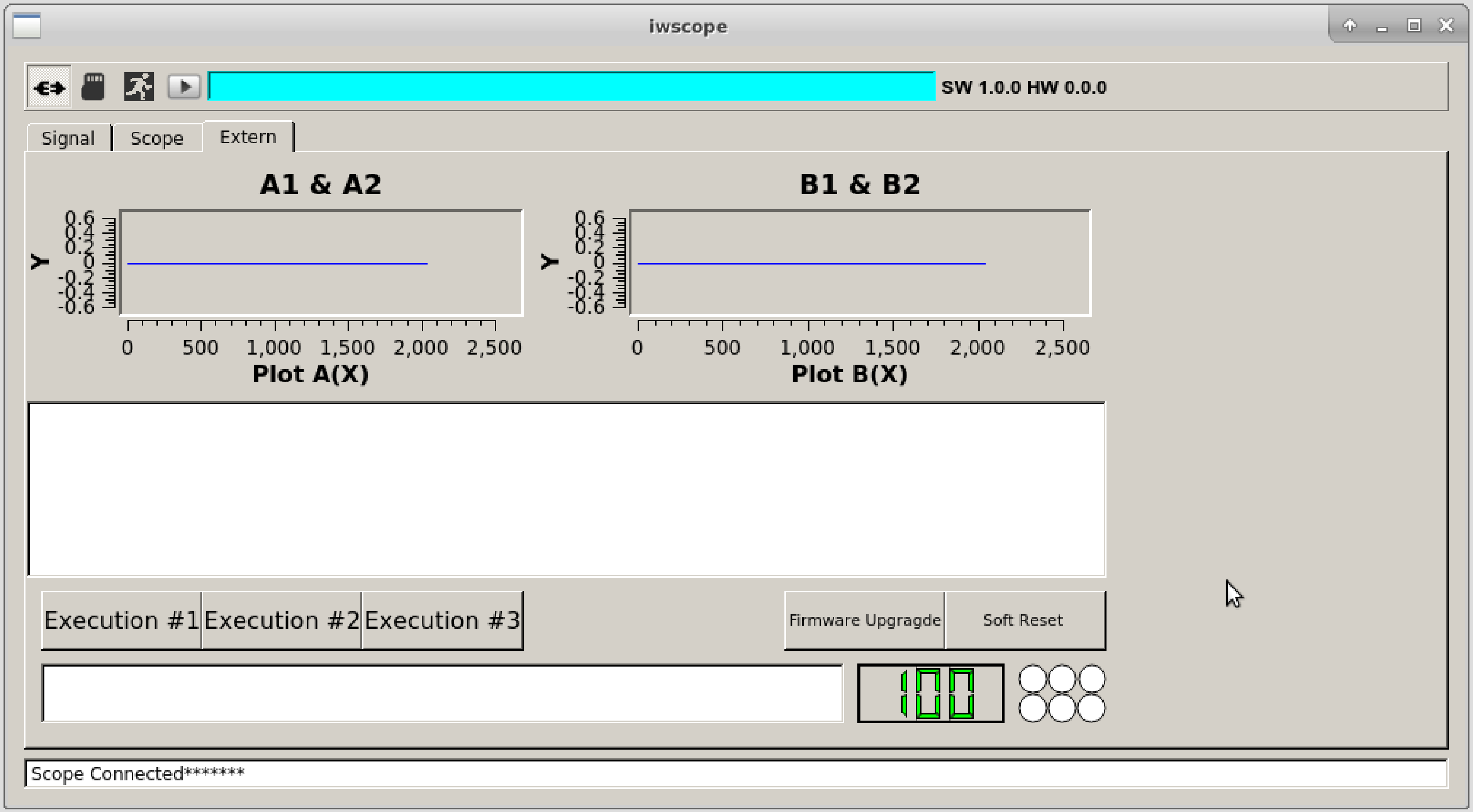


Connect

Start Scope

Select Sampling rate and
number of data points

Plugin Interface



Using First Time (Linux)

1. Open command prompt on Linux terminal (User name vs)
2. Run a sudo command $ sudo addgroup vs dialout
3. Run a sudo command $ sudo addgroup vs plugdev
4. Reboot: Linux Desktop and Open a linux terminal $ prompt
5. $ sudo dmesg -C   ; clears dmesg buffer
6. Connect USB Hardware to Linux machine,
7. Enter $ dmesg command in terminal

Linux setup continued ..

Locate serial port number of Oscilloscope $ dmesg

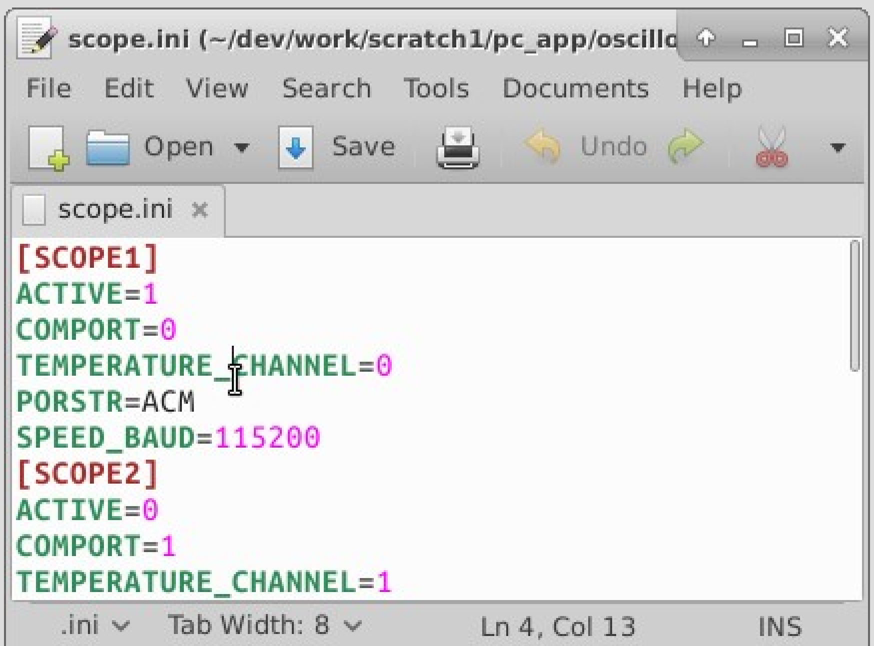
```
[ 9211.320675] usb 2-6.4: USB disconnect, device number 7
[ 9212.046749] usb 2-6.4: new full-speed USB device number 8 using ehci-pci
[ 9212.157482] usb 2-6.4: New USB device found, idVendor=2429, idProduct=0035
[ 9212.157487] usb 2-6.4: New USB device strings: Mfr=1, Product=2, SerialNumb
[ 9212.157490] usb 2-6.4: Product: CDC Calibraion Device TST
[ 9212.157493] usb 2-6.4: Manufacturer: IWSCOPE Inc Massachusetts
[ 9212.157916] cdc_acm 2-6.4:1.0: This device cannot do calls on its own. It is not a modem.
[ 9212.157994] cdc_acm 2-6.4:1.0: ttyACM0: USB ACM device
vs@su64:~/dev/work/scratch1/pc_app/oscilloscope$
```

Running Oscilloscope App

1. Launch iwscope application,

2. Close Application, it creates script.ini file,

3. Edit script.ini using standard light weight editor like Vi or gedit, enter index of comport number, ttyACM0 has COMPORT=0 and ttyACM1 => COMPORT=1 ….



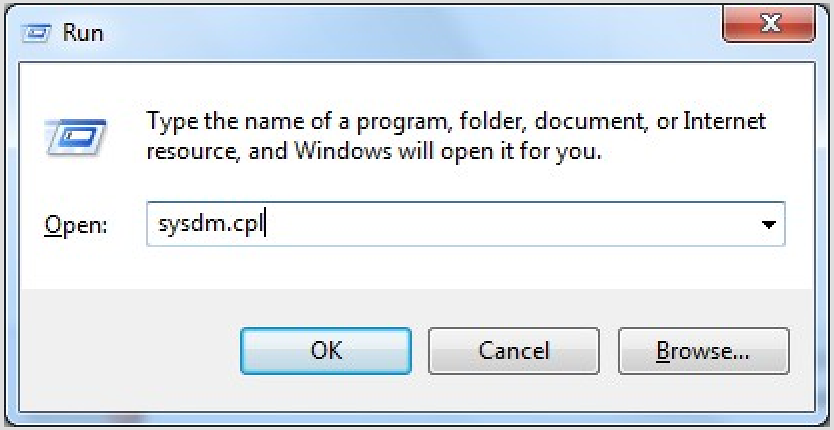
Trouble shooting Linux

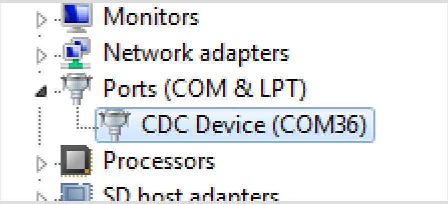
1. Check if user has permission to read and write serial port using minicom or picocom.
2. Provided libraries for following Linux platforms.(Make sure hw/OS matches)
   a. Raspberry PI (Raspebian)
   b. Beagle Bone Black (Debian)
   c. Intel x86 x64 (Ubuntu 32 & 64 bit)
   d. Intel Edison

Using First Time (Windows)

1. press Windows + R dvmgmnt.msc, open devices

2. Run device manager and locate com port section.

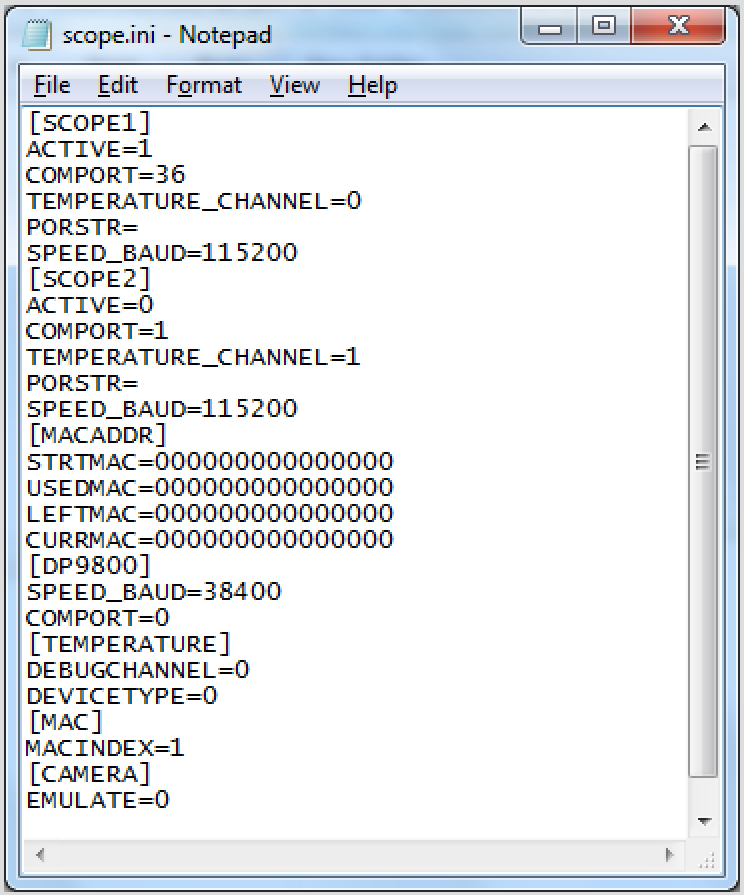


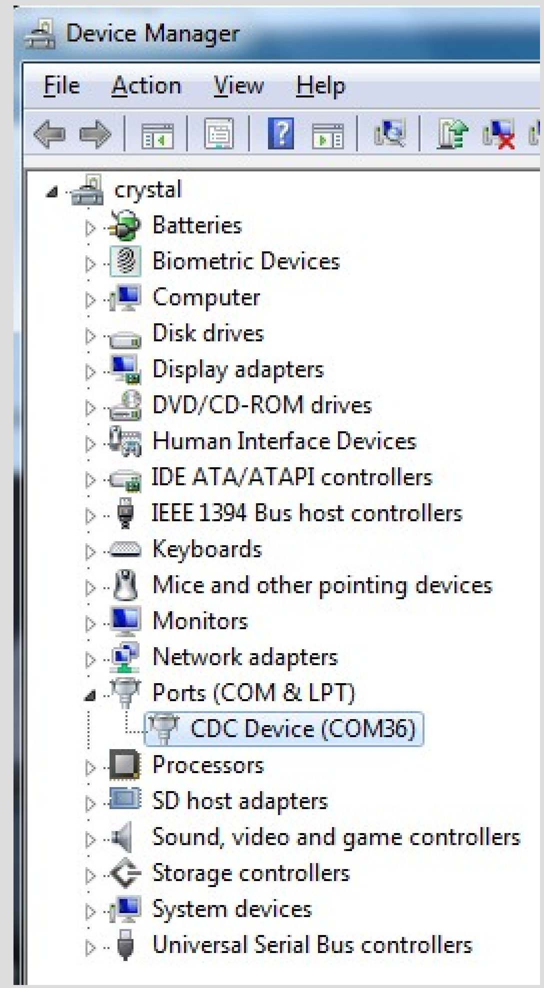


3. This com port should disappear and apper as Scope device disconnected or connected over USB port.

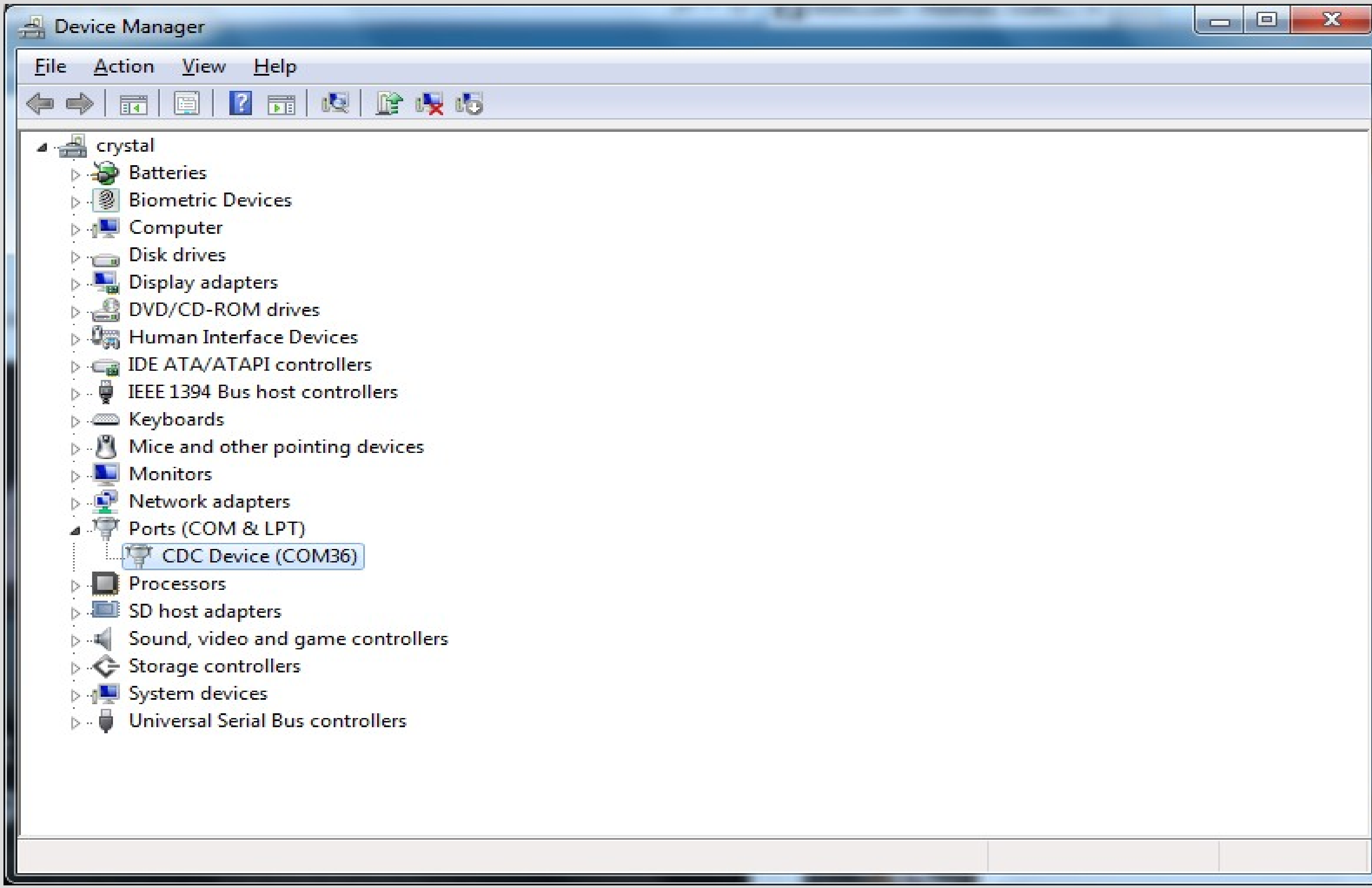
4. Next page shows Edit oscope.ini file, enter correct COM port.

Entering COM port





Locating Windows Serial Port



External custom application integration

1. C++ compiled Libraries for easy integration with external framework.

Example for provided Linux ARM/Intel Windows x86/x64.

2. Sample open source library interface Qt 4.0 code provided as example.

3. Open source TCP/IP plugin interface example code.

4. Fully tested platforms.

Windows XP

Windows 7 (32/64)

Linux 32/64 (Intel x86,x64,Beagle Bone Black,Raspberry PI)

Limitation/Known Issues

1. Because of Microsoft WHQL device certification serial port does not enumerates for Windows 7 x64 bit,(Need to disable driver signing enforcement, press F12 during boot up to disable)

2. USB 2.0 Full speed is speed limitation. On board LED is reversed mounted so does not function.

3. Basic Linux/Windows comport configuration is needed to locate enumerated com port.

4. Plug-in feature requires basic "C" programing knowledge to use feature.

5. All examples uses Qt4.0 frame work.

6. Open source TCP/IP plug-in interface example code.

How to Upgrade Firmware

1. Download latest firmware for this device.

https://github.com/vijayandra/oscopev1

2. Connect Rx/TX of UART 1 as shown in picture, connect USB device, devices maps as **HID** device.

3. Start boot loader application and browse to downloaded file and start boot loader. Estimated boot time is around 30 seconds it should be complete.

4. Unplug USB device, remove jumper joining Rx/Tx so device does not enter in boot loader mode.

5. Start using device, Oscilloscope application should show new version on app when connected.

6. Main application does not need any installer it can be downloaded as single zipped file.

Sample code and other features

1. Following sample code included,
    a. UART/GPIO/I2C Master/Analog/Digital control C# sample code using C++ DLL.
    b. Console C++ application using GCC Windows/Linux ARM/Intel platform which includes input capture/output capture/gpio/i2c master/TTL UART/signal generation and analog data capture.
    c. Sample dual channel signal generation code using csv file, which generates analog signal which using csv file.

2. Step by step how to turn your Raspberry PI/Beagle bone black system into Oscilloscope or Signal generator.

3. Hardware is sold under educational license which is cheaper and support only included via web.

How to USE Software

User may choose to interact with hardware using following three possible modes.

1. Local Direct GUI mode (Under this mode user may choose to run scope which is connected to host system directly via USB CDC)
2. Local Direct Custom Application Plugin Mode (Under this mode user created application interacts with hardware, types of Qt sample c++ cli application library and header files provided along with support library).
3. Remote Plugin Mode,(Under this mode user's custom application interacts via TCP/IP pipes (Port 9999 and 9998 respectively)

Mode #2 and #3 are ideal for creating automated test sequence, Sample C# application also been provided which generates waveform, reads GPIO, creates PWM and measures PWM.

All library and main application are fully tested on Ubuntu (ARM & Intel), Windows (XP, 7, and 8).

How to Control Scope/Signal sampling Freq

User may control sampling freq like example given below,

PIC32 running at 80MHz (80x1000000/TMR)= 80KHz, provided selected (TMRDiv) divider is 0. Similar calculation are applicable for signal generator.

0=Main Clock Divided by 1 (Above 80MKhz Clock)

1=Main Clock Divided by 4 (replace 80 with 20,Result 20KHz)

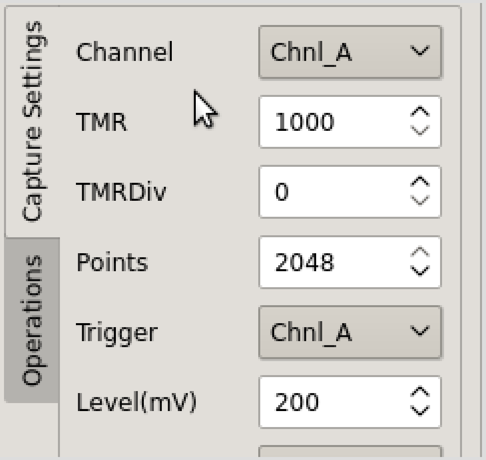
2=Main Clock Divided by 8 (replace 80 with 10,Result 10KHz)

3=Main Clock Divided by 16

4=Main Clock Divided by 32

5=Main Clock Divided by 64

6=Main Clock Divided by 256



Sample I2C Example, custom user Application(Matrix Orbital I2C address 0x28)

```c
// Start a new Frame
lPushTag(INIT_TAG,crFrm.ucbuff,0);
// Initialize I2C Master speed 2000 Hz
lPushTag(I2C2SPEED,NULL,2000);
//Send Command to Hardware
if(CreateMgrCmd(mgrTAG,0,&crFrm))   printf("successfull Arr registered\n");
//Wait Until Completes
while(!usCommandStatus(mgrTAG));

k=0;
while(1)
{
    // Start a new Frame
    lPushTag(INIT_TAG,crFrm.ucbuff,0);
    // I2C Start Condition on I2C Bus
    lPushTag(I2C2START,NULL,0);
    byteArr0[0]= I2C_DISPLAY_ADDR;
    // Followed by Slave 7bit Address ie 0x28 Matrix Orbital
    lPushTag(I2C2WRITEADDR,byteArr0,1);


    // Followed by two bytes, one command and otehr action
    byteArr0[0]= I2C_COMMAND;
    byteArr0[1]= CLEAR_DISPLAY;
    lPushTag(I2C2DATA,byteArr0,2);
    // I2C Stop condition on Bus
    lPushTag(I2C2STOP,NULL,0);
    ////////
    // I2C Start condition
    lPushTag(I2C2START,NULL,0);
    byteArr0[0]= I2C_DISPLAY_ADDR;
    // Slave 7bit Address ie 0x28 Matrix Orbital
    lPushTag(I2C2WRITEADDR,byteArr0,1);

    // Text string to be sent to I2C device
    sprintf(buffer,"#Counter=%d",k++);
    lPushTag(I2C2DATA,(unsigned char *)buffer,strlen(buffer));
    // I2C Stop condition
    lPushTag(I2C2STOP,NULL,0);

    // Send to hardware
    if(CreateMgrCmd(mgrTAG,0,&crFrm))   printf("successfull Arr registered\n");
```