ESTE CUADERNO JUZGO QUE SERIA INTELIGENTE TRANSFORMARLO EN UN .DOCX (word 2010 en adelante) Y SUBIRLO A UN REPOSITORIO GIT Y IR TOCANDOLO CON GIT, ASI HERNAN VÉ Q USAMOS GIT ADEMAS DE Q HACEMOS EL CUADERNO… PARA ESO NECESITO SUS NOMBRE DE USUARIO DE GITHUB PARA AGREGARLOS COMO COLABORADORES EN EL REPOSITORIO DEL GRUPO QUE SE LO PIENSO PASAR A HERNAN….

esto ya lo hice y tengo el repo con el archivo además de que meti ahi un programa en python que hace la consulta del osciloscopio

TOMÁS

19/3/2019

CLASE 1 : git

* git config --global user.name "Guybrush Threepwood"
* git config --global user.email guybrush@example.com

**-configura el usuario realizando los comandos. sirve porque algunas notificaciones respecto de lo que haga el usuario en los repositorios te las manda el git al email indicado.**

* git init

**crea un repositorio local en la carpeta local en donde la terminal esté ubicada.**

* ssh-keygen

**genera 2 archivos:**

**1- nombre-pub es una clave ssh para vincular la terminal local a un servidor git de forma segura, encriptada, debe agregarse a la cuenta de github menu settings>SSH and GPG keys**

**2- otro archivo de mismo nombre sin extensión que es la *private key***

**éste ultimo debe agregarse al agente ssh local mediante los comandos:**

ssh-agent -s

**eso devuelve algo como “>Agent pid 59566” donde el nro será distinto en gral.**

ssh-add ~/.ssh/[nombre archivo sin extensión]

**esto último asocia la private key**

cd ~/.ssh **cambia la ubicación a la subcarpeta que contiene la clave ssh creada**

cat id rsa .pub **muestra la clave creada**

* git clone [URL]

**copia localmente un repositorio remoto que tenga la URL indicada**

Ejercicio (ejemplo): git clone [git@gitlab.com](mailto:git@gitlab.com):talleres-comcom/taller-git-ejercicio.git

* git add [filename]

**pasa el archivo “filename” al estado *staged* en el repositorio actual. o sea se lo presentamos al git.**

Ejemplo: git add archivo.txt

* git commit -m [mensaje]

**confirma todos los cambios de los archivos en estado *staged* y los agrega a un “log” o vitácora con el mensaje indicado entre comillas.**

Ejemplo: git commit -m “cambié tal cosa en tales archivos”

* git commit --amend -m [nuevo mensaje]

**arregla un mensaje mal hecho**

* git revert [hash]

**Buscamos el hash del commit a cambiar usando** git log **y usamos este comando**

* git stash

**Se guarda el estado actual de los archivos**

**modificados y nos deja el directorio limpio.**

**Para volver a mostrar los cambios guardados ejecutamos git stash**

**apply.**

* git rm [archivo]

**preborra un archivo y marca esta acción como staged.**

* git mv [archivo] [nuevo nombre/ubicación]

**pre - mueve/renombra archivo y marca como staged**

* git diff

**Para esto tenemos el comando git diff, que muestra las diferencias**

**entre el estado actual de los archivos y la ´ultima vez que hicimos git add**

**(cambios marcados como staged).**

**Si, en cambio, queremos ver las diferencias entre los cambios marcados**

**como staged y los que confirmamos en el ´ultimo commit, podemos usar**

**git diff --staged**

* git log

**muestra el historial de cambios**

* git branch

**crea una rama en el repositorio**

* git checkout [rama]

**cambia a la rama indicada**

* git merge [nombre de la rama a fusionar]

**fusiona la rama actual la indicada. cambia sólo la rama actual**

* git status

**muestra el estado en el que se encuentran los archivos en el repositorio actual**

ADMINISTRACIÓN DE REPOSITORIOS REMOTOS

* git remote add [nombre repositorio] [URL]

**agrega un repositrio remoto y lo vincula al local.**

* git remote show [repositorio remoto]

**Devuelve información del repositorio remoto indicado, si no se pone nada te dice los nombres de los repositorios remotos asociados.**

* git remote -v

**muestra todos los repositorios remotos asociados**

* git push/pull [remoto] [branch]

**envía/recibe al repositorio remoto los cambios *staged* de archivos locales presentes en el repositorio remoto**

**si se queja porque dice que hay cosas en el remoto que no tenemos y quizá queramos le podemos forzar el push con**

git push -f [repo] [rama]

26/03/19

CLASE 2: COMUNICACIÓN CON INSTRUMENTOS

En el editor raro de mensajes de commit del GitBash, escribir el mensaje después de las líneas con # y rematarlo con :qw o :wq (probar)

Pyvisa **requiere** de NI VISA instalado para funcionar, porque el NI VISA es el que realmente se comunica directamente con los instrumentos

python -m visa info “Para ver los backends instalados mediante línea de comando en la terminal.”

pip install pyvisa “Instala por terminal.”

USB::fabricante::modelo::nroSerie::INSTR “Es el estándar para identificar un dispositivo USB.”

query\_ascii\_values “Comando del paquete pyvisa”

Con este comando tratamos de levantar una tira de datos, este comando debería transformar bien una serie de datos con encoding ascii.

query\_binary\_values

Y luego tratar de obtener la misma tira usando el instrumento con encoding en binario, puesto que recibir los datos así es más eficiente para la transferencia. Este comando de pyvisa lo transforma automáticamente a números de base 10.

(Por ejemplo, en binario, 255 se escribe usando 1 byte y en ascii son 3. Ascii mapea del 1 al 255 un montón de símbolos, letras y números y sirve porque es más legible que el binario…) **Preguntar esto**

\***Preguntar a Hernán** que explique de nuevo el tema de por qué una serie de 8 unos y ceros pesa 1 bit… por ejemplo, la letra *a* merece una tira de unos y ceros, ¿Por qué en binario la *a* pesaría 1 bit???... ¿Un bit no equivale a un número sea la base que sea? ¿No equivale a la relación entre 1 pulso eléctrico digital y un valor?

-por lo que leí el osciloscopio trabaja con bytes de 8 bits….o sea que quizá Hernán dijo que un número cualquiera pesa 1 byte en binario y que se representa con 8 bits, que son 8 zeros y unos… esto tiene sentido.

Manual de programación del osciloscopio:

pág 2-40 del manual

“Internally, the oscilloscope uses one 8-bit data byte to represent each waveform data point, regardless of the acquisition mode.

The DATa:WIDth command lets you specify the number of bytes per data point when transferring data to and from an oscilloscope. This provides compatibility with other digitizing oscilloscopes.”

El osciloscopio trabaja internamente con 8 bits = 1 byte = 1 dato.

**ASCII:** “One-byte-wide data ranges from –128 to 127. Two-byte-wide data ranges from –32768 to 32767.”

O sea que datos de 1 byte de 8 bits permiten 2^8 = 256 =128\*2 (incluye el 0) combinaciones o valores distintos del dato.

Y datos de 2-bytes sería eso al cuadrado: 256^2=32768\*2 OK

“Each data value requires two to seven characters. This includes one character for the minus sign if the value is negative, one to five ASCII characters for the waveform value, and a comma to separate data points”

Esto no entiendo, dice que cada dato llega a 7 caracteres...pero 1 caracter = 8 bits , entonces si el osciloscopio trabaja con datos de 1 byte de 8 bits, ¿cómo puede ser que 1 dato puede tener 7 caracteres componentes que pueden valer 2^8 cosas distintas cada uno en principio! particularmente la memoria interna no lo permite: “7 caracteres: 5 son números del 0 al 9; 1 es el signo (2 valores +/- o 1/0); 1 es una coma (todos de 8 bits cada 1)”

=> 10^5\*2+1=200001 valores en total….no me cierra

Pág 2-248

“Y specifies a normal waveform where one ASCII or binary data point is transmitted for each point in the waveform record.

For Y format, the time (absolute coordinate) of a point, relative to the trigger, can be calculated using the following formula. N ranges from 0 to 2499.

**Xn = XZEro + XINcr (n -- PT\_OFf)**

For Y format, the magnitude (usually voltage, relative to ground)

(absolute coordinate) of a point can be calculated:

For Y format, the magnitude (usually voltage, relative to ground)

(absolute coordinate) of a point can be calculated:

**Yn = YZEro + YMUIty (yn -- YOFf)**” **(nos faltaron estos datos para la transformación)**

**XINcr:**

The set form of this command specifies the interval (seconds per point for non-FFT, Hertz per point for FFT) between samples of the reference waveform specified by the DATa:DESTination command.

The oscilloscope uses this value to calculate the seconds/division or Hertz/division units shown in the status bar and cursor readouts when displaying a reference waveform.

**XUNit**

For all model and firmware combinations except the TDS200 series with a TDS2CMA communications module, the set form of this command specifies the horizontal units (”s” for seconds and “Hz” for Hertz) for the reference waveform specified by the DATa:DESTination command. Setting a reference waveform to Hz causes the oscilloscope to display the waveform as an FFT waveform.

**XZEro**

The set form of this command specifies the position, in XUNits, of the first sample of the reference waveform specified by the DATa:DESTination command, relative to the trigger.

**YMUlt** is a value, expressed in YUNits per digitizer level, used to convert waveform record values to YUNit values using the following formula (where dl is digitizer levels):

**value\_in\_YUNits = ((curve\_in\_dl – YOFF\_in\_dl) \* YMUlt) + YZERO\_in\_YUNits**

The set form of this command sets the vertical scale factor of the reference waveform specified by the DATa:DESTination command, expressed in YUNits per digitizing level.

**YOFf** is a value, expressed in digitizer levels, used to convert waveform record values to YUNit values using the following formula (where dl is digitizer levels):

**value\_in\_YUNits = ((curve\_in\_dl – YOFF\_in\_dl) \* YMUlt) + YZERO\_in\_YUNits**

The set form of this command stores a value for the reference waveform specified by the DATa:DESTination command. This value does not affect how the oscilloscope displays the waveform, but does affect the cursor readouts.

**YUNit**

For all model and firmware combinations except the TDS200 series with a TDS2CMA communications module or a TDS2MM measurement module, the set form of this command sets the vertical units for the reference waveform specified by DATa:DESTination.

**YZEro**

For all model and firmware combinations except the TDS210 or TDS220 oscilloscope (firmware below V 2.00) with a TDS2CMA communications module, YZEro is a value, expressed in YUNits, used to convert waveform record values to YUNit values using the

following formula (where dl is digitizer levels):

**value\_in\_YUNits =((curve\_in\_dl – YOFF\_in\_dl) \* YMUlt) + YZERO\_in\_YUNits**

YZEro is used when calculating cursor readouts.

Si el cable tiene binarios y usamos query\_ascii tira un error… **read** lee y transforma o decodifica y si no coincide la codificación no entiende lo q lee.

osci.read\_raw() me tira la lectura de la info que viene en hexadecimal.

En query\_binary\_values los parámetros tienen que ver con cómo se codifican los datos, esto es clave para poder obtener el resultado en valores humanamente legibles… en el manual de programador del osciloscopio están las codificaciones posibles.