COMPLETO AR 65 BJT MOSFET

Reporte de nuevas funcionalidades agregadas al proyecto

8 de junio del 2021

PROGRAMA "ARDUINO AR 65.ino"

A.1. PROMEDIADOR:

DESCRIPCION: Es un filtro circular promediador de 32 muestras.

FUNCIONAMIENTO: Cada muestra promedida que se envía a Python, es el resultado de:

1. Leer el canal ADC correspondiente 32 veces.
2. Cada lectura se guarda en un buffer
3. La muestra promediada es el resultado del promedio simple de las 32 muestras.
4. Se RESALTA el hecho de que cada muestra promediada es la que se envía al programa Python en un solo tiempo de muestreo, es decir, no se están promediando muestras pasadas no futuras.

PROGRAMA "Completo Ar 65 BJT_MOSFET.py"

PY.1. SINCRONIZACION CON LA COMUNICACIÓN SERIAL CON ARDUINO

DESCRIPCION: El programa Pyhton se sincroniza con el inicio de envío de datos válidos de ARDUINO.

FUNCIONAMIENTO: El programa Python, en su Clase "Muestras":

1. Iinicia la comunicación serial con ARDUINO, esperando recibir la palabra clave (cadena) "inicio". En este paso puede suceder que Python reciba basura del buffer serla de arduino.
2. Esta palabra clave la envía ARDUINO en su sección "Setup".
3. Una vez que dicha palabra clave la detecta Python, los datos que siguen (enviados desde la sección "loop" de ARDUINO), se consideran válidos y se reciben y almacenan de manera secuencial en el programa Python.

PY.2. SEPARACION EN "ARREGLOS" DE LAS MUESTRAS RECIBIDAS DE ARDUINO

DESCRIPCION: En el programa ARDUINO se envían 300 muestras para el trazador de curvas de transistores (BJT/MOSFET). Estas muestras están divididas en paquetes de 50 muestras, los cuales se guardan en arreglos correspondientes a cada "canal" (llave) especificada en Python. Por ejemplo, si el "canal" (llave) se llama "Ic", para la cual se reciben 300 muestras, "Ic" se manejará como un arreglo de longitud 6 de la siguiente manera:

Ic[0] = Arreglo de las muestras 0:49

Ic[1] = Arreglo de las muestras 50:99

Ic[2] = Arreglo de las muestras 100:149
Ic[3] = Arreglo de las muestras 150:199
Ic[4] = Arreglo de las muestras 200:429
Ic[5] = Arreglo de las muestras 250:299
Igual para cada "canal" definido en Python: Vce[0], Vce[1], Vce[2], Vce[3], Vce[4], Vce[5]; Vbe[0], etc.

PY.3. FILTRADO DE LAS MUESTRAS RECIBIDAS DESDE ARDUINO

DESCRIPCION: El programa Python en su Clase "Muestras":

1. Filtra automáticamente las muestras recibidas de ARDUINO a través de dos filtros diferentes: (A) un filtro circular (PROMEDIDOR) similar al que está programado en ARDUINO, con un buffer de 4 muestras; y (B) un filtro pasa bajas con un parámetro alpha de 0.5 [promedio = Promedio_Anterior*alpha + Nueva_Muestra*(1-alpha)]
2. Cada tipo de filtro genera muestras adicionales a las recibidas de ARDUINO, las cuales quedan almacenadas en el diciionario "muestras". La nomenclatura asignada a cada conjunto de muestras de cada filtro es la siguiente:
3. FILTRO CIRCULAR (PROMEDIADOR): Ic_pc[0] correspondiente al filtrado de las muestras respectivas IC[0], Ic_pc[1] corresponden al filtrado de las muestras Ic[1], y así sucesivamente.
4. FILTRO PASA BAJAS: Ic_lp[0] correspondiente al filtrado de las muestras respectivas IC[0], Ic_lp[1] corresponden al filtrado de las muestras Ic[1], y así sucesivamente.

PY.4. DICCIONARIO DE MUESTRAS EXPANDIDO

DESCRIPCION: En el programa Pyhon, en la clase "Muestras", se genera el diccionario "muestras" en donde, además de guardar la muetsra originales recibidas de ARDUINO, se guardan las muestras filtradas con los dos filtros anteriores, de tal manera que el diccionario resulta expandido como se muestra a continuación:

--- VBB : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- Vbe : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- Ib : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- VCC : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- Vce : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- Ic : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- VBB_pc : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- Vbe_pc : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- Ib_pc : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- VCC_pc : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- Vce_pc : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- Ic_pc : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- VBB_lp : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- Vbe_lp : [[0:49][50:99][100:149][150:199][200:249][250:299]]

--- Ib_lp : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- VCC_lp : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- Vce_lp : [[0:49][50:99][100:149][150:199][200:249][250:299]]
--- Ic_lp : [[0:49][50:99][100:149][150:199][200:249][250:299]]

El cual puede accesarse como se indica a continuación:

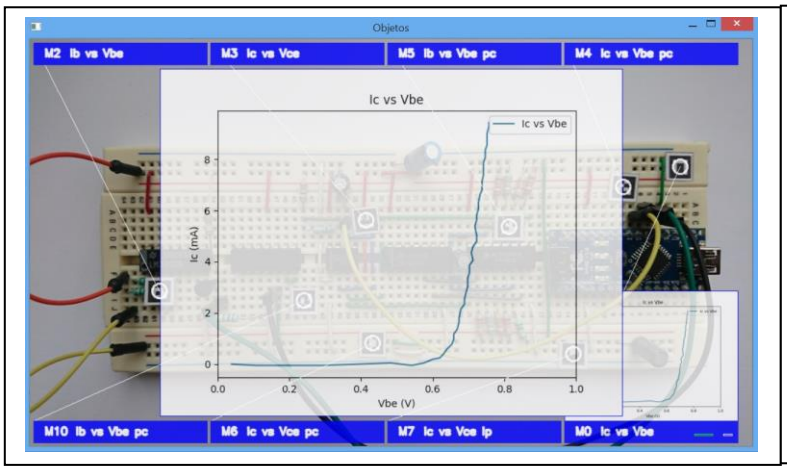
--- VBB[0], VBB[1], VBB[2], VBB[3], VBB[4], VBB[5]
--- Vbe[0], Vbe[1], Vbe[2], Vbe[3], Vbe[4], Vbe[5]
--- Ib[0] …
--- VCC[0] …
--- Vce …
--- Ic …
--- VBB_pc …
--- Vbe_pc …
--- Ib_pc …
--- VCC_pc …
--- Vce_pc …
--- Ic_pc …
--- VBB_lp …
--- Vbe_lp …
--- Ib_lp …
--- VCC_lp …
--- Vce_lp …
--- Ic_lp …

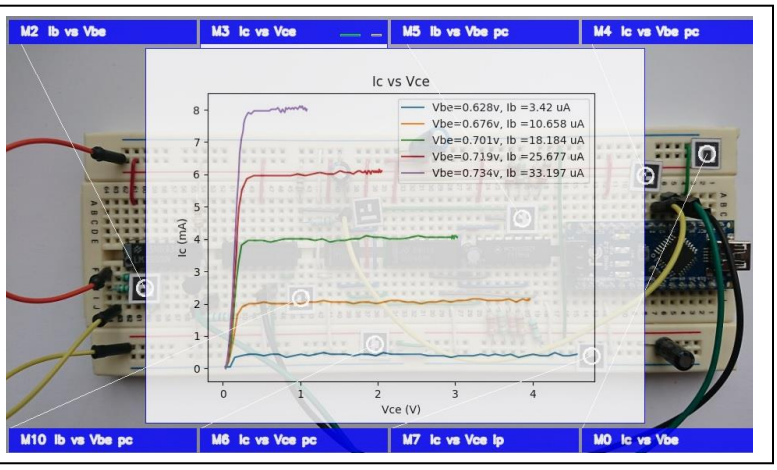
Los primero cinco arreglos [del 0 al 4] corresponden a la familia de curvas Ic vs Vce (Id vs Vds), por lo cual sólo esos arreglos contienen información válida para estas gráficas. El último arreglo (5) contiene información válida para las curvas Ic vs Vce (Id vs Vgs) e Ib vs Vbe (Ig vs Vgs).

RESULTADOS OBTENIDOS:

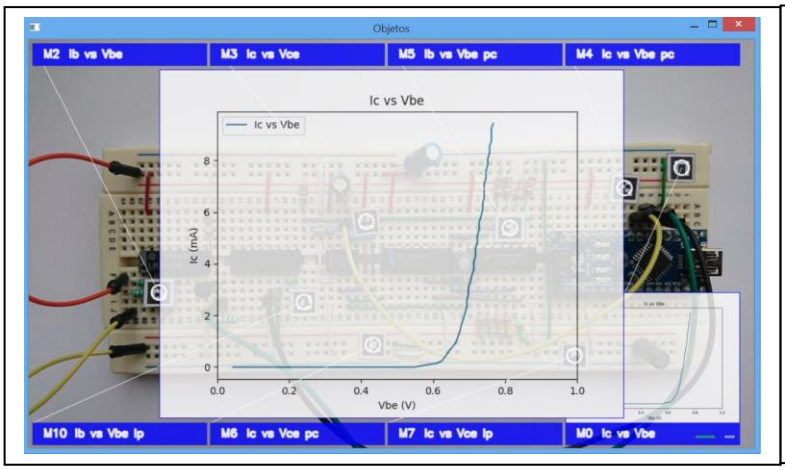
Con los cambios propuestos se obtuvieron los resultados mostrado a continuación, en donde se puede observar una gran mejora en la “limpieza” de las curvas obtenidas.

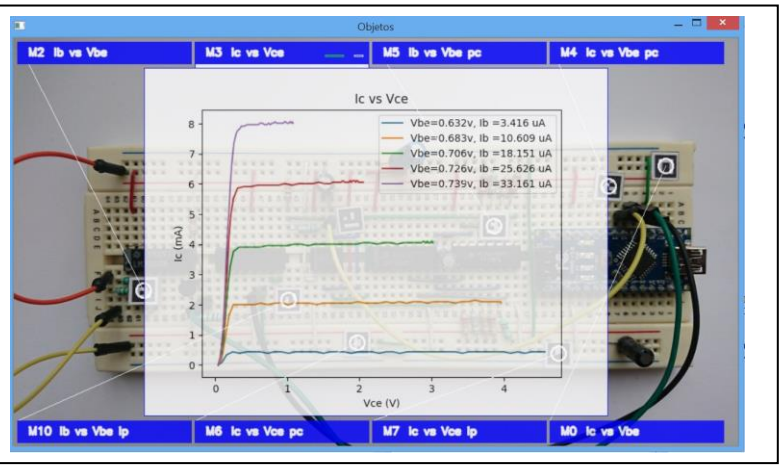
1. Curvas obtenidas SIN utilizar FILTRO en ARDUINO y SIN utilizar FILTRO en Python





2. Curvas obtenidas CON FILTRO en ARDUINO y SIN utilizar FILTRO en Python





3. Curvas obtenidas CON FILTRO en ARDUINO y CON FILTRO (PROMEDIADOR) en Python