國立虎尾科技大學 NATIONAL FORMOSA UNIVERSITY

DSDL Digital System Design Lab

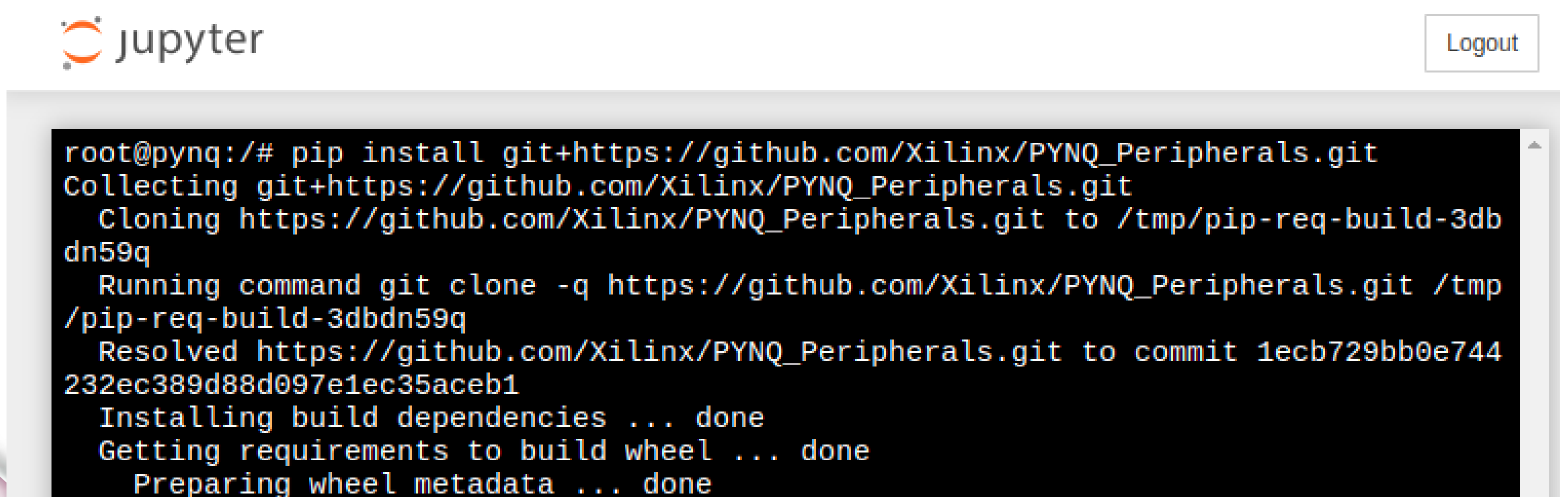
PYNQ-ZU Lab 2

2023/03

Outline

1. Install PYNQ_peripherals
2. Study sensor's characteristics
3. Examine example projects
4. GUI control on Jupyter Lab

1. Install PYNQ_peripherals



1. Install PYNQ_peripherals

- Run commands:

$ pip install git+https://github.com/Xilinx/PYNQ_Peripherals.git

$ pynq get-notebooks pynq_peripherals -p $PYNQ_JUPYTER_NOTEBOOKS

- To learn more:

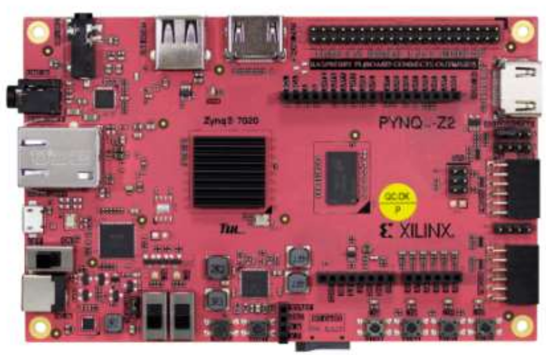
https://github.com/Xilinx/PYNQ_Peripherals.git

1. Install PYNQ_peripherals

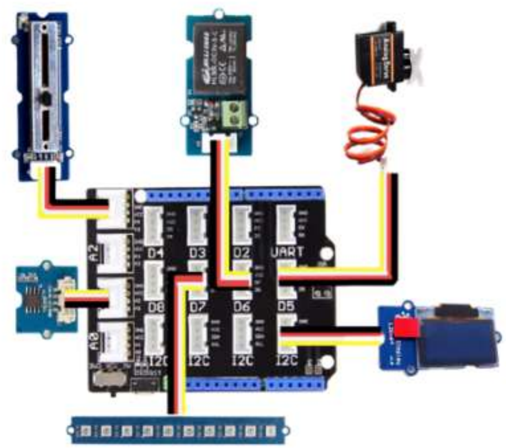
README.md

PYNQ Peripherals

This repository contains drivers for Arduino, PMOD, Grove and Raspberry Pi peripherals supported on the recommended **PYNQ-Z2** board. Each peripheral driver comes with Jupyter notebooks which show how to use it. Application notebooks can be developed solely in Python using the Python API provided for each driver. The repository also contains a command line interface tool to generate peripheral driver templates for new peripherals. Peripheral drivers are developed on-board in Plain C and the tools will export the Python API for the driver. Please refer to CONTRIBUTING.md for details on how to develop new peripheral drivers.



Recommended Board



Grove Base Arduino Shield and Grove Modules

2. Study sensor's characteristics

- Study about sensor's characteristics:

Grove Modules

Name	Link to notebook	Link to module
Grove I2C ADC	Notebook	Click Here
Grove I2C Barometer	Notebook	Click Here
Grove buzzer	Notebook	Click Here
Grove I2C Gesture	Notebook	Click Here

LEARN AND DOCUMENTS

Documentations

Resources

[Attachment] I2C ADC Eagle File

[Attachment] ADC121C021 Datasheet

TEXAS INSTRUMENTS

ADC121C021, ADC121C021Q, ADC121C027

ADC121C021/ADC121C021Q/ADC121C027 I²C-Compatible, 12-Bit Analog-to-Digital Converter with Alert Function

Check for Samples: ADC121C021, ADC121C021Q, ADC121C027

FEATURES

- I²C-Compatible 2-Wire Interface Which Supports Standard (100kHz), Fast (400kHz), and High Speed (3.4MHz) Modes
- Extended Power Supply Range (+2.7V to +5.5V)
- Up to Nine Pin-Selectable Chip Addresses (VSSOP Only)
- Out-of-Range Alert Function
- Automatic Power-Down Mode while Not Converting
- Very Small 6-Pin SOT and 8-Pin VSSOP Packages
- ADC121C021Q is an Automotive Grade Product that is AEC-Q100 Grade 2 Qualified

APPLICATIONS

- System Monitoring
- Peak Detection
- Portable Instruments
- Medical Instruments
- Test Equipment
- Automotive

KEY SPECIFICATIONS

- Resolution: 12 Bits (No Missing Codes)
- Conversion Time: 1µs (Typ)
- INL & DNL: ±1 LSB (Max) (Up to 22ksps)
- Throughput Rate: 188.9 ksps (Max)
- Power Consumption (at 22 ksps)
  - 3V Supply: 0.26 mW (Typ)
  - 5v Supply: 0.78 mW (Typ)

DESCRIPTION

These converters are low-power, monolithic, 12-bit, analog-to-digital converters (ADCs) that operates from a +2.7 to 5.5V supply. The converter is based upon a successive approximation register architecture with an internal track-and-hold circuit that can handle input frequencies up to 11MHz. These converters operate from a single supply which also serves as the reference. The device features an I²C-compatible serial interface that operates in all three speed modes, including high speed mode (3.4MHz).

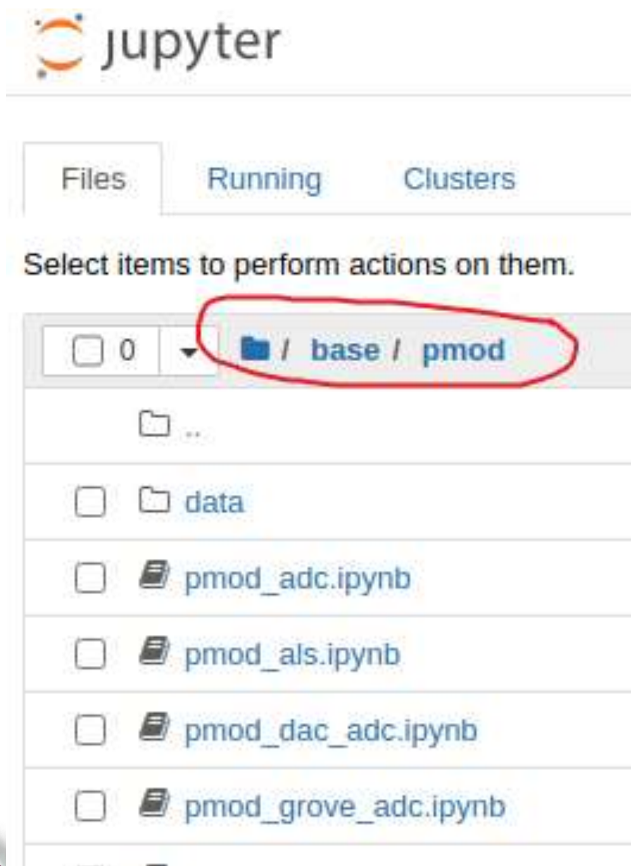
The ADC121C021's Alert feature provides an interrupt that is activated when the analog input violates a programmable upper or lower limit value. The device features an automatic conversion mode, which frees up the controller and I²C interface. In this mode, the ADC continuously monitors the analog input for an "out-of-range" condition and provides an interrupt if the measured voltage goes out-of-range.

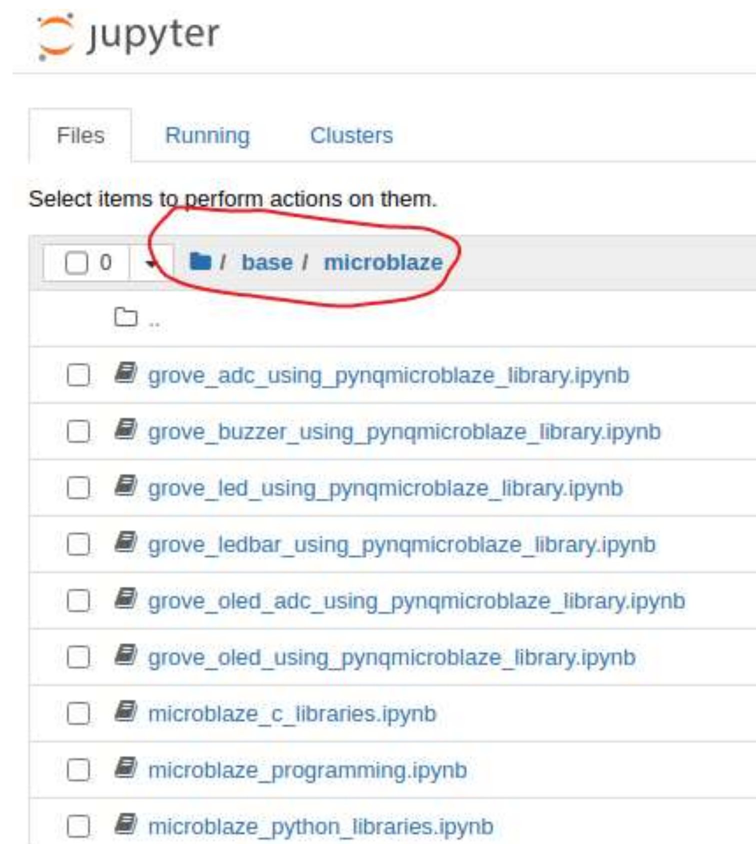
The ADC121C021 comes in two packages: a small 6-pin SOT package with an alert output, and an 8-pin VSSOP package with an alert output and two address selection inputs. The ADC121C021Q is available in a 6-pin SOT package. The ADC121C027 comes in a small 6-pin SOT package with an address selection input. The ADC121C027 provides three pin-selectable addresses while the 8-pin VSSOP version of the ADC121C021 provides nine pin-selectable addresses. Pin-compatible alternatives to the 6-pin SOT options are available with additional address options.

Normal power consumption using a +3V or +5V supply is 0.26mW or 0.78mW, respectively. The automatic power-down feature reduces the power consumption to less than 1µW while not converting. Operation over the industrial temperature range of −40°C to +105°C is ensured. Their low power consumption and small packages make this family of ADCs an excellent choice for use in battery operated

3. Examine example projects

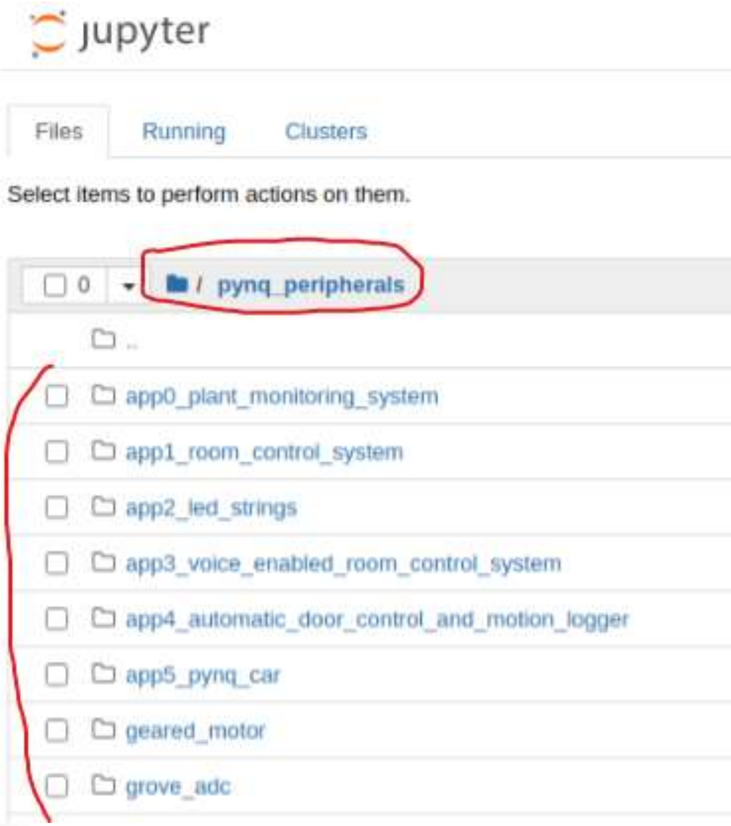
- Study from example notebooks (1)

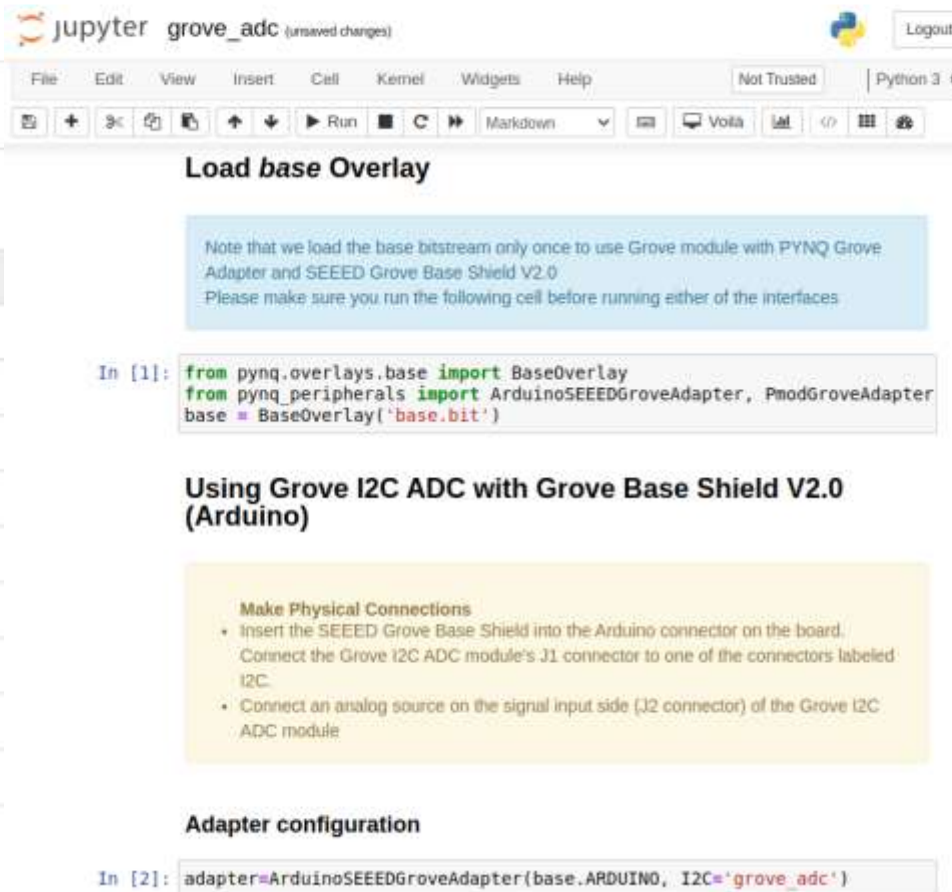




3. Examine example projects

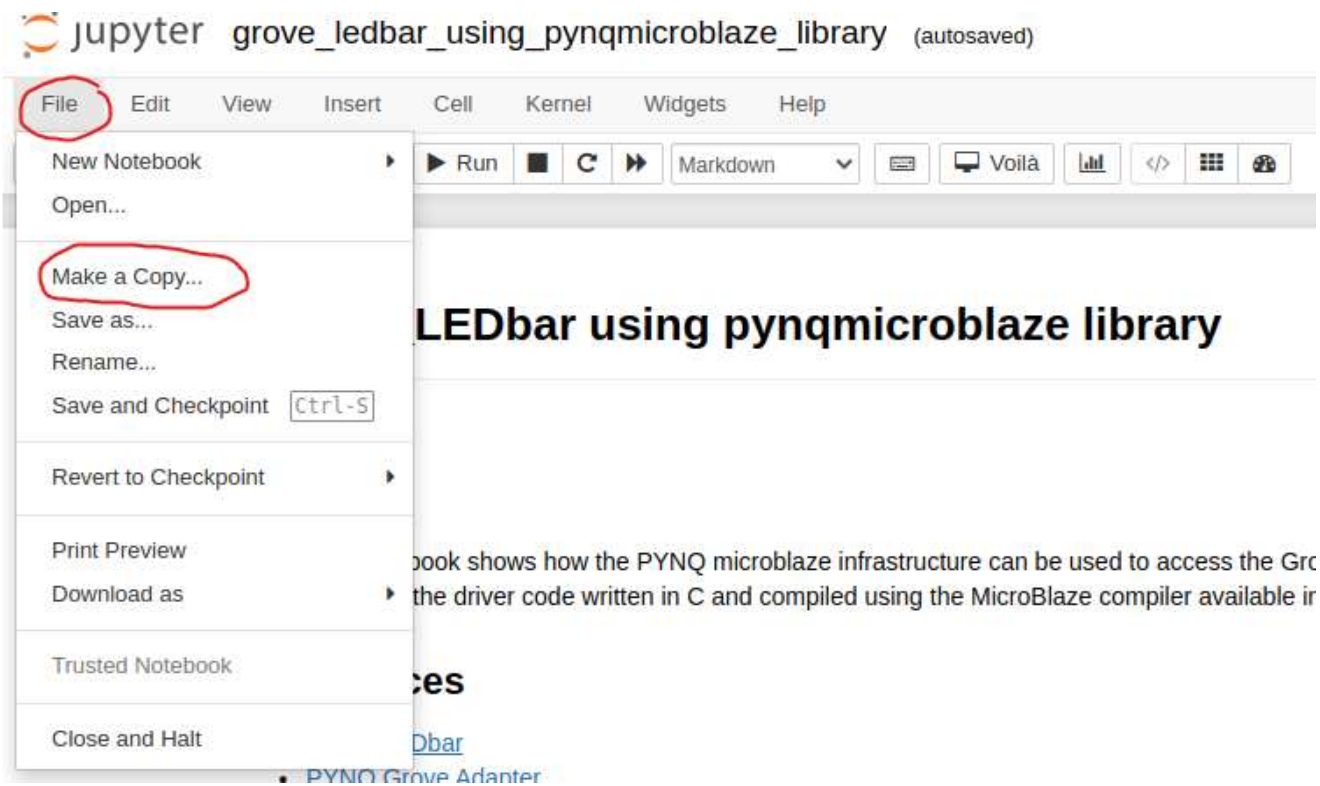
- Study from example notebooks (2)





3. Examine example projects

- DON'T MAKE CHANGES IN EXAMPLES.
- MAKE A COPY!!!



4. GUI control on Jupyter Lab

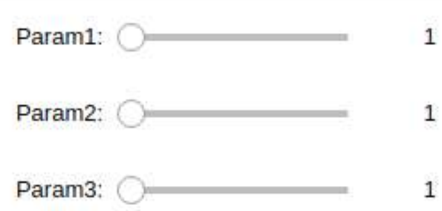
jupyter PMOD-Example_Ledbar-and-GUI-control Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [ ]: from pynq.lib import MicroblazeLibrary
        from pynq.overlays.base import BaseOverlay
        base = BaseOverlay('base.bit')

In [ ]: lib = MicroblazeLibrary(base.PMODB, ['grove_ledbar', 'pmod_grove'])

In [ ]: ledbar = lib.grove_ledbar_init_pins(lib.PMOD_G1_A, lib.PMOD_G1_B)

In [ ]: ledbar.set_level(7, 200, 1)

In [ ]: import ipywidgets as widgets

        params = [0,0,0]

        slider1 = widgets.IntSlider(
            value=1,
            min=1,
            max=255,
            step=1,
            description='Param1:',
            disabled=False,
```

4. GUI control on Jupyter Lab

```python
def set_ledbar(params):
    ledbar.set_level(params[0], params[1], params[2])

def on_change1(change):
    global params
    params[0] = change['new']
    set_ledbar(params)

def on_change2(change):
    global params
    params[1] = chanipywidgetsge['new']
    set_ledbar(params)

def on_change3(change):
    global params
    params[2] = change['new']
    set_ledbar(params)

slider1.observe(on_change1, names='value')
slider2.observe(on_change2, names='value')
slider3.observe(on_change3, names='value')

display(slider1, slider2, slider3)
```



Students can download this example on ulearn

Learn more: https://ipywidgets.readthedocs.io/en/latest/