Neuromorphic engineering I Lab 1: Automated Data Acquisition and Analysis Group number: Team member 1: Team member 2: Date: You need to bring your own USB cable to connect the board with your own PC. The objectives of this lab are as follows: To become acquainted with the experimental board setup. To become acquainted with Python for data acquisition and manipulation. • To measure and characterize a NFET. The aim of this first lab is to familiarize everyone with the lab equipment and software. 1. Getting started 1.1 Class chip documentation You can find the documentation for the classchip here: https://drive.google.com/drive/u/0/folders/1VBPKVfS9zwu_I2ExR1D0jU2eCSgleoQG 1.2 Python If you are new to Python, you can find a detailed tutorial at the link: https://docs.python.org/3/tutorial/. In particular the concepts in 3. An Informal Introduction to Python and in 4. More Control FLow Tools will be useful for the exercises you will have to solve. 1.3 Report You hand in the report (prelab + lab report + postlab) in both ipynb and pdf format as a group. Make sure that the markdowns are ran and the generated figures are shown. The deadline is the beginning of the next lab. There is no prelab for this week, but from the next week on you should finish the prelab before the lab starts. 1.4. Virtual machine If you are a Linux user you can skip this step. If you are a Windows or Mac user, we have set-up an Ubuntu-based virtual machine with the necessary libraries to communicate with the board. To import the existent virtual machine, Windows users should install VirtualBox. Mac users can either install VirtualBox or Parallels. If you want to use a pre-configured VirtualBox virtual machine with all the tools pre-installed then follow the instructions in the below. To install VirtualBox, please follow these steps: For Windows users **Step 1.** Dowload VirtualBox You can download VirtualBox from the link: https://www.virtualbox.org/wiki/Downloads, VirtualBox 6.1.26 platform packages -> Windows hosts Execute the .exe file and follow the steps. In the window "Would you like to install this device software" select Install. Once the installation is complete, press Finish and launch the VirtualBox. Step 2. Download the Ubuntu 20.04 ISO file You can download the necessary files for the NE-I virtual machine from the link: https://ubuntu.com/. Click 'Download' option on the Ubuntu website, on the left side of 'Ubuntu Desktop', you can click the green '20.04 LTS' button to download Ubuntu 20.04 ISO file. Step 3. Import the file in VirtualBox and install Ubuntu Once you launch VirtualBox, the window Oracle VM VirtualBox Manager should appear on your screen. Then follow this link https://itsfoss.com/install-linux-in-virtualbox/ from 'Step 3: Install Linux using VirtualBox'. For Mac users **Step 1.** Dowload VirtualBox You can download VirtualBox from the link: https://www.virtualbox.org/wiki/Downloads , VirtualBox 6.1.26 platform packages -> Windows hosts Execute the .dmg file and follow the steps. Once the installation is complete, press Finish and launch the VirtualBox. Step 2. Download the Ubuntu 20.04 ISO file You can download the necessary files for the NE-I virtual machine from the link: https://ubuntu.com/. Click 'Download' option on the Ubuntu website, on the left side of 'Ubuntu Desktop', you can click the green '20.04 LTS' button to download Ubuntu 20.04 ISO file. Step 3. Import the file in VirtualBox and install Ubuntu Once you launch VirtualBox, the window Oracle VM VirtualBox Manager should appear on your screen. Then follow this link https://medium.com/tech-lounge/how-to-install-ubuntu-on-mac-using-virtualbox-3a26515aa869 2. Python exercises 2.1 Jupyter Lab You can install Jupyter Lab or Jupyter Notebook according to this link:https://jupyter.org/install 2.2 Making plots • Plot a Sine curve from 0 to 2π with 20 points. (Hint: Do not forget to properly label the axis and add figure legends when necessary.) import numpy as np import matplotlib.pyplot as plt y = ... plt.plot(..., ..., '') plt.xlabel(...) plt.ylabel(...) plt.legend(...) plt.grid() plt.show() • Add a plot of a Cosine (Use the same range to the Sine plot, but set a different point marker and color.) In []: • Make a plot of the equation $I_{ds} = I_0 e^{\frac{\kappa}{U_T} V_g}$ using the following parameters. Generate two plots, one with linear scaling on the y axis. Put the right labels on both x axis and y axis. (Hint: use plt.semilogy) Vg = np.arange(0, 1, 0.001)I0 = 1e-13k = 0.6UT = 25e-3In [6]: Ids = ... In []: In []: 2.3 Saving and loading data You may want to work on the data for your report after the lab. Use np.savetxt('data.csv', data, delimiter=',') to save it to the working directory. # define data data = [Vg,Ids] # save to csv file Check if the data saved is correct by loading it again using np.loadtxt('data.csv', delimiter=',') and plot. # load from csv file $x, y = \dots$ # plot 2.4 Fitting data with a line Sometimes you may need to find the relationship between data using regression. Try to extract the slope and intercept of the following given data using linear regression with np.polyfit x = np.array([1, 3, 5, 7])y = np.array([6, 3, 9, 5])# compare the original data points and the fitted line 3 Experiments From now on you will be using the real board! 3.0 How to install pyplane In order to communicate with the chip through Python you need to install pyplane, a library that provides an easy interface to control the chip from jupyter notebook. The interface requires to use Ubuntu 20.04. You can follow different methods based on your machine's operating system. 3.0.1 Installing *pyplane* using pip Open a command window and run pip install pyplane 3.0.2 Set up the USB connection of the teensy board in the Virtual machine Windows users 1. check teensy vendor and product id Isusb you should see something like "XXXX:YYYY" for teensy 1. add usb rules cd /etc/udev/rules.d sudo gedit 10-my-usb.rules Add the information in this 10-my-usb.rules file and save it: ATTR{idVendor}=="XXXX", ATTR{idProduct}=="YYYY", MODE="0666", GROUP="dialout" 1. add user to group and change mode sudo usermod -a -G dialout \$USER sudo chmod a+rw /dev/ttyACM0 1. restart the virtual machine Mac Users Step 1 Settings --> Ports --> USB Add Teensyduino USB Serial Step 2 Device --> Connect Step 3 Change mode: sudo chmod a+rw /dev/ttyACM0 Step 4 Restart kernel in Jupyter noytebook Step 5 Run the code again 3.1 Load the firmware (you can skip this step) You don't need to load the firmware, we have done it for you. To load the firmware, please follow these steps: Step 1. Dowload and install the Teensy Loader wget https://www.pjrc.com/teensy/teensy_linux64.tar.gz wget https://www.pjrc.com/teensy/00-teensy.rules sudo cp 00-teensy.rules /etc/udev/rules.d/ tar -xvzf teensy_linux64.tar.gz ./teensy & Teensy -File Operation Help **Press Button** manually enter Program Mode main.hex, 2% used You will see Step 2. Load main.hex Connect the board to your computer by USB Teensy -File Operation Help Press Button on Teensy to manually enter Program Mode main.hex, 2% used Open maim.hex file Teensy -File Operation Help main.hex, 2% used Press button on Teensy to manually enter Program Mode Teensy -File Operation Help Program Teensy -File Operation Help main.hex, 2% used Reboot 3.2 Set up the communication between jupyter notebook and PCB Verify that the LEDs of your board are on as in the following picture # import the necessary library to communicate with the hardware import sys import pyplane import numpy as np import matplotlib.pyplot as plt In [2]: # create a Plane object and open the communication if 'p' not in locals(): p = pyplane.Plane() p.open('/dev/ttyACM0') except RuntimeError as e: print(e) Make sure all these steps are executed correctly (the [*] in front of the line turns into a number) # Send a reset signal to the board reset_type = pyplane.ResetType(0) p.reset(pyplane.ResetType.Soft) Out[3]: <TeensyStatus.Success: 0> Was the LED flashing? How? #NOTE: You must send this request events every time you do a reset operation, otherwise the recieved data is wrong. p.request events(1) You could check firmware version, which should be 1.8.3. In [6]: p.get_firmware_version() Out[6]: (1, 8, 3) See all the possible functions In [6]: dir(pyplane) Out[6]: ['AdcChannel', 'BitDepth', 'Coach', 'CoachInputEvent', 'CoachOutputEvent', 'CurrentRange', 'DacChannel', 'Plane', 'ResetType', 'TeensyStatus', '__builtins__', _cached__', _doc__', _file__' _loader__' _name__', __package__', _path__', _spec__', 'get_version', 'pyplane'] 3.3 Basic function operation You can set and read voltage using function p.set_voltage and p.read_voltage. Very importantly, all voltage you set on this board must be between 0 and 1.8 V! Now set a voltage at AIN0 p.set_voltage(pyplane.DacChannel.AIN0, 0.62) Out[7]: 0.6193549036979675 Because of the quantization error of the DAC, you may want to see the actual value you have set using p.get_set_voltage In [8]: p.get_set_voltage(pyplane.DacChannel.AIN0) Out[8]: 0.6193549036979675 Read the voltage at adc channel AOUT0 p.read_voltage(pyplane.AdcChannel.AOUT0) Out[9]: 3.064746141433716 Read the voltage at adc channel GO22 In [10]: p.read_current(pyplane.AdcChannel.G022) Out[10]: 4.6386719532165444e-07 **3.4 AER** # uses schemdraw, you may have to install it in order to run it on your PC import schemdraw import schemdraw.elements as elm d = schemdraw.Drawing() Q = d.add(elm.NFet, reverse=True) d.add(elm.Dot, xy=Q.gate, lftlabel='gate=AIN0') d.add(elm.Dot, xy=Q.drain, toplabel='drain=G022') d.add(elm.Dot, xy=Q.source, botlabel='source=GO20') d.draw() Out[11]: drain=GO22 source=GO20 Read the current of the drain of the transistor. print("The measured drain current is {} A".format(I_d)) Do you think this current is reasonable? WHy? Now try to set voltage of this transistor by AER (Address Event Representation). Find the documentation "chip_architecture.pdf" for the classchip introduced in 1.1. See how to select signal communication on pages 11 and 21. Because you need to read the current and write the voltage of the NFET. You have to set demultiplexer by sending the configuration event: In [15]: events = [pyplane.Coach.generate_aerc_event(\ pyplane.Coach.CurrentOutputSelect.SelectLine5, \ pyplane.Coach.VoltageOutputSelect.NoneSelected, \ pyplane.Coach.VoltageInputSelect.SelectLine2, \ pyplane.Coach.SynapseSelect.NoneSelected, 0)] p.send coach events(events) Make sure the chip receives the event by a blink of LED1, if it's not the case, the chip is dead. Now set source GO20 voltage using the function introduced above In []: Set drain GO22 voltage In []: Set trial gate AINO voltage (you can try different voltage between 0~1.8V to see different output current) In []: Read drain GO22 current In []: print("The measured drain current is {} A".format(I_d)) Compare this current with the drain current measured above. Set trial gate AINO voltage 0 and see if the drain current is also zero. If not, why? Now you can try some challenging experiments! Sweep gate voltage between 0~1V and see how the output current change In []: import time #Sweep gate voltage #Initialize current variables #Set Vg = 0 and wait 0.5 second for it to settle time.sleep(0.5) # delay 0.5 second #Read leakage current Ids=Ids0 #Read Ids at Vg sweep and wait for it to settle for n in range(N_samples): p.set_voltage(pyplane.DacChannel.AIN0,Vg[n]) time.sleep(0.05) # delay 0.1 second #Subtract leakage and shunt resistance from read current value #Plot in linear scale 4. Clean up Well done! That's all for today. Remember you have to clean up in the end just as in a real lab! Close you device and release memory by doing In [16]: **del** p Save your changes Download the files you need for the report to your own PC 5. Postlab Questions 1. Why is there no pin for the bulk of NFET? What is it's voltage then? 1. How precise are the measurements of voltage and current using DAC? 1. Do you think building a "computer" whose inputs and outputs are analog voltage/current signals is a good idea? Why or why not?