**TRƯỜNG ĐẠI HỌC QUỐC GIA TPHCM**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**BỘ MÔN ĐIỆN – ĐIỆN TỬ**

🙡∙•∙🙣



**BÁO CÁO NGHIÊN CỨU**

**Đề tài:**

**MẠCH HẠ ÁP BUCK**

**GVHD : Hồ Thanh Phương**

**SV : Nguyễn Khôi Nguyên**

**MSSV : 1612287**

**Lớp : DD16LT09**

MỤC LỤC

[Lời cảm ơn 3](#_Toc514366422)

[MỞ ĐẦU 4](#_Toc514366423)

[CHƯƠNG 1: TỔNG QUAN VỀ BỘ BIẾN ĐỔI BUCK 5](#_Toc514366424)

[1.1. Mục đích 5](#_Toc514366425)

[1.1.1. Nguyên lý hoạt động 5](#_Toc514366426)

[1.1.2. Mô hình bộ biến đổi 10](#_Toc514366427)

[CHƯƠNG 2: THIẾT KẾ BỘ BUCK 11](#_Toc514366428)

[2.1. Thiết kế Schematic 11](#_Toc514366429)

[2.1.1. Lựa chọn linh kiện 11](#_Toc514366430)

[2.1.2. Sơ đồ Schematic 11](#_Toc514366431)

[2.2. Thiết kế PCB 13](#_Toc514366432)

[2.2.1. Một số nguyên tắc cần lưu ý 13](#_Toc514366433)

[2.2.2. Sơ đồ PCB 15](#_Toc514366434)

[2.3. Firmware và thiết kế bộ điều khiển PID 15](#_Toc514366435)

[2.3.1. Tổng quan về firmware 15](#_Toc514366436)

[2.3.2. Giao tiếp với MSP430 qua ngôn ngữ C 16](#_Toc514366437)

[2.3.3. Thiết kế bộ điều khiển PID 36](#_Toc514366438)

[CHƯƠNG 3: THÍ NGHIỆM VÀ ĐÁNH GIÁ 39](#_Toc514366439)

[3.1. Thí nghiệm đáp ứng ngỏ ra 39](#_Toc514366440)

[3.1.1. Mô hình mạch BUCK hoàn chỉnh 39](#_Toc514366441)

[3.1.2. Kết quả thí nghiệm 39](#_Toc514366442)

[3.2. Đánh giá và nhận xét 44](#_Toc514366443)

[3.2.1. Tính toán độ nhấp nhô áp ra 44](#_Toc514366444)

[3.2.2. Nhận xét ưu, nhược điểm và biện pháp khắc phục 44](#_Toc514366445)

[KẾT LUẬN 45](#_Toc514366446)

[TÀI LIỆU THAM KHẢO 46](#_Toc514366447)

# Lời cảm ơn

**🙢🏵🙠**

Em xin gửi lời cảm ơn sâu sắc đối với các thầy, cô và các bạn tại trường đã tạo điều kiện, giúp đỡ em trong suốt quá trình của học môn .

Đặc biệt, em xin chân thành cảm ơn thầy Hồ Thanh Phương đã tận tâm hướng dẫn chúng em qua từng buổi học trên lớp. Nếu không có những lời hướng dẫn, dạy bảo của thầy thì em nghĩ bài thu hoạch này của em rất khó có thể hoàn thiện được.

Trong quá trình làm bài báo cáo thực tập, khó tránh khỏi sai sót, rất mong các thầy, cô bỏ qua. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp thầy, cô để em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn bài báo cáo tốt nghiệp sắp tới.

Em xin chân thành cảm ơn!

Tp.HCM, ngày 15 tháng 5 năm 2018

# MỞ ĐẦU

Trong môi trường kỹ thuật hiện đại ngày nay, việc tạo ra các bộ nguồn có chất lượng và kích thước nhỏ gọn là một nhu cầu hết sức cần thiết.

Hiện nay bộ biến đổi DC- DC đang được sử dụng ngày càng rộng rãi, nhất là trong những thiết bị có kích thước nhỏ và cần hiệu suất cao như laptop, điện thoại thông minh, những máy y tế,… Nhu cầu biến đổi một điện thế cao sang một điện thế thấp hơn để có thể cung cấp cho nhiều thiết bị điện tử khác nhau là rất lớn. Bộ biến đổi BUCK được tạo ra để đáp ứng những nhu cầu này. Cấu trúc của mạch tuy không phức tạp nhưng cần những kỹ thuật cao về chọn linh kiện, vẽ mạch và lập trình để tạo ra một bộ nguồn ổn định và có hiệu suất cao.

Vì vậy, với đề tài “ Mạch hạ áp BUCK” sẽ đáp ứng nhu cầu cung cấp điện áp cho nhiều thiết bị có các mức điện áp khác nhau một cách thuận tiện và ổn định vì phần áp ra sẽ được điều chỉnh bằng thuật toán phần mềm, ít phụ thuộc vào phần cứng.

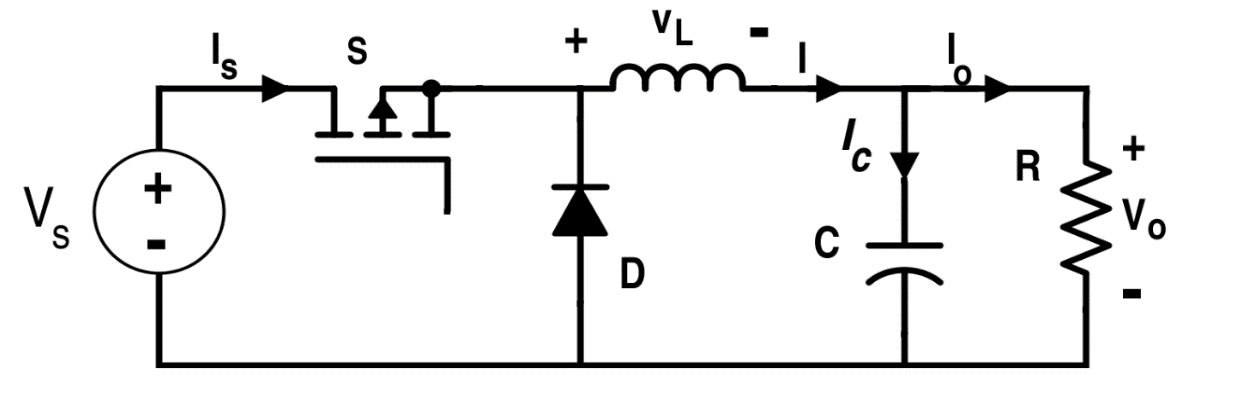
# CHƯƠNG 1: TỔNG QUAN VỀ BỘ BIẾN ĐỔI BUCK

## Mục đích

Mục đích của bộ biến đổi DC-DC là tạo ra các điện áp một chiều điều chỉnh được để cung cấp cho các linh kiện điện tử khác nhau. Trong một số trường hợp chúng ta có một nguồn áp cao và phải hạ áp xuống tương ứng với các linh kiện. Ta có hai phương thức để thực hiện, sử dụng bộ hạ áp tuyến tính bằng trở. Tuy cho điện áp khá phẳng nhưng hiệu suất rất thấp chỉ khoảng 35% và áp ra phụ thuộc hoàn toàn vào phần cứng không thể thay đổi tùy ý. Trong trường hợp cần tính hiệu quả và tùy biến cao mạch hạ áp xung hay còn gọi là BUCK được sử dụng để khắc phục hạn chế trên.

### Nguyên lý hoạt động

1. ***Sơ đồ bộ biến đổi***

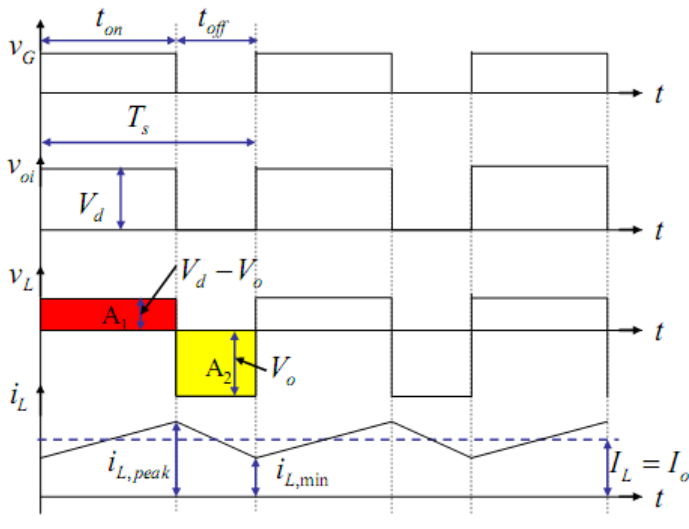


***Hình 1.1****: Sơ đồ nguyên lý bộ Buck*

Bộ Buck gồm 1 transistor loại mosfet, 1 diode, 1 cuộn dây, 1 tụ điện. Mosfet hoạt động như một cộng tắc đóng mở được điều khiển bằng một sóng vuông tại chân G. Nó làm việc ở 2 chế độ: liên tục và không liên tục.

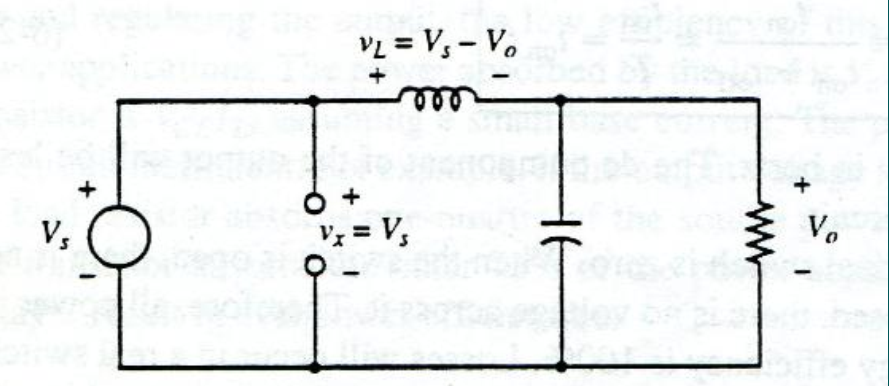
1. ***Nguyên lý làm việc ở chế độ liên tục***

Một bộ chuyển đổi buck hoạt động trong chế độ liên tục có dòng điện chạy qua cuộn dây IL không bao giờ giảm về không trong suốt chu kì chuyển mạch. Trong chế độ này, nguyên lý hoạt động mô tả trong hình 3. Gọi duty circle D = tON/T.



***Hình 1.2:*** *Sơ đồ đóng cắt mạch và dòng điện qua L*

***Xét khoảng S đóng:***

******

***Hình 1.3****: Sơ đồ tương đương khi S đóng*

Áp ra VO = Vd nên áp cuộn dây tăng và năng lượng được tích trữ trong cuộng cảm L. Áp của cuộn cảm V­L=VS – VO vì điện dung C xem như vô cùng lớn và áp ra là hằng số.

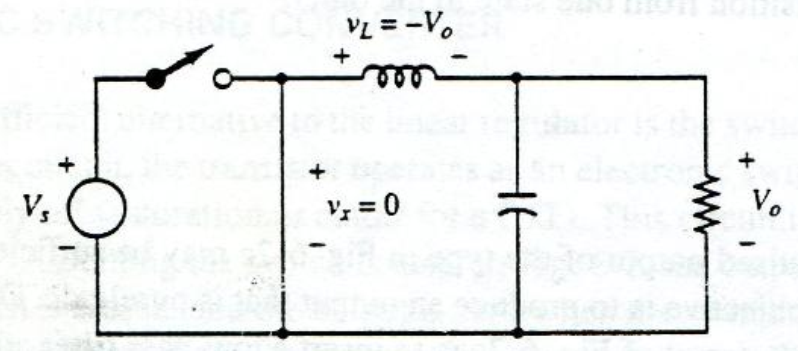
Ta có các công thức sau:

vL = L = VS + VO = L = L

t1 =

=

***Xét khoảng S ngắt:***

******

***Hình 1.4:*** *Sơ đồ tương đương khi S mở*

Áp ra Vo = -VL. Năng lượng trong cuộn cảm được xả qua C và diode được kích dẫn.

Ta có các công thức sau:

VL = L = -VO = L = -L

t2 =

=

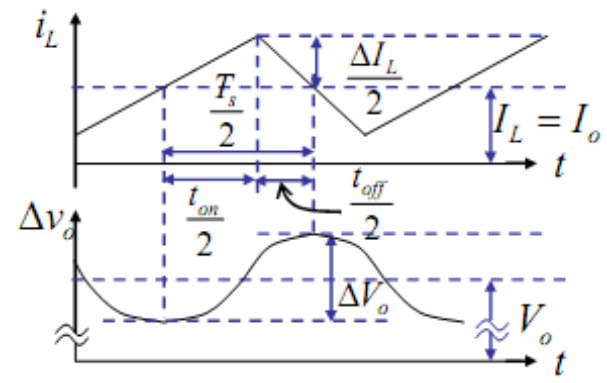
Hệ quả:

Trị trung bình áp tải:

Trị trung bình dòng điện tải: ;

Độ nhấp nhô dòng điện tải: (1)

***Ranh giới chế độ làm việc liên tục và gián đoạn:***

******

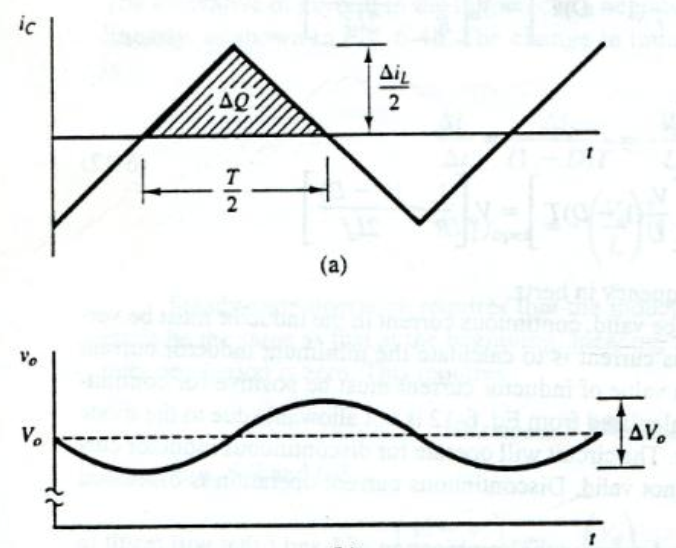
***Hình 1.5:*** *Đồ thị ranh giới chế độ liên tục và gián đoạn*

Ta có:

Giá trị Lmin để dòng IL liên tục:

(2)

***Độ nhấp nhô điện áp tải:***

******

***Hình 1.6:*** *Đồ thị nhấp nhô áp tải*

Ta có:

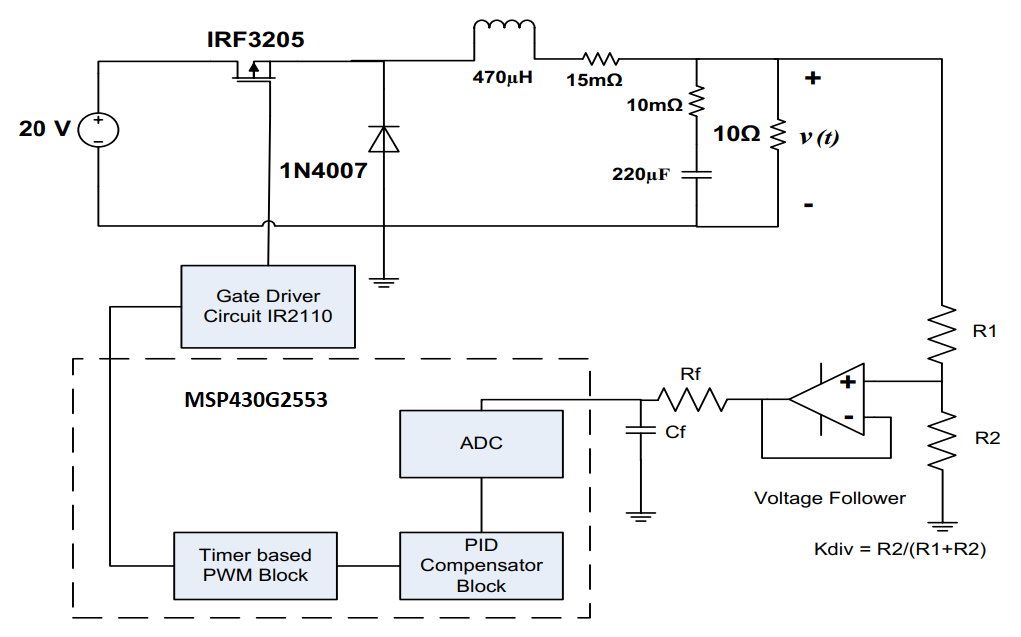
iC = iL – iR

Q = CV0

Suy ra:

(3)

### Mô hình bộ biến đổi



***Hình 1.7****: Mô hình bộ biến đổi*

# CHƯƠNG 2: THIẾT KẾ BỘ BUCK

## 2.1. Thiết kế Schematic

### 2.1.1. Lựa chọn linh kiện

* *Cuộn cảm H:*

Để dòng I liên tục với Dmin = 0.1, R=10(ohm), f = 16KHz. Theo công thức (2) ta được Lmin = 281uH. Chọn tụ có điện cảm H = 330uH.

* *Tụ điện Cout:*

Để độ nhấp nhô áp dưới 5%, Dmin=0.1. Theo công thức (3) ta được Cout = 31.26uF. Chọn tụ Cout = 100Uf

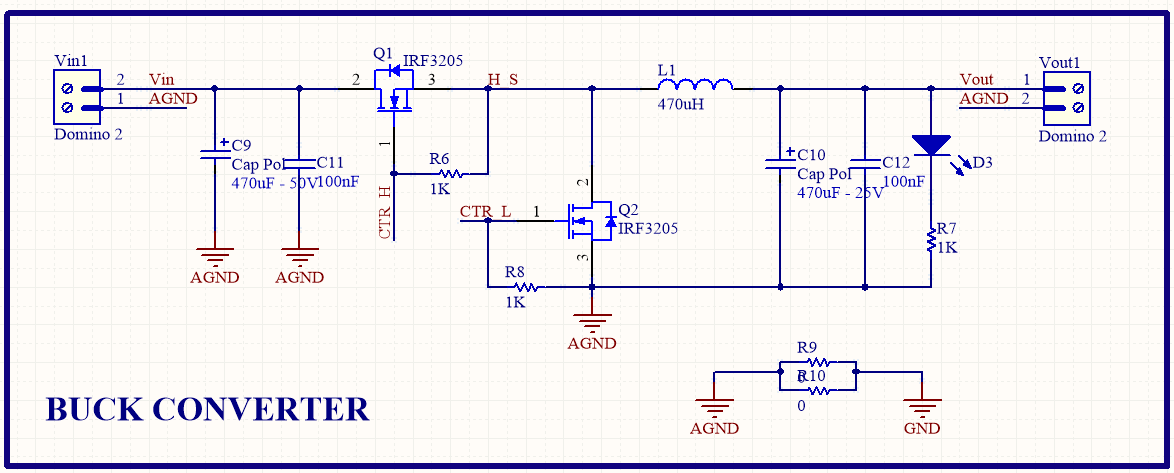
* *Mosfet IRF3205:*

Vì đây là mosfet kênh N nên cần kỹ thuật boosttrap để mạch hoạt động. Ta sử dụng thêm ic booststrap và thay diode bằng một on mosfet IRF3205

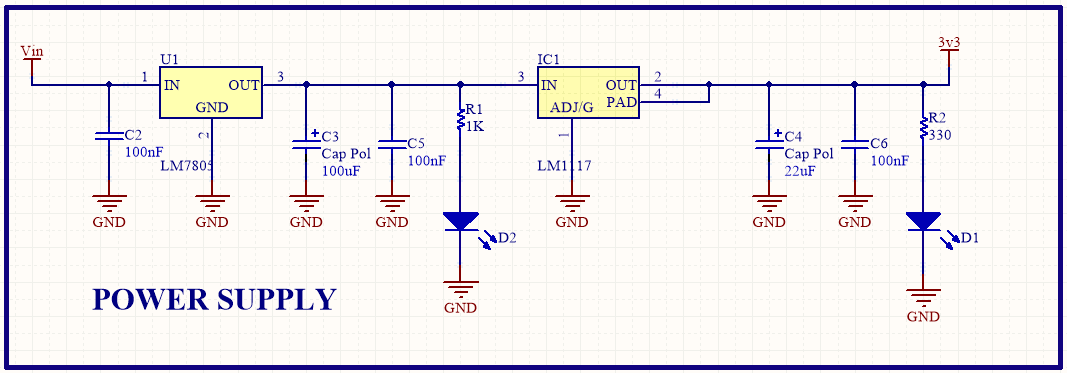
* *IC: IR2110 để tạo mạch booststrap*
* *IC nguồn: LM7805, LM1117*
* *Vi điều khiển MSP430G2553*
* *Tụ và trở như trong Schematic*

### 2.1.2. Sơ đồ Schematic

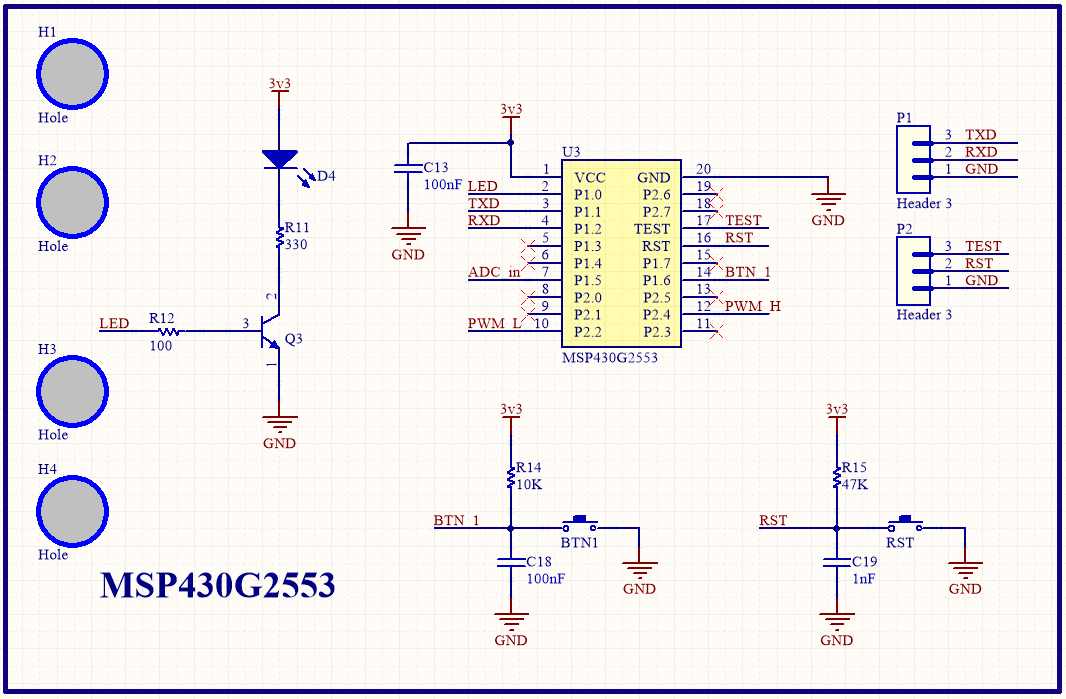
Gồm 5 khối: khối BUCK, khối nguồn tuyến tính, khối vi điều khiển, khối hồi tiếp, khối booststrap



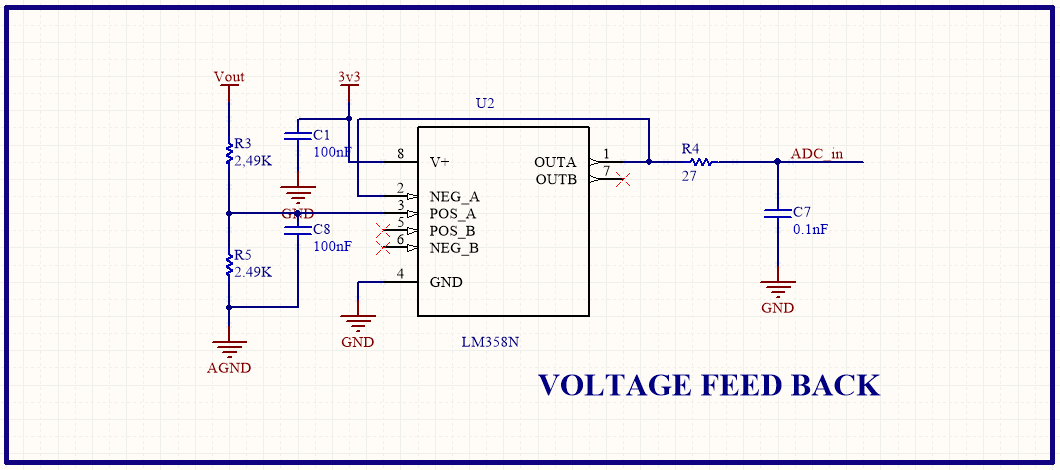
***Hình 2.1****: Sơ đồ khối BUCK*



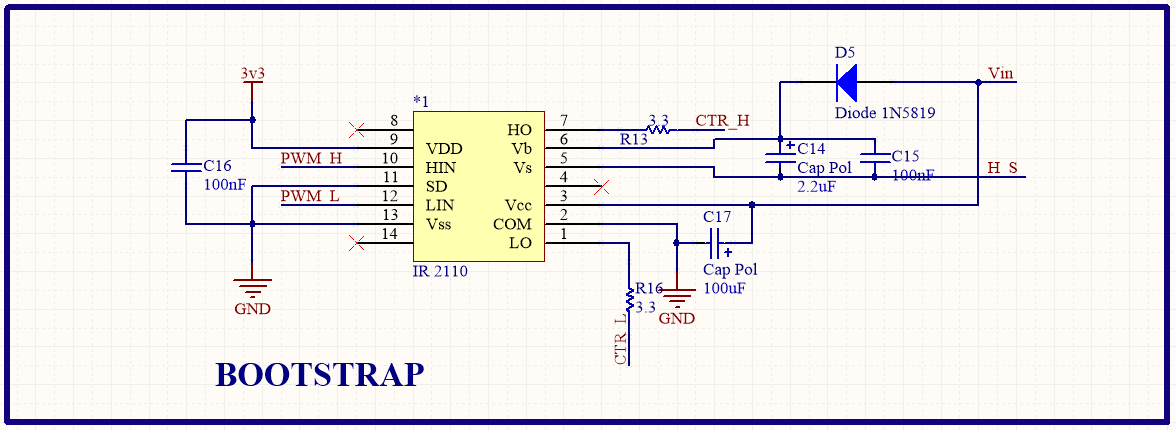
***Hình 2.2****: Sơ đồ khối mạch nguồn tuyến tính*



***Hình 2.3:*** *Sơ đồ khối vi điều khiển*



***Hình 2.4:*** *Sơ đồ khối hồi tiếp áp ra*



***Hình 2.5****: Sơ đồ khối Bootstrap*

## Thiết kế PCB

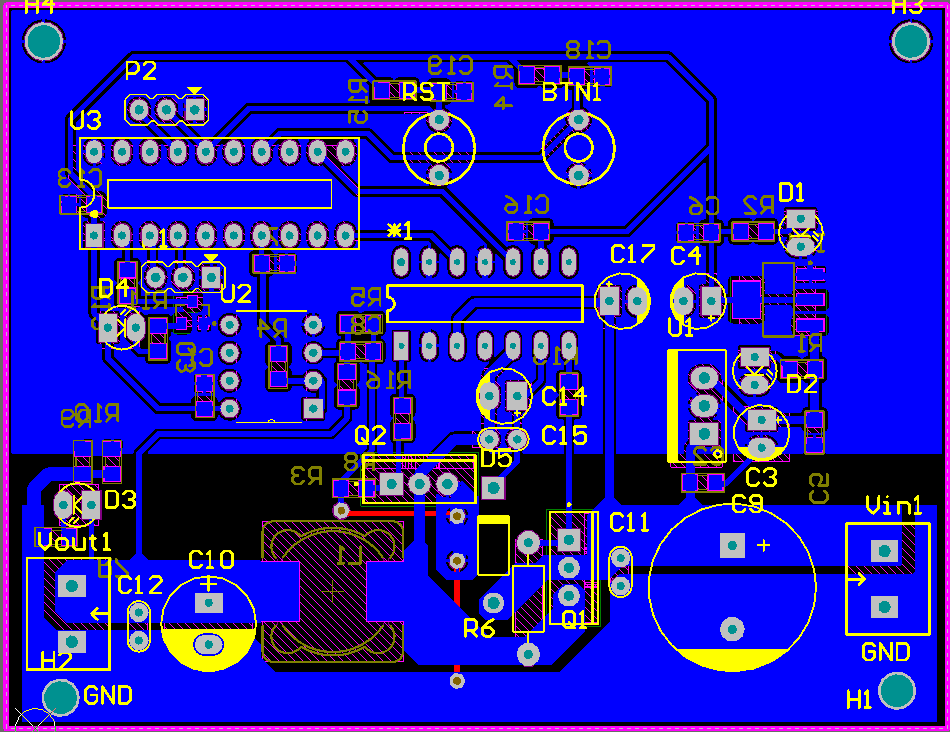
### Một số nguyên tắc cần lưu ý

* Đặt ngõ vào của tụ và diode trên cùng 1 lớp PCB, càng gần nguồn vào của IC càng tốt
* Thêm tản nhiệt nếu cần thiết
* Đặt cuộn cảm gần IC, nhưng không cần phải gần như tụ. Điều này giúp làm giảm bức xạ và mạch sẽ nhỏ hơn
* Đặt đầu ra của tụ điện gần cuộn cảm.
* Đặt dây dẫn hồi tiếp xa vùng gây nhiễu, như là cuộn cảm và diode
* Không mở rộng Board đồng nếu không cần thiết
* Không đi dây đất phía dưới cuộn cảm
* Để ý đến khoảng trống giữa cuộn cảm

**\*Feedback:**

* Dây nối giữa ngõ ra feedback và mạng dây chéo của điện trở nên ngắn nhất có thể
* Dây chia áp đặt gần và song song với feedback
* Đi dây ra xa nút điện áp của cuộn cảm và diode.
* Không được đi dây trực tiếp qua chân cuộn cảm và diode và không được song song với dây nguồn

### Sơ đồ PCB



***Hình 2.6:*** *Sơ đồ mạch PCB*

## Firmware và thiết kế bộ điều khiển PID

### Tổng quan về firmware

* + - * + Code C bao gồm file main.c và các thư viện Buck-boost.c, UART.c, ADC.c, Basic\_config.c
        + Tổng quan về code: hàm main sẽ gọi đến các thư viện để khởi động CLOCK và các ngoại vi như timerA0, timerA1, UART, ADC sau đó sẽ lập vô tận việc gửi sai số áp (so với mong muốn) và tín hiệu điều khiển PID về máy tính.
        + Timer 0 được config ở chế đệ đếm lên sau mỗi 1ms tiến hành lấy mẫu ADC và lọc trung bình kết quả đó. ADC sau khi lọc sẽ qua bộ PID tính toán output và cập nhật lại duty circle của xung PWM thông qua TA1CCR1 vá TA1CCR2.
        + Timer 1 được config ở chế độ Outmode tạo ra 2 xung PWM ngược nhau với deadtime 2us.

### Giao tiếp với MSP430 qua ngôn ngữ C

* + - * ***Hàm main***

**#include** "msp430.h"

**#include** <stdio.h>

**#include** <stdint.h>

**#include** "lib/Buck\_boost.h"

**volatile** uint16\_t duty=0;

uint16\_t voltage\_error;

uint16\_t voltage\_out;

**volatile** uint32\_t time = 0;

**int** **main**(**void**)

{

Config\_stop\_WDT();

Config\_Clocks();

GPIO\_init();

uart\_init();

Timer\_start();

ADC10\_Init(*ON2\_5V*);

PWM\_start();

**char** str[100];

//void PID\_init(&voltage\_pid, 1, 1,1,1, 1,1);

voltage\_pid.Kp = 0.01;

voltage\_pid.Ki = 0;

voltage\_pid.Kd = 0.001;

voltage\_pid.saturate\_l = 50;

voltage\_pid.saturate\_h = 450;

voltage\_pid.T = 1;

voltage\_pid.I\_part\_sat = 50;

**while**(1)

{

voltage\_error = voltage\_pid.Error;

voltage\_out = voltage\_pid.Out;

**sprintf**(str, "0 %d %d 500", (uint16\_t)voltage\_error,(uint16\_t)voltage\_out);

uart\_puts(str);

uart\_puts("\n\r");

delay\_ms(1);

}

}

* + - * **Buck\_boost.h**

**#ifndef** BUCK\_BOOST\_H\_

**#define** BUCK\_BOOST\_H\_

**#include** "UART.h"

**#include** "Basic\_config.h"

**#include** "ADC.h"

**#include** "pid.h"

**#include** <stdint.h>

//#define TIMER

**#define** TIME\_PERIOD 16 //calculate 1us

**#define** TIME\_INTERVAL 50 // in microsecond

**#define** DUTY\_CIRCLE 0.5 // 0 < DUTY\_CIRCLE < 1

**#define** PWM\_CIRCLE (uint16\_t)(TIME\_INTERVAL\*SMCLK\_)

**#define** SMCLK\_ 16

**#define** SAMPLING\_TIME (SMCLK\_\*1000) // sampling at 100us with clock at 16MHz

**#define** PIN\_IN BIT5 // read form P1.5

**#define** PIN\_OUT (BIT2 + BIT4) // output at P2.2, P2.4

**#define** LED\_RED BIT0

**#define** LED\_GREEN BIT6

**#define** BTN1 BIT3

**#define** SAMPLE\_NUM 10

**#define** VOLTAGE\_K 6.0

**#define** V\_REF 2.5

**#define** DESIRE\_VOLTAGE 5.0 //Set out put voltage

**#define** ADC\_RESOLUTION 1024.0

**#define** ADC\_SET\_POINT (DESIRE\_VOLTAGE/VOLTAGE\_K/V\_REF\*ADC\_RESOLUTION)

**volatile** uint16\_t voltage\_buf[SAMPLE\_NUM];

**volatile** pidparams voltage\_pid;

**void** **Buck\_boost\_init**(**void**);

**void** **GPIO\_init**(**void**);

**void** **Timer\_start**(**void**);

**void** **Timer\_stop**(**void**);

**void** **PWM\_start**(**void**);

**void** **PWM\_update**(uint16\_t duty);

**void** **PWM\_stop**(**void**);

**void** **PID\_update**(pidparams \*pid);

**float** **Average\_filter**(**volatile** uint16\_t \*array);

**#endif** /\* LIB\_BUCK\_BOOST\_H\_ \*/

* + - * ***Buck\_boost.c***

**#include** <msp430.h>

**#include** "Buck\_boost.h"

**int** i=0;

**volatile** uint8\_t counter=0;

**volatile** **float** voltage\_error;

**volatile** uint16\_t voltage\_out\_;

**unsigned** **int** vout;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTIONS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **GPIO\_init**(**void**)

{

P1SEL &= ~(BTN1 + LED\_GREEN + LED\_RED);

P1SEL2 &= ~(BTN1 + LED\_GREEN + LED\_RED);

//Init button

P1DIR &= ~BTN1;

P1REN |= BTN1;

P1OUT |= BTN1;

P1IE |= BTN1;

P1IES |= BTN1;

//Init LED GREEN

P1DIR |= LED\_GREEN;

P1OUT &= ~LED\_GREEN;

//Init LED RED

P1DIR |= LED\_RED;

P1OUT &= ~LED\_RED;

//Init PWM output

P2DIR |= PIN\_OUT;

P2OUT &= ~PIN\_OUT;

P2SEL|= PIN\_OUT;

//Global interupt

\_BIS\_SR(GIE);

}

**void** **Timer\_start**(**void**)

{

TA0CCR0 = SAMPLING\_TIME;

TA0CTL = TASSEL\_2 + MC\_1 + TAIE;// SMCLK, Up mode, Overflow interrupt

TA0CCTL0 &= ~CCIFG;

}

**void** **Timer\_stop**(**void**)

{

TA0CTL = 0 ;

TA0R = 0;

P1OUT &= ~LED\_GREEN;

}

**void** **PWM\_start**(**void**)

{

TA1CCR0 = 512;

TA1CCR2 = 150;

TA1CCR1 = TA1CCR2 + 20;

TA1CTL = TASSEL\_2 + MC\_3 ; // SMCLK, Up mode

TA1CCTL1 = OUTMOD\_6; // falling edge & raising edge, capture mode, capture/compare

TA1CCTL2 = OUTMOD\_2;

}

**void** **PWM\_update**(uint16\_t duty)

{

TA1CCR1 = duty + 20;

TA1CCR2 = duty;

}

**void** **PWM\_stop**(**void**)

{

TA1CCR0 = 0;

TA1CCR1 = 0;

TA1CTL = 0 ;

TA1CCTL1 = 0;

P2OUT &= ~PIN\_OUT;

}

**float** **Average\_filter**(**volatile** uint16\_t \*array)

{

**int** i;

**float** sum;

**for**(i=0;i<SAMPLE\_NUM;i++)

{

sum += array[i];

}

**return** sum/SAMPLE\_NUM;

}

**#pragma** vector=TIMER0\_A1\_VECTOR

**\_\_interrupt** **void** **Timer0\_A1\_ISR**(**void**)

{

**switch**( TA0IV )

{

**case** 2: // CCR1 not used

**break**;

**case** 4: // CCR2 not used

**break**;

**case** 10: // overflow

**#ifdef** TIMER

time++;

**#else**

**if**(counter >9)

counter=0;

//read voltage from P1.7

voltage\_buf[counter] = ADC10\_Read\_Channel(5);

voltage\_pid.Error = ADC\_SET\_POINT - Average\_filter(voltage\_buf);

PID\_process(&voltage\_pid,voltage\_error);

voltage\_out\_ = voltage\_pid.Out;

PWM\_update((uint16\_t)voltage\_out\_);

counter++;

// P1OUT ^= BIT0;

**break**;

**#endif**

}

}

* + - * ***UART.h***

**#ifndef** UART\_H\_

**#define** UART\_H\_

**#include** "Buck\_boost.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* USER DEFINITIONS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**#ifndef** SMCLK\_F

**#define** SMCLK\_F 16000000 // frequency of Sub-System Master Clock in Hz

**#endif**

// This definition supports to UART delay functions. You should change it to the

// right MCLK frequency that you configure through "Config\_Clocks" function.

**#define** BAUDRATE 9600 // may be ... 1200, 2400, 4800, 9600, 19200, ...

// With Launch Pad, the back channel UART to the target MSP is done by

// bit-banging the otherwise unused I/O port lines of the TUSB interface chip

// (by the specific TUSB firmware), and this is limited to 9600

// So you should not try the Baud rate above 9600.

**#define** UART\_RX\_INT\_EN 0

// If you enable the UART receive interrupt (define 1), you don't need to

// wait receive data by function uart\_get & cuart\_gets.

// Instead, you must enable GIE and write your processing code in USCI0RX\_ISR

// (can be found in uart.c)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTIONS 'S PROTOTYPES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// For further description, see UART.c

**void** **uart\_init**();

**void** **uart\_disable**();

**void** **uart\_putc**(**char** c);

**void** **uart\_puts**(**const** **char** \*s);

**void** **uart\_put\_num** (**unsigned** **long** val, **char** dec, **unsigned** **char** neg);

**char** **uart\_data\_ready**();

**#endif** /\* UART\_H\_ \*/

* + - * **UART.c**

**#include** <msp430.h>

**#include** "UART.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTIONS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **uart\_init**(**void**)

{

P1SEL = BIT1 + BIT2 ; // P1.1 = RXD, P1.2=TXD

P1SEL2 = BIT1 + BIT2 ; // P1.1 = RXD, P1.2=TXD

UCA0CTL1 |= UCSSEL\_2; // SMCLK

UCA0CTL1 |= UCSWRST;

UCA0BR0 = 104; // 1MHz 9600

UCA0BR1 = 0; // 1MHz 9600

UCA0MCTL = UCBRS\_0 +UCBRF\_3 + UCOS16; // Modulation UCBRSx = 1

UCA0CTL1 &= ~UCSWRST; // \*\*Initialize USCI state machine\*\*

IE2 |= UCA0RXIE;

}

**void** **uart\_disable**(**void**)

{

IE2 = 0;

}

**void** **uart\_putc**(**char** c)

{

**while**(!(IFG2&UCA0TXIFG)); // Wait until USCI\_A0 TX buffer empty

UCA0TXBUF = c; // assign character to TX buffer

}

**void** **uart\_puts**(**const** **char**\* s)

{

**while**(\*s != '\0'){

uart\_putc(\*s);

s++;

}

}

**char** **uart\_data\_ready**()

{

**if**(UCA0RXIFG) **return** 1;

**else** **return** 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* USCI\_A0 RECEIVE INTERRUPT SERVICE ROUTINE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#pragma** vector = USCIAB0RX\_VECTOR

**\_\_interrupt** **void** **USCI0RX\_ISR**(**void**)

{

uart\_gets(uart\_rev.rev\_buf);

}

* + - * ***ADC.h***

**#ifndef** \_\_ADC\_\_H\_\_

**#define** \_\_ADC\_\_H\_\_

**#include** <stdint.h>

//Chu y thach anh ngoai khong mo phong bang protues duoc

**typedef** **enum** { *ON2\_5V*, //Dien ap tham chieu noi 2.5V

*ON1\_5V*, //Dien ap tham chieu noi 2.5V

*VCC*,//Dien ap nguon,luu y phai loc nhieu tot cho nguon neu dung che do nay

*VeREF*

} Vref;

//Khoi tao ADC

**void** **ADC10\_Init**(Vref V\_tham\_chieu);

//Doc 1 kenh ADC

**unsigned** **int** **ADC10\_Read\_Channel**(**unsigned** **char** channel); //1->16

**#endif**

* + - * ***ADC.c***

**#include** <msp430g2253.h>

**#include** "ADC.h"

**volatile** **unsigned** **char** Reading\_channel=0;

**volatile** **unsigned** **char** PORT\_ADC=BIT4;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\

\* Function (noi dung ham)

\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **ADC10\_Init**(Vref V\_tham\_chieu)

{

ADC10CTL0 &= ~ENC;

**while**(ADC10CTL1 & ADC10BUSY);

ADC10AE0=PORT\_ADC;

ADC10CTL0 = ADC10SHT\_3 + ADC10ON;

**switch**(V\_tham\_chieu)

{

**case**(*ON2\_5V*):

{

ADC10CTL0|= SREF\_1 + REFON + REF2\_5V ;

**break**;

}

**case**(*ON1\_5V*):

{

ADC10CTL0|= SREF\_1 + REFON ;

**break**;

}

**case**(*VCC*):

{

ADC10CTL0|= SREF\_0;

**break**;

}

**case**(*VeREF*):

{

ADC10CTL0|= SREF\_2 + REFOUT ;

**break**;

}

}

//Bien doi tung kenh

ADC10CTL1 = CONSEQ\_0;

ADC10DTC0=0; //Tat DTC

ADC10DTC1=0;

// Cho phep chuyen doi

ADC10CTL0 |= ENC;

// Bat dau chuyen doi

ADC10CTL0 |= ADC10SC;

}

**unsigned** **int** **ADC10\_Read\_Channel**(**unsigned** **char** channel)

{

//Neu kenh doc khac lan cuoi cung doc

**if**(channel != Reading\_channel)

{

//Disable chuyen doi

ADC10CTL0&=~( ADC10SC + ENC );

//Thay doi kenh can chuyen doi-chi dung trong che do don kenh

ADC10CTL1 &= 0x0FFF;

ADC10CTL1|=(channel<<12);

Reading\_channel = channel;

ADC10CTL0 &= ~ADC10IFG; //Tat co ngat

ADC10CTL0|= (ADC10SC + ENC);

//\_\_bis\_SR\_register(CPUOFF + GIE); // Ngat CPU cho den khi chuyen doi xong

**while**(!(ADC10CTL0 & ADC10IFG)); //Cho den khi chuyen doi xong

**return** ADC10MEM; //Lay gia tri ADC

}

**else** //Neu channel giong channel da doc lan truoc

{

ADC10CTL0 &= ~ADC10IFG; //Tat co ngat

ADC10CTL0|= (ADC10SC + ENC); //Cho phep chuyen doi

//\_\_bis\_SR\_register(CPUOFF + GIE); // Ngat CPU cho den khi chuyen doi xong

**while**(!(ADC10CTL0 & ADC10IFG)); //Cho den khi chuyen doi xong

**return** ADC10MEM; //Lay gia tri ADC

}

}

**#pragma** vector=ADC10\_VECTOR

**\_\_interrupt** **void** **ADC10\_ISR**(**void**)

{

**\_\_bic\_SR\_register\_on\_exit**(CPUOFF); // Clear CPUOFF bit from 0(SR)

}

* + - * ***Basic\_config.h***

**#ifndef** BASIC\_CONFIG\_H\_

**#define** BASIC\_CONFIG\_H\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* DEFINITIONS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These definitions help you to notice the system clocks

**#define** MCLK\_F 16 // frequency of Master Clock in MHz

**#define** SMCLK\_F 16000000 // frequency of Sub-System Master Clock in Hz

**#define** ACLK\_F 12000 // frequency of Auxiliary Clock in Hz

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTIONS 'S PROTOTYPES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **Config\_stop\_WDT** (**void**);

**void** **Config\_Clocks** (**void**);

**void** **delay\_us** (**int** t);

**void** **delay\_ms** (**int** t);

**#endif** /\* BASIC\_CONFIG\_H\_ \*/

* + - * ***Basic\_config.c***

**#include** <msp430.h>

**#include** "Basic\_config.h"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Stop Watch-dog Timer

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **Config\_stop\_WDT**(**void**)

{

WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Clocks Configurations

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **Config\_Clocks**(**void**)

{

**if** (CALBC1\_16MHZ ==0xFF || CALDCO\_16MHZ == 0xFF) // Check if constants cleared

{

**while**(1); // If cal constants erased, trap CPU!!

}

BCSCTL1 = CALBC1\_16MHZ; // Set DCO range & ACLK prescaler

DCOCTL = CALDCO\_16MHZ; // Set DCO step + modulation

BCSCTL3 |= LFXT1S\_2 + XCAP\_3; // configure ACLK Source

**while**(IFG1 & OFIFG) // wait for OSCFault to clear

{

IFG1 &= ~OFIFG;

**\_\_delay\_cycles**(100000);

}

BCSCTL2 |= SELM\_0 + DIVM\_0; // select MCLK, SMCLK clock and prescaler

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Delay for t microseconds

// The delay is not exact when t is too small

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **delay\_us** (**int** t)

{

**int** i;

**for** (i = 0; i<t; i++ )

\_delay\_cycles(MCLK\_F);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Delay for t milliseconds

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **delay\_ms** (**int** t)

{

**int** i;

**for** (i = 0; i<t; i++ )

\_delay\_cycles(MCLK\_F\*1000);

}

### Thiết kế bộ điều khiển PID

* + - * ***PID.h***

**#ifndef** PID\_H\_

**#define** PID\_H\_

**typedef**

**struct** {

**float** Ki;

**float** Kp;

**float** Kd;

**float** D\_part;

**float** I\_part;

**float** P\_part;

**float** T;

**float** pre\_Out;

**float** Out;

**loat** pre\_pre\_Error;

**float** pre\_Error;

**float** saturate\_h;

**float** saturate\_l;

**float** I\_part\_sat;

**float** Error;

}

pidparams;

**void** **PID\_process**(**volatile** pidparams \*pid, **float** error);

**void** **PID\_init**(**volatile** pidparams \*pid, **float** Kp,**float** Ki,**float** Kd,

**float** saturate\_l, **float** saturate\_h,**float** t,**float** I\_part\_sat);

**#endif** /\* LIB\_PID\_H\_ \*/

* + - * ***PID.c***

**#include** "pid.h"

**void** **PID\_process**(**volatile** pidparams \*pid, **float** error)

{

pid->P\_part = pid->Kp \* pid->Error;

pid->I\_part += pid->Ki \* pid->Error / pid->T;

pid->I\_part = pid->I\_part < pid->I\_part\_sat ? pid->I\_part : pid->I\_part\_sat;

pid->I\_part = pid->I\_part > -pid->I\_part\_sat ? pid->I\_part : -pid->I\_part\_sat;

pid->D\_part = pid->Kd \* (pid->Error - pid->pre\_Error) \* pid->T;

pid->Out += pid->P\_part + pid->I\_part + pid->D\_part;

pid->pre\_Error = pid->Error;

**if** (pid->Out < pid->saturate\_l)

pid->Out = pid->saturate\_l;

**else** **if**(pid->Out > pid->saturate\_h)

pid->Out = pid->saturate\_h;

}

**void** **PID\_init**(**volatile** pidparams \*pid, **float** \_Kp,**float** \_Ki,**float** \_Kd,

**float** \_saturate\_l, **float** \_saturate\_h,**float** \_t,**float** \_I\_part\_sat)

{

pid->Kp = \_Kp;

pid->Ki = \_Ki;

pid->Kd = \_Kd;

pid->saturate\_l = \_saturate\_l;

pid->saturate\_h = \_saturate\_h;

pid->T = \_t;

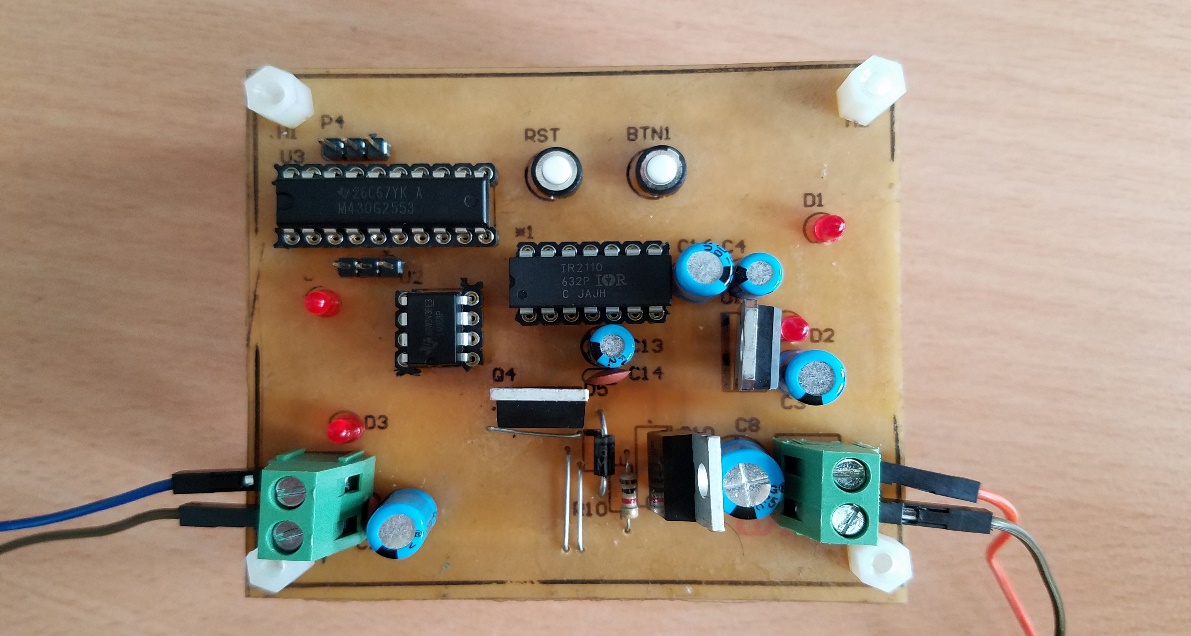
pid->I\_part\_sat = \_I\_part\_sat;

}

# CHƯƠNG 3: THÍ NGHIỆM VÀ ĐÁNH GIÁ

## 3.1. Thí nghiệm đáp ứng ngỏ ra

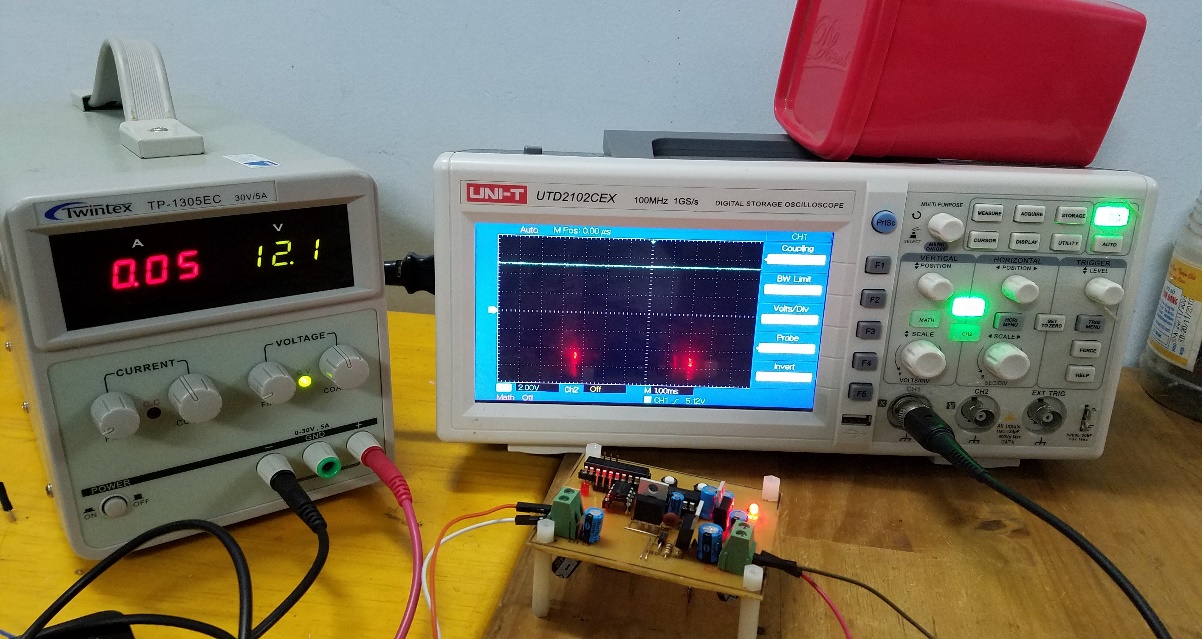
### 3.1.1. Mô hình mạch BUCK hoàn chỉnh



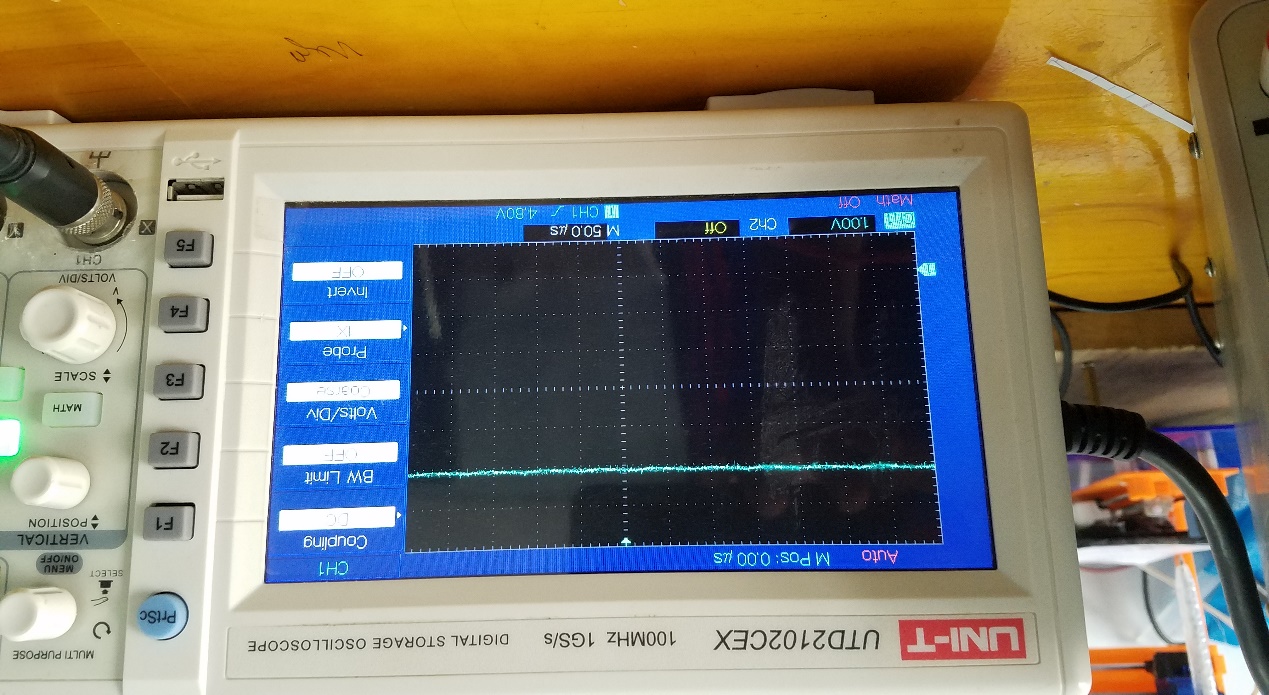
***Hình 3.1****: Mạch hoàn chỉnh*

### 3.1.2. Kết quả thí nghiệm

* + - * ***Thí nghiệm không tải, Vin=12V***



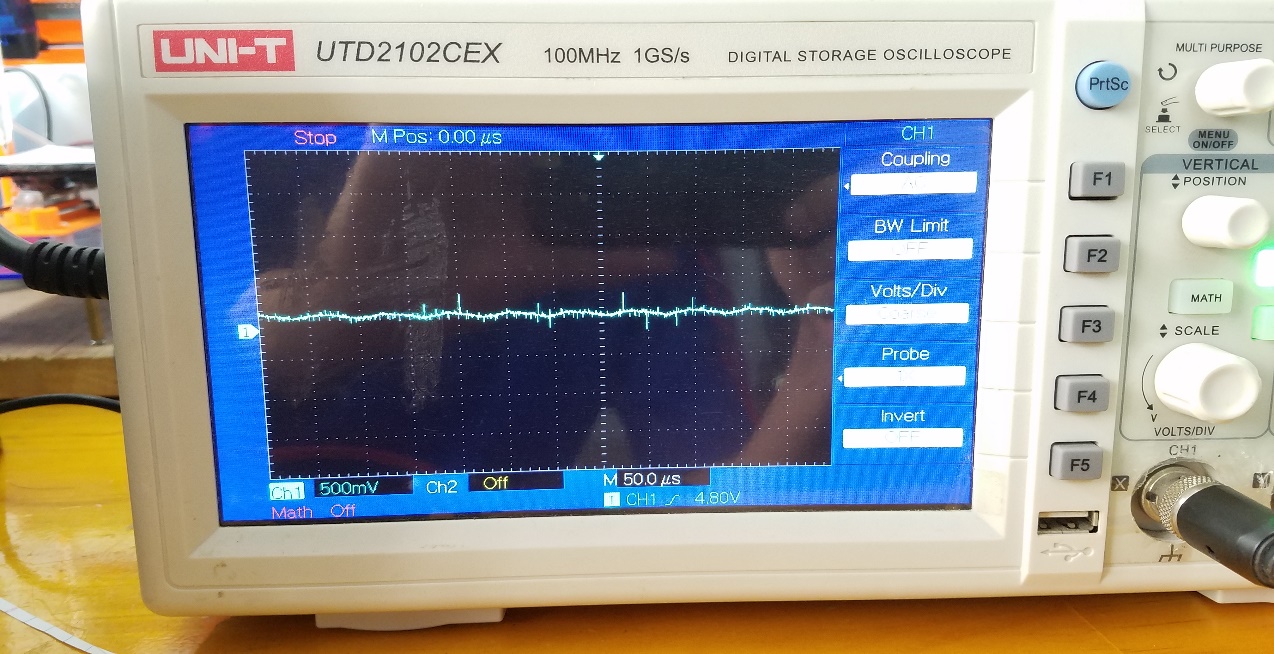
***Hình 3.2****: Thí nghiệm không tải, đo áp ngỏ ra*



***Hình 3.3:*** *Thí nghiệm không ải, đo áp ngỏ ra Vout*



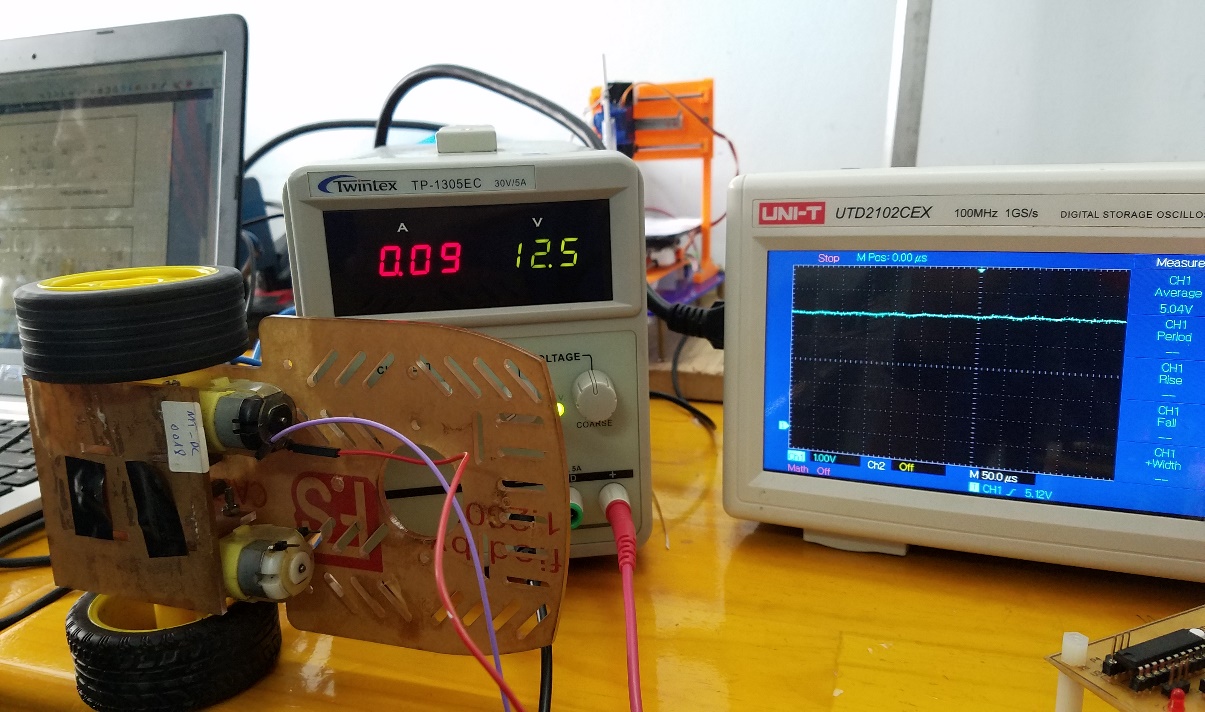
***Hình 3.4:*** *Thí nghiệm không tải, đo các thông số của Vout*



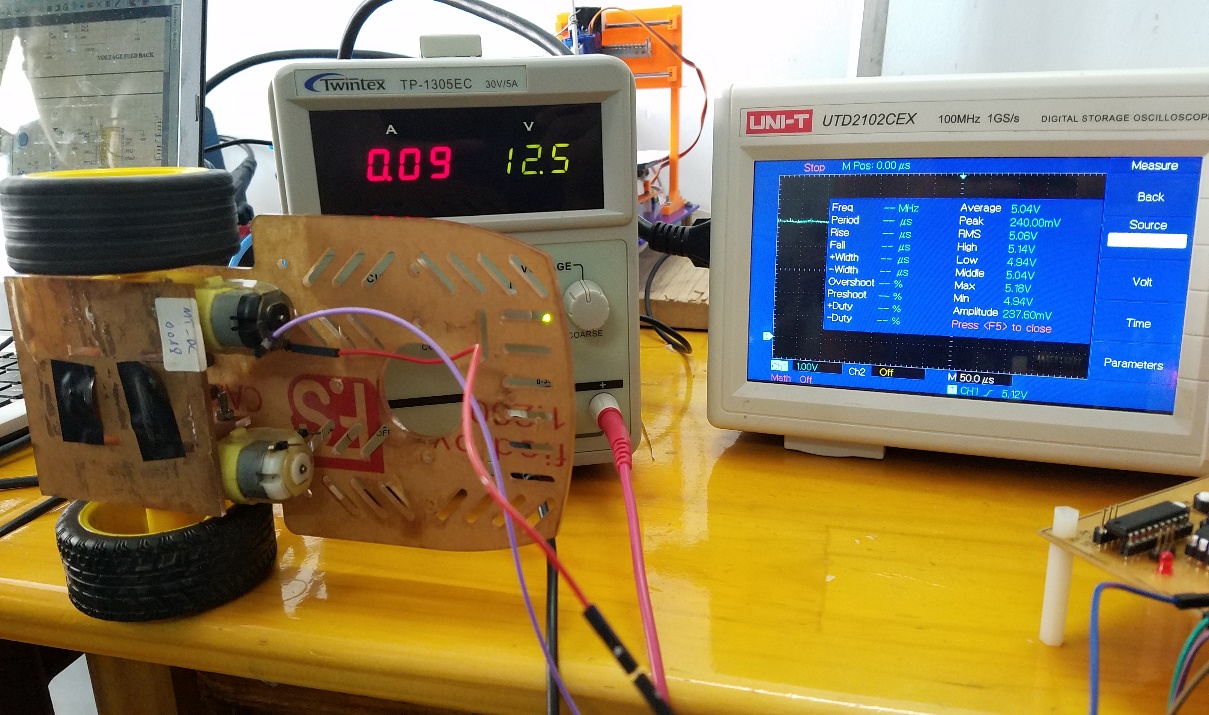
***Hình 3.5:*** *Thí nghiệm không tải, đo thành phần AC của áp ra*

* + - * ***Thí nghiệm với tải là động cơ DC***

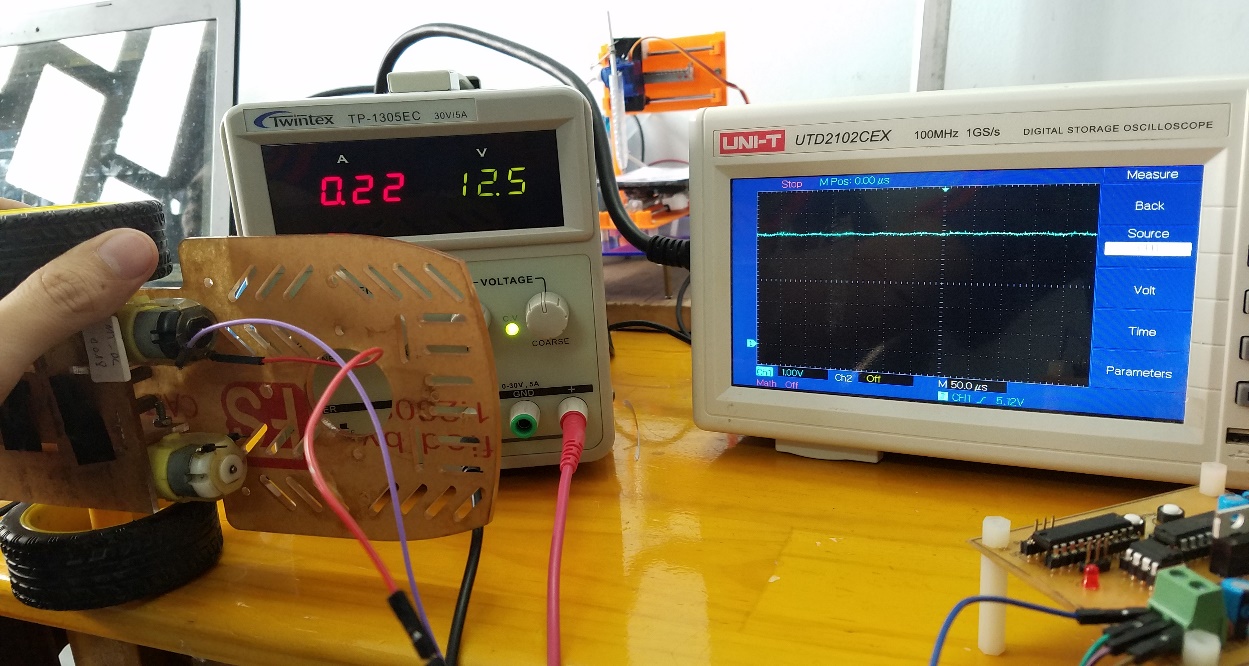
*Video kết quả thí nghiệm:* [*https://youtu.be/-o3ZBtGTLWE*](https://youtu.be/-o3ZBtGTLWE)



*Hình 3.6: Thí nghiệm động cơ không tải, đo áp ngỏ ra*



*Hình 3.7: Thí nghiệm động cơ DC không tải, đo thông số áp ra*



***Hình 3.8:*** *Thí nghiệm động cơ DC full tải, đo áp ngỏ ra*



***Hình 3.9:*** *Thí nghiệm động cơ DC full tải, đo các thông số ngỏ ra*

* + - * + ***Thí nghiệm tải trở R:***

*Video kết quả thí nghiệm:* [*https://youtu.be/bHc4NsDcsmk*](https://youtu.be/bHc4NsDcsmk)

## 3.2. Đánh giá và nhận xét

### 3.2.1. Tính toán độ nhấp nhô áp ra

* + - * Trường hợp không tải: 4.37%
      * Trường hợp động cơ DC không tải: 4.76%
      * Trường hợp động cơ DC full tải: 5.53%

### 3.2.2. Nhận xét ưu, nhược điểm và biện pháp khắc phục

* + - * Ưu điểm:
* Mạch có thể hạ áp từ điện áp bất kỳ từ 20V đến 12V xuống 10V đến 3V, thích hợp với nhiều loại nguồn vào và có thể cung cấp điện cho nhiều loại linh kiện có các mức điện áp khác nhau.
* Điện áp ra tương đối ổn định với độ nhấp nhô 5% và có thể cung cấp điện cho tải trở và tải động cơ hoạt động bình thường.
* Mạch có công suất lớn có thể chạy với dòng tải lên đến 3A.
  + - * Khuyết điểm:
* Áp ngỏ ra chưa thật sự phẳng do tần số đóng cắt mạch bị hạn chế bởi vi điều khiển.
* Các thành phần hài bậc cao vẫn chưa được lọc hoàn toàn.
* Nhiễu tập trung nhiều tại các thời điểm đóng cắt mạch.
  + - * Biện pháp khắc phục:
* Thay đổi vi điều khiển có tần số hoạt động cao hơn.
* Tăng giá trị tụ lọc tại áp ra nhưng quá lớn sẽ gây ra trễ nếu muốn đáp ứng điện áp tức thời.
* Thiết kế mạch tuân thủ các quy định tốt hơn.

# KẾT LUẬN

Mạch làm ra tương đối nhỏ gọn, hoạt động tương đối ổn định với điện áp bé hơn 2A và có thể cung cấp cho nhiều loại linh kiện có các mức điện áp nhác nhau. Tuy nhiên điện áp ra chưa tuyệt đối phẳng do còn giới hạn về phần cứng cũng như vẽ mạch PCB chưa được tốt nhất. Qua dự án em đã thực sự hiểu rõ hơn về mạch Buck cũng như cách mạch có thể hạ từ một điện áp cao xuống một điện áp thấp hơn. Do còn hạn chế về mặt kiến thức nên em rất mong những ý kiến đóng góp từ các thầy cô để mạch của em thêm hoàn thiện. Em chân thành cảm ơn!

# TÀI LIỆU THAM KHẢO

[1] Nguyễn Văn Nhờ, Điện tử công suất, Nhà xuất bản đại học quốc gia TPHCM

[2] PGS.TS Lê Minh Phương, Slide bài giảng Điện tử công suất

[3] Muhammad Umar Abbasi, Digital Control Of A Buck Converter Using An 8 Bit STM Microcontroller, INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 6, ISSUE 04, APRIL 2017

[4] International Rectifier, Application Note AN-978

[5] ROHM semiconductor, Switching Regulator Series - PCB Layout Techniques of Buck Converter

[6] Texas Instruments Incorporated, MSP430x2xx Family User's Guide