**Assignment IV**

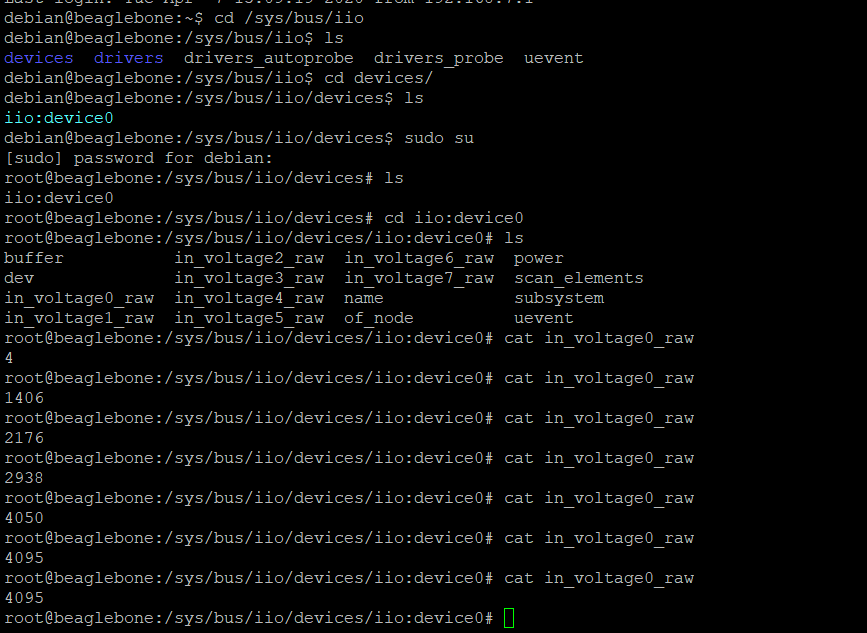
**Title : ADC**

การทำ Enable ADC ผ่าน Linux Command

- เข้าไปที่ sys/bus/iio โดยใช้คำสั่ง *cd*

- แล้วเรียกดูว่ามี Directory อะไรให้ใช้บ้าง โดยใช้คำสั่ง จากภาพจะพบว่ามี Directory ให้เราใช้

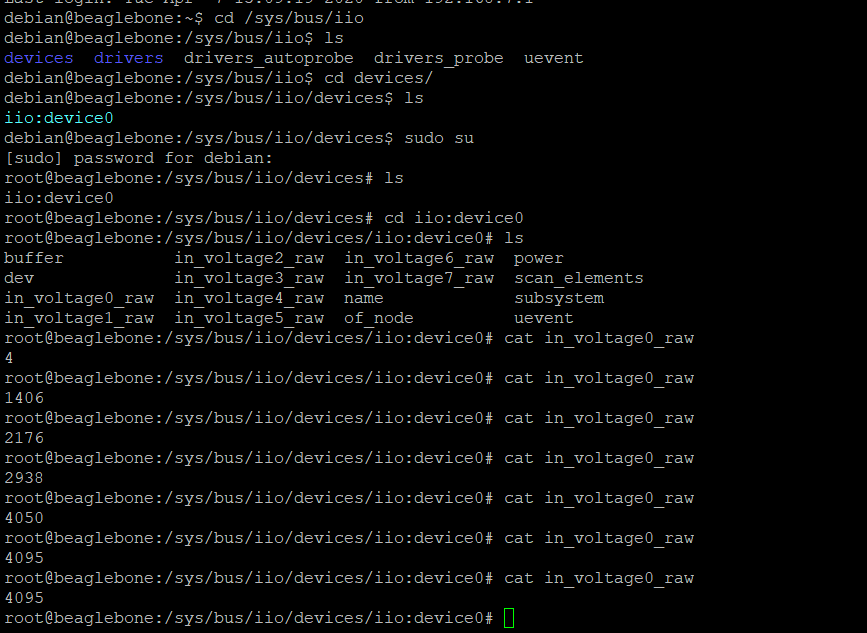
- ในขั้นตอนต่อมา ให้เราเข้าไปที่ Directory ที่ชื่อว่า **devices** โดยใชคำสั่ง *cd* ทำการเช็คว่าภายในมีอะไรให้เราใช้งานบ้างโดยใช้คำสั่ง *ls* เช่นเดิม ดังที่แสดง



*(Figure 1)*

- พิมพ์คำสั่ง *sudo su* เพื่อเข้าสู่ root แล้วทำการเช็คว่ามี **device0** แล้วหรือยัง โดยใช้คำสั่ง *ls*

- ต่อมาเราจะใช้งาน **ADC channel 0** จึงใช้ in\_voltage0\_raw เมื่อเจอแล้วให้ใช้ตัวแปลง ADC ได้โดยใช้คำสั่ง *cat* ถ้าต้องการเปลี่ยน channel สามารถทำได้โดยเปลี่ยนจาก 0 เป็นเลขใดก็ได้แต่ต้องอยู่ในช่วง 0 - 7

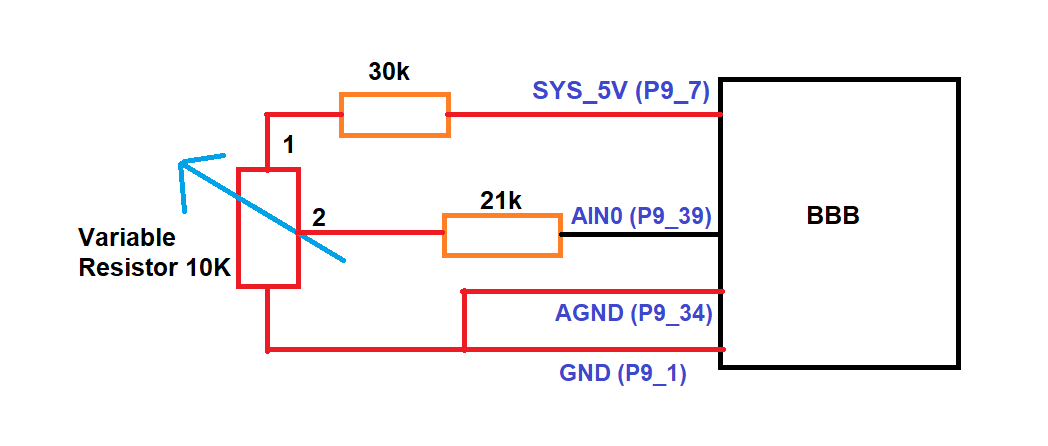


*(Figure 2)*

**Exercise 13**

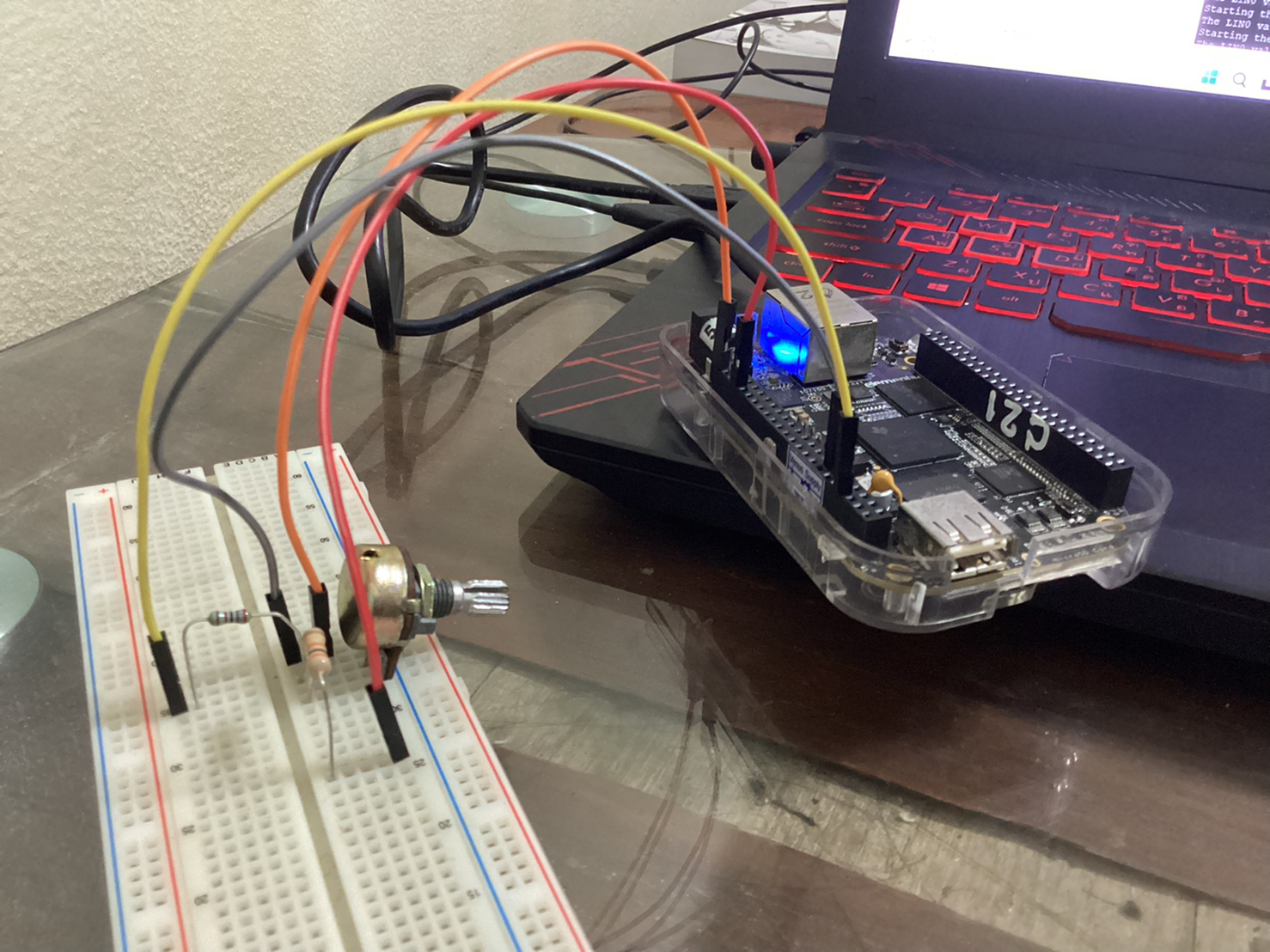
**งานที่ได้รับมอบหมาย :** 1. ให้ไปศึกษาด้วยตนเอง รายละเอียดคำสั่งทุกบรรทัด ทดสอบการทำงานของ ADC และทำการบันทึกภาพการแสดงค่าที่อ่านได้

**Schematic**



*(Figure 3)*

**Circuit**



*(Figure 4)*

คำสั่งที่ใช้งาน

#include<iostream>

#include<fstream>

#include<string>

#include<sstream> //ดึง Library มาใช้

using namespace std;

#define LIN0\_PATH "/sys/bus/iio/devices/iio:device0/in\_voltage" //กำหนด path

int readAnalog(int number){ //ฟังก์ชันสำหรับอ่านค่า ADC

stringstream ss; //ประกาศตัวแปรเก็บค่า SS

ss << LIN0\_PATH << number << "\_raw"; //ดึงค่าจาก LIN0\_PATH มายัง ss

fstream fs; //ประกาศตัวแปร fs

fs.open(ss.str().c\_str(), fstream::in); //เปิดไฟล์จากตัวแปร ss แบบอ่าน

fs >> number; //นำค่าจากตัวแปร fs ไปเก็บที่ number

fs.close(); //ปิดไฟล์

return number; //return ค่า

}

int main(int argc, char\* argv[ ]){

cout << "Starting the readLIN0 program" << endl; //แสดงข้อความ

int value = readAnalog(0); //อ่านค่าจาก ADC0 แล้วเก็บไว้ที่ตัวแปร value

cout << "The LIN0 value was " << value << " out of 4095." << endl; //แสดงข้อความและค่าที่ตัวแปร value บันทึกมาจาก readAnalog(0)

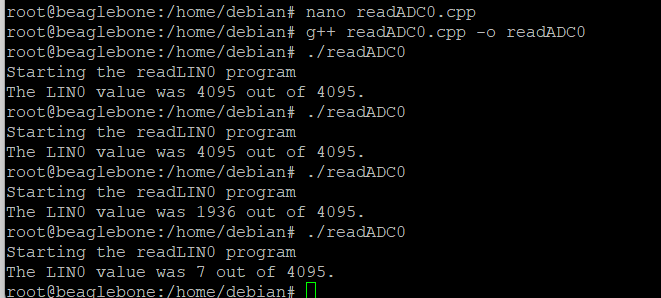
return 0; // return ค่า

}

- หลังจากพิมพ์คำสั่งโดยใช้คำสั่งสร้างไฟล์ nano แล้ว

- ให้เราทำการ Compile โดยใช้คำสั่ง *g++ readADC0.cpp -o readADC0*

- เมื่อเสร็จแล้วให้เราทำการรันโปรแกรม



*(Figure 5)*

จากภาพข้างต้น เนื่องจากเราไม่ได้ใช้เพิ่มคำสั่งให้ทำการวนลูปหรือทำการแสดงผลซ้ำ ๆ ไม่รู้จบ เราต้องทำการพิมพ์คำสั่ง *./readADC0* เองทุกครั้ง (Manual)เมื่อต้องการทราบค่าที่แปลงมาจาก ADC0

เมื่อเราทำการปรับค่าความต้านทานจากตัวต้านทานปรับค่าได้ เมื่อเราปรับในทิศทวนเข็มนาฬิกา (ค่าความต้านทานลดลง) จะทำให้ค่าที่แปลงมาจาก ADC0 มีค่าเพิ่มขึ้น เราสามารถคำนวณได้ว่า ค่าที่แปลงมานั้นเมื่อเทียบกับค่าในสัญญาณอนาล็อกมีค่าเท่าใดโดยใช้วิธีคำนวณดังนี้

โดยที่ แรงดันที่บอร์ดจ่ายให้วงจรคือ 1.8 V () และจำนวนบิตที่ ADC แปลงได้คือ 12 หรือก็คือ 2e12 หรือ 4096

ตัวอย่าง ADC แปลงสัญญาณอนาล็อกมาเป็นสัญญาณดิจิตอลได้เป็น 1936 ดังนั้น แรงดันที่แปลงมา คือ

**งานที่ได้รับมอบหมาย :** 2. ดัดแปลงโปรแกรมให้แสดง-อ่านค่า ADC ได้ต่อเนื่องไม่รู้จบ

- เพิ่ม library : **unistd.h** สำหรับใช้คำสั่ง *usleep()* เพื่อทำการ Delay การแสดงผลของคำสั่ง

- เพิ่มคำสั่ง *while (1)* เพื่อให้โปรแกรมวนลูป อ่านค่าได้ต่อเนื่องไม่รู้จบ

- เพิ่มคำสั่ง *usleep(1500000)* เพื่อให้โปรแกรม Delay 1.5 sec

คำสั่งที่ใช้งาน

#include<iostream>

#include<fstream>

#include<string>

#include<sstream>

#include<unistd.h>

using namespace std;

#define LIN0\_PATH "/sys/bus/iio/devices/iio:device0/in\_voltage"

int readAnalog(int number){

stringstream ss;

ss << LIN0\_PATH << number << "\_raw";

fstream fs;

fs.open(ss.str().c\_str(), fstream::in);

fs >> number;

fs.close();

return number;

}

int main(int argc, char\* argv[ ]){

while(1){

cout << "Starting the readLIN0 program" << endl;

int value = readAnalog(0);

cout << "The LIN0 value was " << value << " out of 4095." << endl;

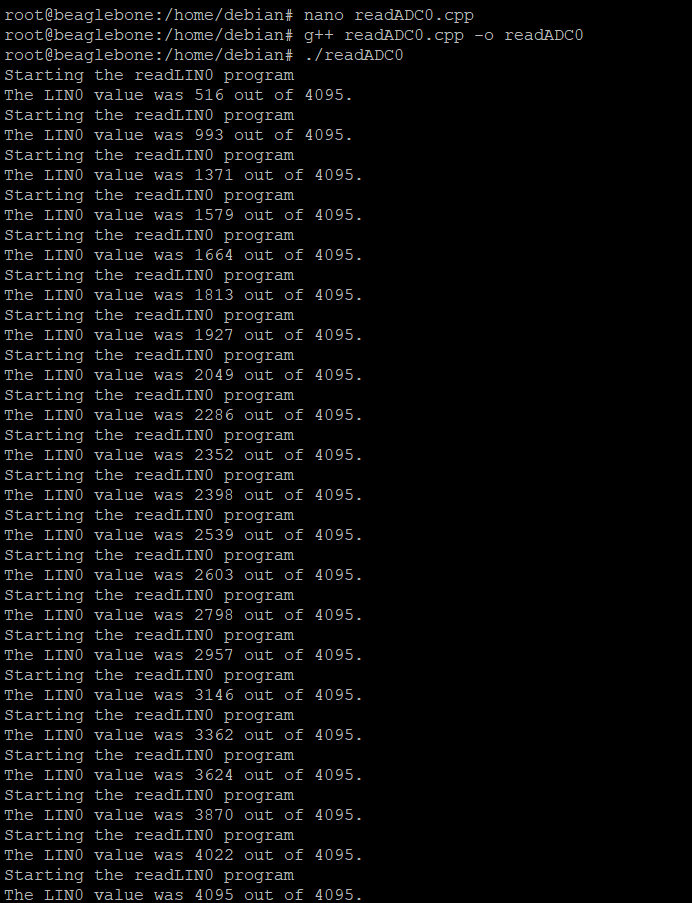
usleep(1500000);

}

return 0;

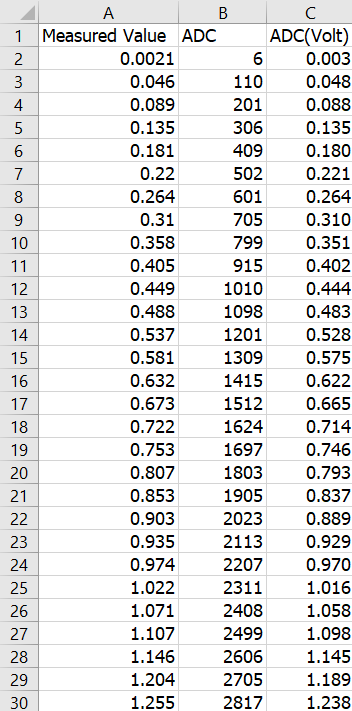
}

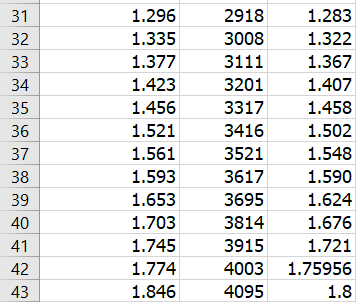
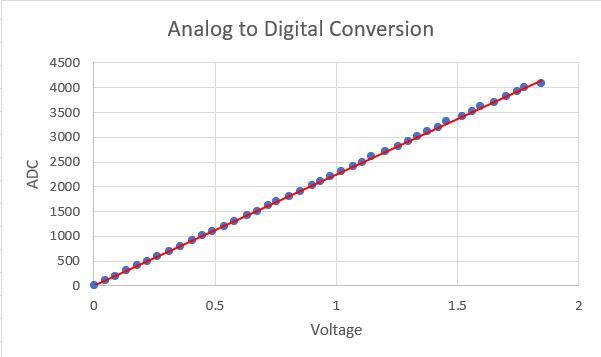
**การแสดงผล**



*(Figure 6)*

จากภาพจะเห็นว่า คำสั่งจะทำการรันเองอัตโนมัติไม่รู้จบ จนกว่าเราจะกดปุ่ม ctrl + C โปรแกรมถึงจะหยุดทำงาน ทำให้ง่ายต่อการอ่านค่า ADC ที่บอร์ดแปลงมา โดยที่ค่า R ยิ่งเพิ่มมากขึ้นเท่าไหร่ ค่าที่แปลงมาจาก ADC จะยิ่งลดลงเท่านั้น ซึ่งเป็นไปตามกฎของโอห์ม (Ohm’s law)

**ผลการทดสอบจากห้องปฏิบัติการและกราฟแสดงแนวโน้ม**



Table

Graph