EMBEDDED GPU LAB MANUAL

NOTE: Common Requirements for all the Experiments include Jetson Nano CPU(GPU), Display Monitor, Mouse, Keyboard and Power Adapter of Jetson Nano.

Pre-Checks:

1. The Nvidia Jetson Nano GPU is to be pre-loaded with Nvidia Ubuntu 18.04 LTS Operating System.
2. Hereby, All the Experiments are performed on real time Nvidia Ubuntu 18.04 LTS Operating System.
3. Make sure the short pins on Jetson Nano GPU are shunt using Standard Header Pin Jumper Cap. Without this, the Jetson Nano will not be able to work.
4. For further information, please refer to https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-2gb-devkit#prepare

EXPERIMENT 1: GPIO programming (LED Blinking) on Jetson Nano.

AIM: To Interface LED with Jetson Nano and Control it using GPIO Programming.

Requirements: Common Requirements, LEDs, *Jetson.GPIO* (Library from pip3)

CODE:

```
import Jetson.GPIO as GPIO
import time
led_pin = 7
GPIO.setmode(GPIO.BOARD)
GPIO.setup(led_pin, GPIO.OUT, initial=GPIO.HIGH)

while True:
  time.sleep(2)
  GPIO.output(led_pin, GPIO.HIGH)
  print("LED is ON")
  time.sleep(2)
  GPIO.output(led_pin, GPIO.LOW)
  print("LED is OFF")
```

NOTE: Connect LED to Pin 7 on Jetson Nano Board.

Result: You can See LED Blinking.

Terminal OUTPUT:

EXPERIMENT 2: Interfacing sensors and actuators to Jetson Nano.

AIM: To Interfacing MPU6050 to Jetson Nano using *SMBUS* library.

Requirements: Common Requirements, MPU6050, *SMBUS* (Library from python3)

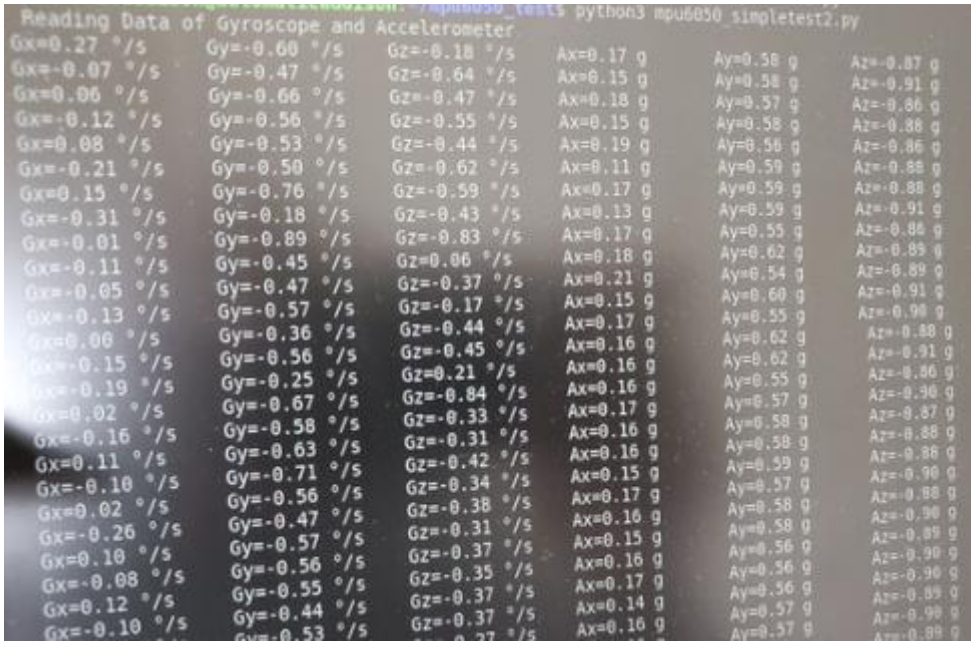
CODE:

```
import smbus
from time import sleep

#some MPU6050 Registers and their Address
PWR_MGMT_1   = 0x6B
SMPLRT_DIV   = 0x19
CONFIG       = 0x1A
GYRO_CONFIG  = 0x1B
INT_ENABLE   = 0x38
ACCEL_XOUT_H = 0x3B
ACCEL_YOUT_H = 0x3D
ACCEL_ZOUT_H = 0x3F
GYRO_XOUT_H  = 0x43
GYRO_YOUT_H  = 0x45
GYRO_ZOUT_H  = 0x47


def MPU_Init():
    #write to sample rate register
    bus.write_byte_data(Device_Address, SMPLRT_DIV, 7)

    #Write to power management register
    bus.write_byte_data(Device_Address, PWR_MGMT_1, 1)

    #Write to Configuration register
    bus.write_byte_data(Device_Address, CONFIG, 0)

    #Write to Gyro configuration register
    bus.write_byte_data(Device_Address, GYRO_CONFIG, 24)

    #Write to interrupt enable register
    bus.write_byte_data(Device_Address, INT_ENABLE, 1)

def read_raw_data(addr):
    #Accelero and Gyro value are 16-bit
        high = bus.read_byte_data(Device_Address, addr)
        low = bus.read_byte_data(Device_Address, addr+1)

        #concatenate higher and lower value
        value = ((high << 8) | low)

        #to get signed value from mpu6050
        if(value > 32768):
```

```
                value = value - 65536
        return value


bus = smbus.SMBus(1)    # or bus = smbus.SMBus(0) for older version boards
Device_Address = 0x68    # MPU6050 device address

MPU_Init()

print (" Reading Data of Gyroscope and Accelerometer")

while True:

    #Read Accelerometer raw value
    acc_x = read_raw_data(ACCEL_XOUT_H)
    acc_y = read_raw_data(ACCEL_YOUT_H)
    acc_z = read_raw_data(ACCEL_ZOUT_H)

    #Read Gyroscope raw value
    gyro_x = read_raw_data(GYRO_XOUT_H)
    gyro_y = read_raw_data(GYRO_YOUT_H)
    gyro_z = read_raw_data(GYRO_ZOUT_H)

    #Full scale range +/- 250 degree/C as per sensitivity scale factor
    Ax = acc_x/16384.0
    Ay = acc_y/16384.0
    Az = acc_z/16384.0

    Gx = gyro_x/131.0
    Gy = gyro_y/131.0
    Gz = gyro_z/131.0


    print ("Gx=%.2f" %Gx, u'\u00b0'+ "/s", "\tGy=%.2f" %Gy, u'\u00b0'+
"/s", "\tGz=%.2f" %Gz, u'\u00b0'+ "/s", "\tAx=%.2f g" %Ax, "\tAy=%.2f g"
%Ay, "\tAz=%.2f g" %Az)
    sleep(1)
```

NOTE: Connect VCC to pin 17, GND to pin GND, SCL to pin 5, SDA to pin 3 on Jetson Nano Board.

Terminal OUTPUT:



EXPERIMENT 3: Interfacing camera and modules with Jetson Nano.

AIM: To Interfacing camera and modules with Jetson Nano.

Requirements: Common Requirements, Camera, *Opencv2* (Python3 Library)

CODE:

```
import numpy as np
import cv2


cap = cv2.VideoCapture(0)
while(cap.isOpened()):

    while True:

        ret, img = cap.read()
        cv2.imshow('img', img)
        if cv2.waitKey(30) & 0xff == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
else:
    print("Alert ! Camera disconnected")
```

NOTE: Connect the USB Camera to Jetson Nano Board.

Terminal Output: One Can observe Live Video Steam.

EXPERIMENT 4: To perform data classification using Jetson Nano.

AIM: To perform data classification using Jetson Nano.

TO Be Updated!!!

EXPERIMENT 5: Write a CUDA program to demonstrate squaring an array using CUDA kernel.

AIM: To Write a CUDA program to demonstrate squaring an array using CUDA kernel.

Requirements: Common Requirements.

CODE:

```
#include <iostream>
#include <numeric>
#include <stdlib.h>
#include <cuda.h>

const int N = 128;

_global_ void f(int *dev_a) {
   unsigned int tid = threadIdx.x;

   if(tid < N) {
      dev_a[tid] = tid * tid;
   }
}

int main(void) {

   int host_a[N];
   int *dev_a;
   cudaMalloc((void**)&dev_a, N * sizeof(int));
   for(int i = 0 ; i < N ; i++) {
      host_a[i] = i*i;
   }
   cudaMemcpy(dev_a, host_a, N * sizeof(int), cudaMemcpyHostToDevice);
   f<<<1, N>>>(dev_a);
```

```
    cudaMemcpy(host_a, dev_a, N * sizeof(int), cudaMemcpyDeviceToHost);

    for(int i = 0 ; i < N ; i++) {
        printf("%d ", host_a[i]);
    }
}
```

Terminal OUTPUT:

```
#include <iostream>
#include <numeric>
#include <stdlib.h>
#include <cuda.h>

const int N = 128;

__global__ void f(int *dev_a) {
    unsigned int tid = threadIdx.x;

    if(tid < N) {
        dev_a[tid] = tid * tid;
    }
}

int main(void) {

    int host_a[N];
    int *dev_a;
    cudaMalloc((void**)&dev_a, N * sizeof(int));
    for(int i = 0 ; i < N ; i++) {
        host_a[i] = i*i;
    }
    cudaMemcpy(dev_a, host_a, N * sizeof(int), cudaMemcpyHostToDevice);
    f<<<1, N>>>(dev_a);

    cudaMemcpy(host_a, dev_a, N * sizeof(int), cudaMemcpyDeviceToHost);

    for(int i = 0 ; i < N ; i++) {
        printf("%d ", host_a[i]);
    }
}
```

```
0 1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400 441 484 529 576 625 676 729 784 841 900 961 1024 1089 1156 1225 1296 1369 1444 1521 1600 1681 17
```