TRƯỜNG ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC BÁCH KHOA

BỘ MÔN ĐIỆN – ĐIỆN TỬ

ꕥ

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

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.

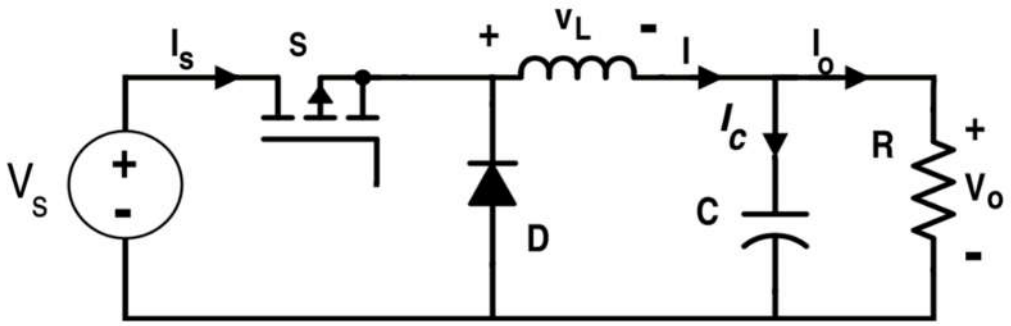
CHƯƠNG 1: TỔNG QUAN VỀ BỘ BIẾN ĐỔI BUCK

1.1. 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.

1.1.1. Nguyên lý hoạt động

a. Sơ đồ bộ biến đổi

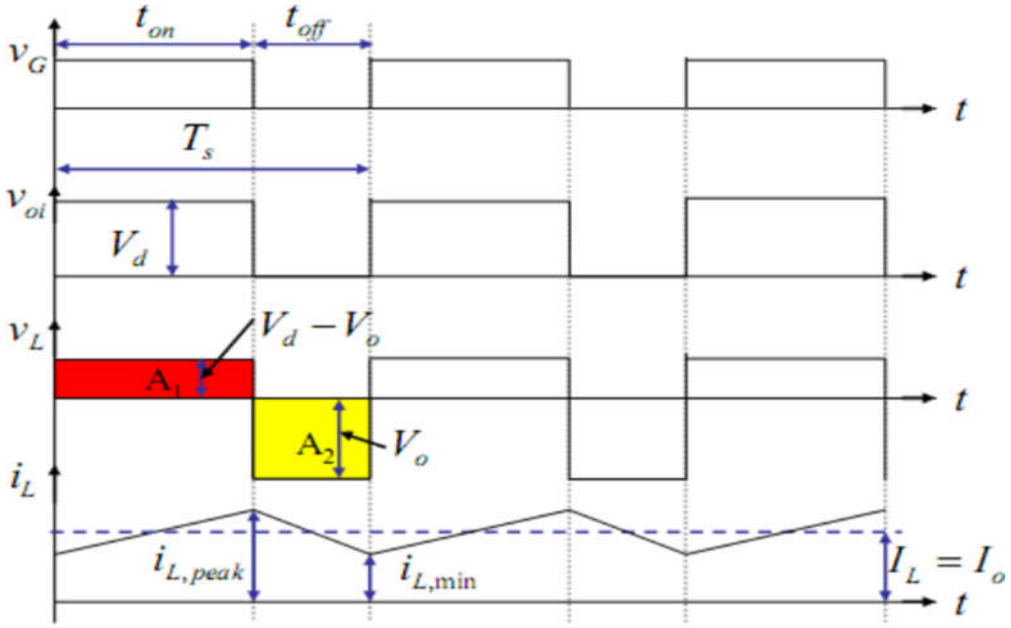


***Hình 1.1**: Sơ đồ nguyên lý bộ Buck*

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.

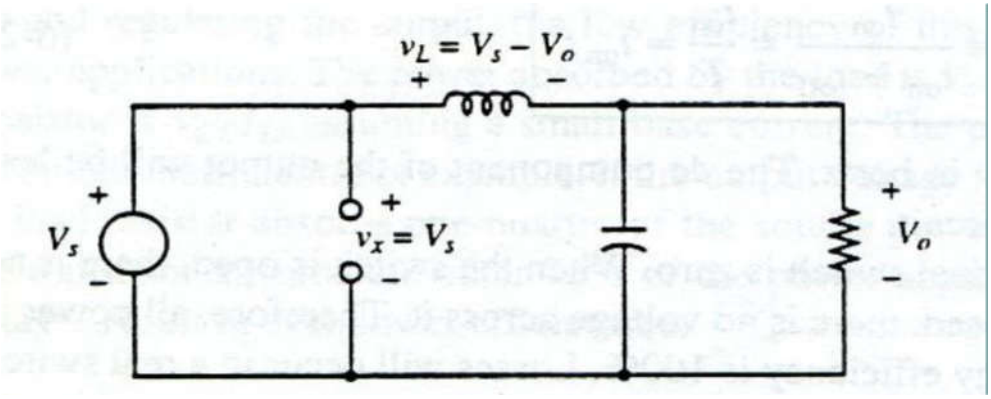
b. Nguyên lý làm việc ở chế độ liên tục

Một bộ chuyển đổi buck hoạt động trong chế độ liên tục có dòng điện chạy qua cuộn dây I_L 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 = t_{ON}/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 V_O = Vd nên áp cuộn dây tăng và năng lượng được tích trữ trong cuộng cảm L. Áp của cuộn cảm V_L=V_S – V_O vì điện dung C xem như vô cùng lớn và áp ra là hằng số.

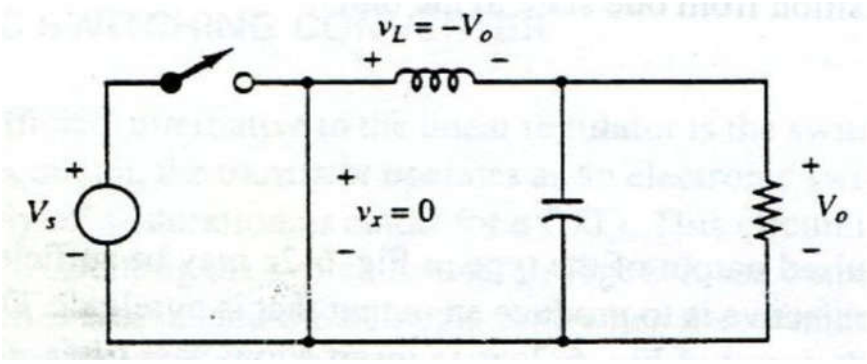
Ta có các công thức sau:

$$v_L = L\frac{di_L}{dt} = V_S + V_O = L\frac{I_2 - I_1}{t1} = L\frac{\Delta I_L}{t1}$$

$$t_1 = \frac{\Delta I_L}{V_S - V_O}$$

$$\Delta I_L = \frac{(V_S - V_O)}{L} DT$$

Xét khoảng S ngắt:



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

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

Ta có các công thức sau:

$$V_L = L\frac{di_L}{dt} = -V_O = L\frac{I_1 - I_2}{t2} = -L\frac{\Delta I_L}{t_2}$$

$$t_2 = \frac{\Delta I_L L}{V_o}$$

$$\Delta I_L = \frac{-V_o}{L}(1 - D)T$$

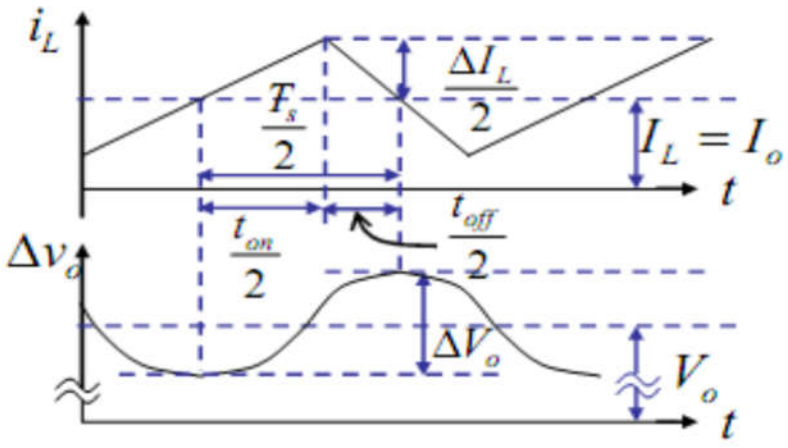
Hệ quả: $\Delta I = \frac{(V_S - V_O)}{L} DT = \frac{V_O}{L}(1 - D)T$

Trị trung bình áp tải: $V_O = DV_S$

Trị trung bình dòng điện tải: $I_O = I_L \;;\; I_S = DI_O$

Độ nhấp nhô dòng điện tải: $\Delta I = \frac{(V_S - V_O)\, DT_S}{L}$ (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:

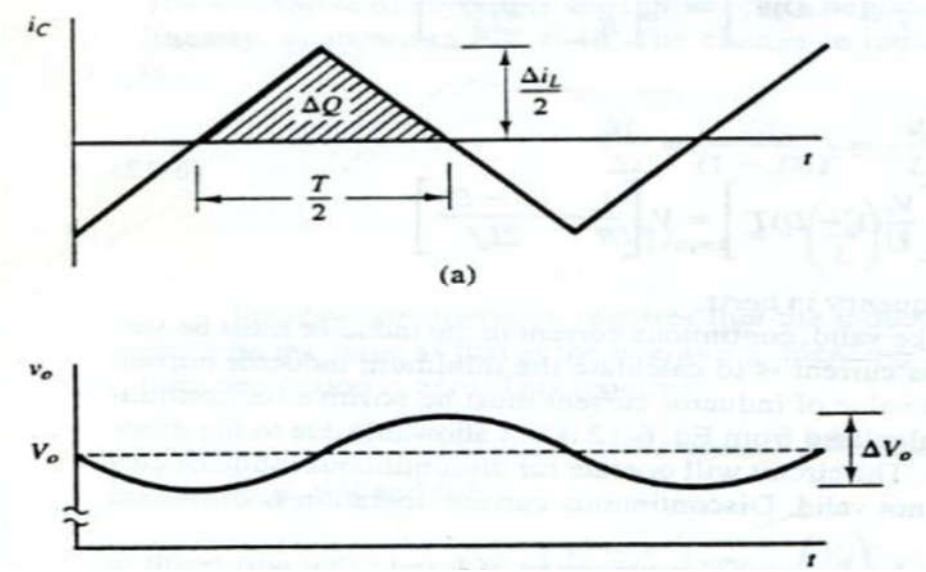
$$I_{max} = I_L + \frac{\Delta i_L}{2} = \frac{V_o}{R} + \frac{1}{2}\left[\frac{V_o}{L}(1-D)T\right] = V_o[\frac{1}{R} + \frac{(1-D)}{2Lf}]$$

$$I_{min} = I_L - \frac{\Delta i_L}{2} = \frac{V_o}{R} - \frac{1}{2}\left[\frac{V_o}{L}(1-D)T\right] = V_o[\frac{1}{R} - \frac{(1-D)}{2Lf}]$$

Giá trị L_{min} để dòng I_L liên tục:

$$L_{min} = \frac{(1-D)R}{2f} \quad (2)$$

Độ nhấp nhô điện áp tải:



Hình 1.6: *Đồ thị nhấp nhô áp tải*

Ta có:

$$i_C = i_L - i_R$$

$$Q = CV_0$$

$$\Delta Q = C\Delta V_0$$

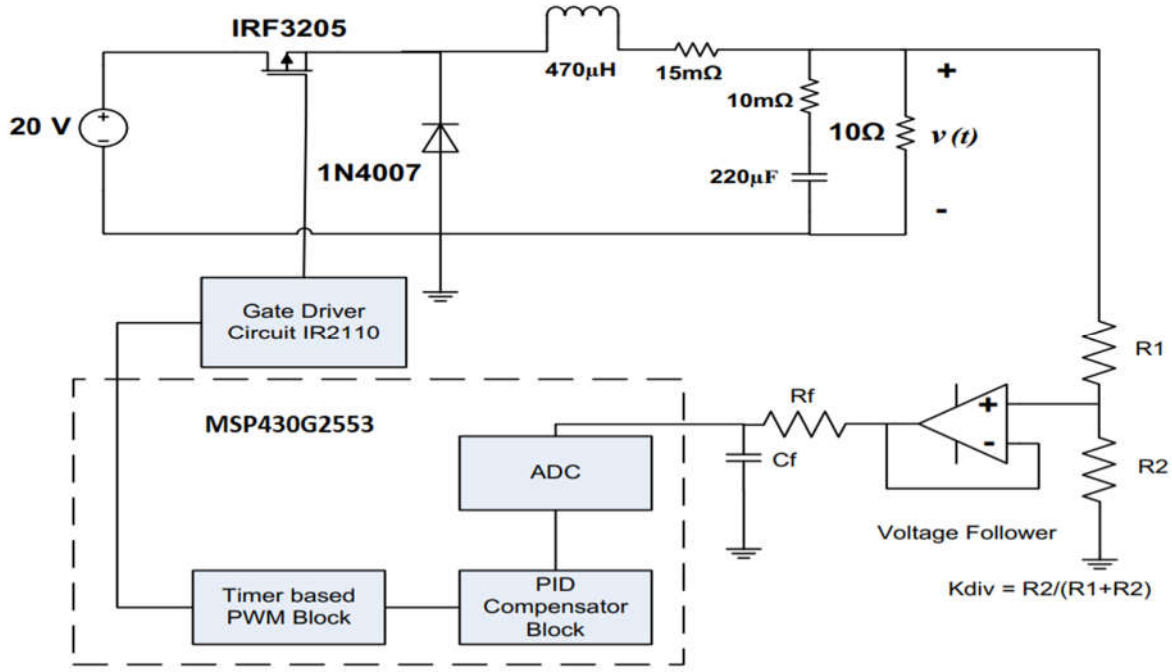
$$\Delta V_0 = \frac{\Delta Q}{C}$$

Suy ra:

$$\Delta Q = \frac{1}{2}\left(\frac{T}{2}\right)\left(\frac{\Delta i_L}{2}\right) = \frac{T\Delta i_L}{8}$$

$$\frac{\Delta V_0}{V_0} = \frac{1-D}{8LCf^2} \quad (3)$$

1.1.2. 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 D_{min} = 0.1, R=10(ohm), f = 16KHz. Theo công thức (2) ta được L_{min} = 281uH. Chọn tụ có điện cảm H = 330uH.

❖ *Tụ điện C_{out}:*

Để độ nhấp nhô áp dưới 5%, D_{min}=0.1. Theo công thức (3) ta được C_{out} = 31.26uF. Chọn tụ C_{out} = 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*

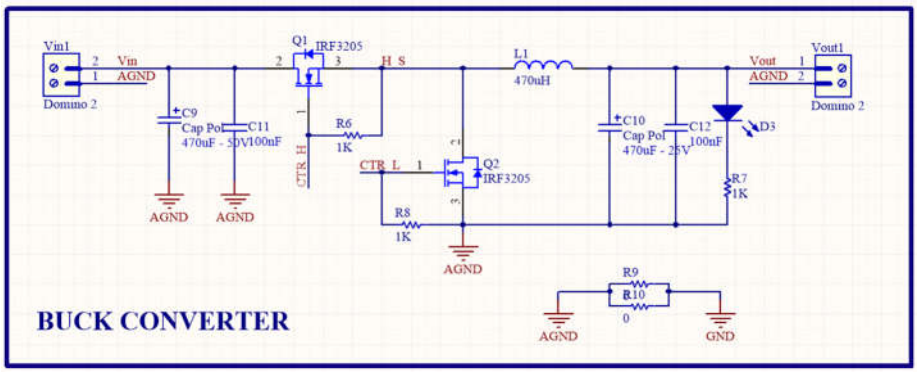
❖ *IC nguồn: LM7805, LM1117*

❖ *Vi điều khiển MSP430G2553*

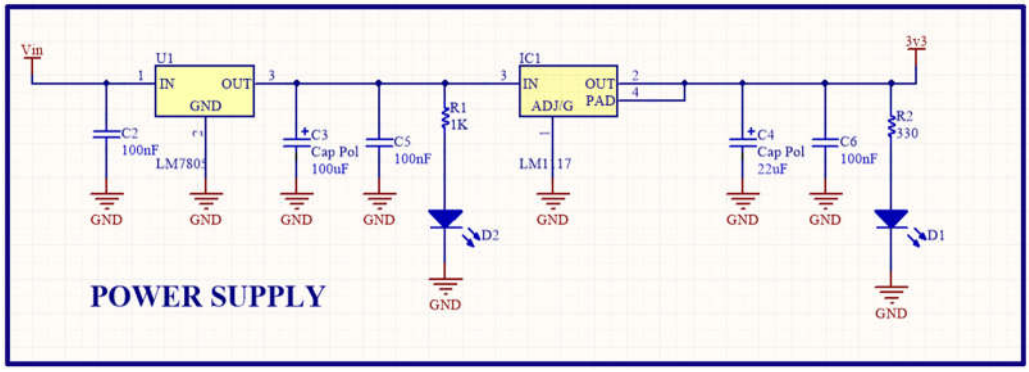
❖ *Tụ và trở như trong Schematic*

2.1.2. Sơ đồ Schematic

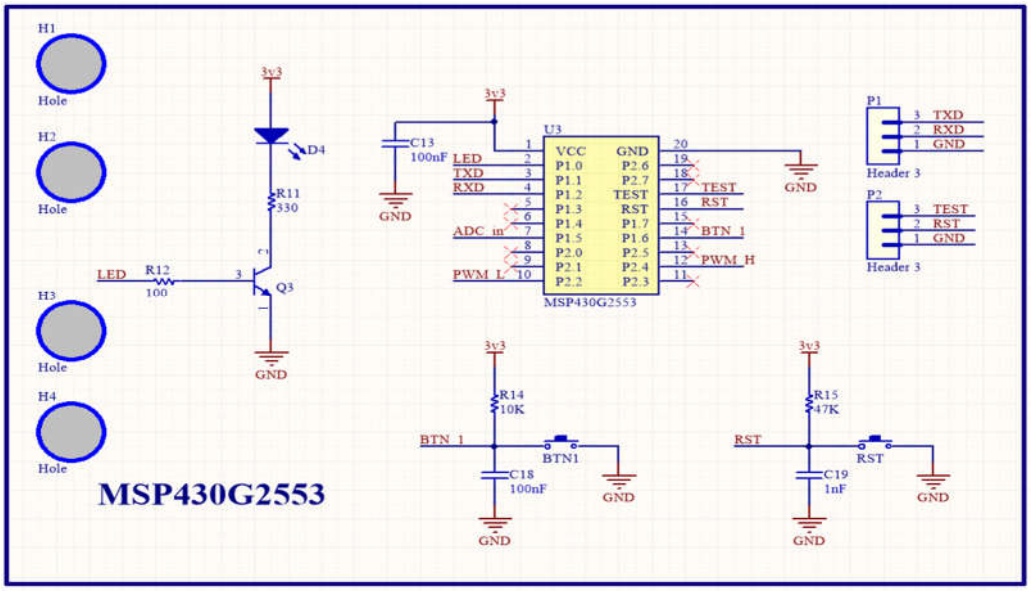
Gồm 5 khối: khối BUCK, khối nguồn tuyến tính, khối vi điều khiển, khối hồi tiếp, khối booststrap



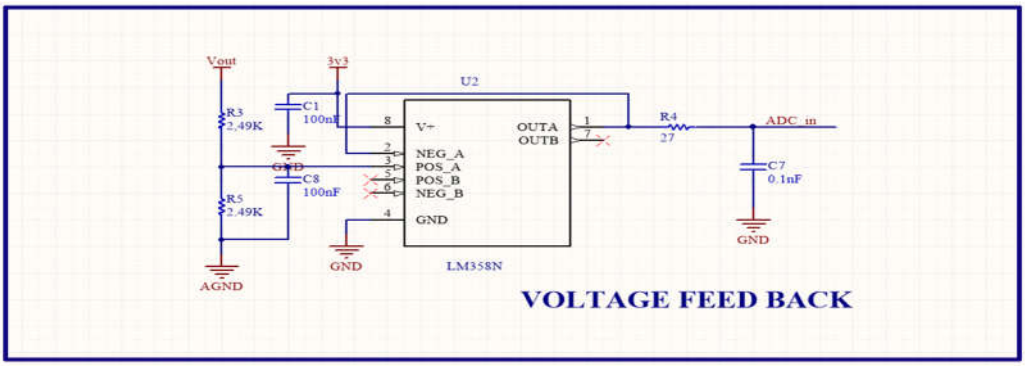
***Hình 2.1**: Sơ đồ khối BUCK*



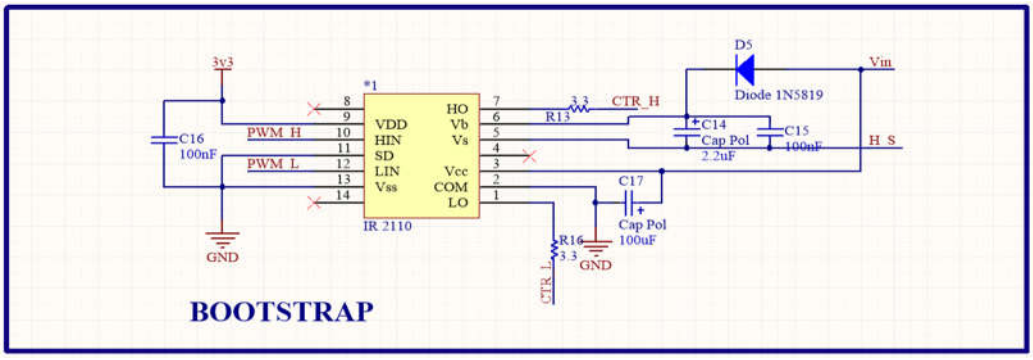
***Hình 2.2**: Sơ đồ khối mạch nguồn tuyến tính*



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*

2.2. Thiết kế PCB

2.2.1. 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

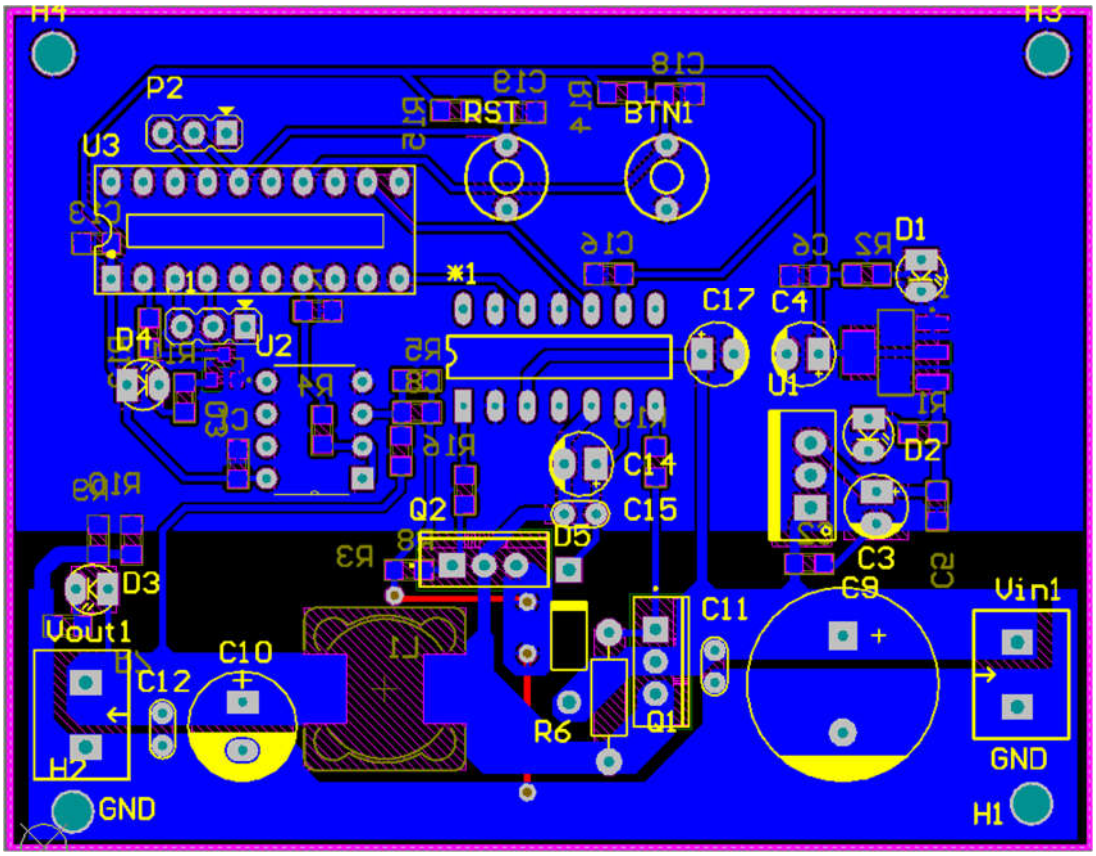
- Đặt cuộn cảm gần IC, nhưng không cần phải gần như tụ. Điều này giúp làm giảm bức xạ và mạch sẽ nhỏ hơn

- 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

2.2.2. Sơ đồ PCB



Hình 2.6: *Sơ đồ mạch PCB*

2.3. Firmware và thiết kế bộ điều khiển PID

2.3.1. 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.

2.3.2. 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();
```

```
20.
21.     char str[100];
22.
23.     //void PID_init(&voltage_pid, 1, 1,1,1, 1,1);
24.     voltage_pid.Kp = 0.01;
25.     voltage_pid.Ki = 0;
26.     voltage_pid.Kd = 0.001;
27.     voltage_pid.saturate_l = 50;
28.     voltage_pid.saturate_h = 450;
29.     voltage_pid.T = 1;
30.     voltage_pid.I_part_sat = 50;
31.
32.     while(1)
33.     {
34.     voltage_error = voltage_pid.Error;
35.     voltage_out = voltage_pid.Out;
36.     sprintf(str, "0 %d %d 500", (uint16_t)voltage_error,(uint16_t)voltage_out);
37.     uart_puts(str);
38.     uart_puts("\n\r");
39.     delay_ms(1);
40.     }
41.     }
```

❖ **Buck_boost.h**

```
42.     #ifndef BUCK_BOOST_H_
43.     #define BUCK_BOOST_H_
44.
45.     #include "UART.h"
```

```
46.  #include "Basic_config.h"
47.  #include "ADC.h"
48.  #include "pid.h"
49.  #include <stdint.h>
50.
51.  //#define TIMER
52.
53.  #define TIME_PERIOD    16                  //calculate 1us
54.
55.  #define TIME_INTERVAL      50              // in microsecond
56.  #define DUTY_CIRCLE        0.5             //        0        <
DUTY_CIRCLE < 1
57.  #define PWM_CIRCLE
      (uint16_t)(TIME_INTERVAL*SMCLK_)
58.
59.  #define SMCLK_             16
60.  #define SAMPLING_TIME      (SMCLK_*1000)   // sampling at 100us
with clock at 16MHz
61.
62.  #define PIN_IN             BIT5            // read form P1.5
63.  #define PIN_OUT            (BIT2 + BIT4)           //   output   at   P2.2,
P2.4
64.
65.  #define LED_RED            BIT0
66.  #define LED_GREEN              BIT6
67.  #define BTN1                   BIT3
68.
```

```
69.  #define SAMPLE_NUM          10
70.  #define VOLTAGE_K           6.0
71.  #define V_REF               2.5
72.
73.  #define DESIRE_VOLTAGE      5.0          //Set out put voltage
74.  #define ADC_RESOLUTION      1024.0
75.  #define                                  ADC_SET_POINT
  (DESIRE_VOLTAGE/VOLTAGE_K/V_REF*ADC_RESOLUTION)
76.
77.  volatile uint16_t voltage_buf[SAMPLE_NUM];
78.  volatile pidparams voltage_pid;
79.
80.  void Buck_boost_init(void);
81.  void GPIO_init(void);
82.  void Timer_start(void);
83.  void Timer_stop(void);
84.
85.  void PWM_start(void);
86.  void PWM_update(uint16_t duty);
87.  void PWM_stop(void);
88.
89.  void PID_update(pidparams *pid);
90.
91.  float Average_filter(volatile uint16_t *array);
92.
93.  #endif /* LIB_BUCK_BOOST_H_ */
```

❖ ***Buck_boost.c***

```
94.   #include <msp430.h>
95.   #include "Buck_boost.h"
96.
97.   int i=0;
98.   volatile uint8_t counter=0;
99.   volatile float voltage_error;
100.  volatile uint16_t voltage_out_;
101.  unsigned int vout;
102.  /*************************************************************
      ***********
103.  * FUNCTIONS
*************************************************************************
****

104.  void GPIO_init(void)
105.  {
106.    P1SEL  &= ~(BTN1 + LED_GREEN + LED_RED);
107.    P1SEL2 &= ~(BTN1 + LED_GREEN + LED_RED);
108.    //Init button
109.    P1DIR  &= ~BTN1;
110.    P1REN  |= BTN1;
111.    P1OUT  |= BTN1;
112.    P1IE   |= BTN1;
113.    P1IES  |= BTN1;
114.    //Init LED GREEN
```

```
115.        P1DIR |= LED_GREEN;
116.        P1OUT &= ~LED_GREEN;
117.        //Init LED RED
118.        P1DIR |= LED_RED;
119.        P1OUT &= ~LED_RED;
120.        //Init PWM output
121.        P2DIR |= PIN_OUT;
122.        P2OUT &= ~PIN_OUT;
123.        P2SEL|= PIN_OUT;
124.        //Global interupt
125.        _BIS_SR(GIE);
126.
127.        }
128.
129.        void Timer_start(void)
130.        {
131.        TA0CCR0 = SAMPLING_TIME;
132.        TA0CTL = TASSEL_2 + MC_1 + TAIE;// SMCLK, Up mode, Overflow
interrupt
133.        TA0CCTL0 &= ~CCIFG;
134.        }
135.
136.       void Timer_stop(void)
137.       {
138.     TA0CTL = 0 ;
139.     TA0R = 0;
140.     P1OUT  &= ~LED_GREEN;
```

```
141.    }
142.
143.
144.    void PWM_start(void)
145.    {
146.    TA1CCR0 = 512;
147.    TA1CCR2 = 150;
148.    TA1CCR1 = TA1CCR2 + 20;
149.    TA1CTL = TASSEL_2 + MC_3 ; // SMCLK, Up mode
150.    TA1CCTL1 = OUTMOD_6;        // falling edge & raising edge, capture mode, capture/compare
151.    TA1CCTL2 = OUTMOD_2;
152.    }
153.
154.    void PWM_update(uint16_t duty)
155.    {
156.    TA1CCR1 = duty + 20;
157.    TA1CCR2 = duty;
158.    }
159.
160.    void PWM_stop(void)
161.    {
162.    TA1CCR0 = 0;
163.    TA1CCR1 = 0;
164.    TA1CTL = 0 ;
165.    TA1CCTL1 = 0;
166.    P2OUT &= ~PIN_OUT;
```

```
167.    }
168.
169.    float Average_filter(volatile uint16_t *array)
170.    {
171.    int i;
172.    float sum;
173.    for(i=0;i<SAMPLE_NUM;i++)
174.    {
175.    sum += array[i];
176.    }
177.    return sum/SAMPLE_NUM;
178.    }
179.
180.    #pragma vector=TIMER0_A1_VECTOR
181.    __interrupt void Timer0_A1_ISR(void)
182.    {
183.      switch( TA0IV )
184.      {
185.      case  2:                    // CCR1 not used
186.      break;
187.      case  4:                    // CCR2 not used
188.       break;
189.      case 10:                     // overflow
190.    #ifdef TIMER
191.      time++;
192.    #else
193.      if(counter >9)
```

```
194.        counter=0;
195.        //read voltage from P1.7
196.        voltage_buf[counter] = ADC10_Read_Channel(5);
197.        voltage_pid.Error = ADC_SET_POINT - Average_filter(voltage_buf);
198.
199.        PID_process(&voltage_pid,voltage_error);
200.        voltage_out_ = voltage_pid.Out;
201.        PWM_update((uint16_t)voltage_out_);
202.
203.        counter++;
204.        //    P1OUT ^= BIT0;
205.        break;
206.    #endif
207.        }
208.    }
```

❖ ***UART.h***

```
1.    #ifndef UART_H_
2.    #define UART_H_
3.
4.    #include "Buck_boost.h"
5.
/*********************************************************************
6.     * USER DEFINITIONS
7.    ********************************************************************
**********
8.
```

```
9.  #ifndef SMCLK_F
10. #define SMCLK_F 16000000 // frequency of Sub-System Master Clock in Hz
11. #endif
12. // This definition supports to UART delay functions. You should change it to the
13. // right MCLK frequency that you configure through "Config_Clocks" function.
14.
15. #define BAUDRATE 9600 // may be ... 1200, 2400, 4800, 9600, 19200, ...
16. // With Launch Pad, the back channel UART to the target MSP is done by
17. // bit-banging the otherwise unused I/O port lines of the TUSB interface chip
18. // (by the specific TUSB firmware), and this is limited to 9600
19. // So you should not try the Baud rate above 9600.
20.
21. #define UART_RX_INT_EN 0
22. // If you enable the UART receive interrupt (define 1), you don't need to
23. // wait receive data by function uart_get & cuart_gets.
24. // Instead, you must enable GIE and write your processing code in USCI0RX_ISR
25. // (can be found in uart.c)
26.
27. /*************************************************************
**********
28. * FUNCTIONS 'S PROTOTYPES
```

```
29.  ************************************************************
**********/
30.  // For further description, see UART.c
31.  void uart_init();
32.  void uart_disable();
33.  void uart_putc(char c);
34.  void uart_puts(const char *s);
35.  void uart_put_num (unsigned long val, char dec, unsigned char neg);
36.  char uart_data_ready();
37.  #endif /* UART_H_ */
```

❖ **UART.c**

```
1.  #include <msp430.h>
2.  #include "UART.h"
3.
4.  /************************************************************
**********
5.  * FUNCTIONS
6.  ************************************************************
**********
7.  void uart_init(void)
8.  {
9.  P1SEL = BIT1 + BIT2 ;                  // P1.1 = RXD, P1.2=TXD
10.  P1SEL2 = BIT1 + BIT2 ;                 // P1.1 = RXD, P1.2=TXD
11.  UCA0CTL1 |= UCSSEL_2;                    // SMCLK
12.  UCA0CTL1 |= UCSWRST;
13.  UCA0BR0 = 104;                        // 1MHz 9600
```

```
14.     UCA0BR1 = 0;                        // 1MHz 9600
15.     UCA0MCTL = UCBRS_0 +UCBRF_3 + UCOS16;                   // Modulation UCBRSx = 1
16.     UCA0CTL1 &= ~UCSWRST;                   // **Initialize USCI state machine**
17.     IE2 |= UCA0RXIE;
18.     }
19.
20.     void uart_disable(void)
21.     {
22.             IE2 = 0;
23.     }
24.
25.     void uart_putc(char c)
26.     {
27.     while(!(IFG2&UCA0TXIFG));   // Wait until USCI_A0 TX buffer empty
28.     UCA0TXBUF = c;                  // assign character to TX buffer
29.     }
30.
31.     void uart_puts(const char* s)
32.     {
33.             while(*s != '\0'){
34.     uart_putc(*s);
35.     s++;
36.     }
37.     }
38.
```

```
39.  char uart_data_ready()
40.  {
41.  if(UCA0RXIFG) return 1;
42.  else return 0;
43.  }
44.
45.  /*************************************************************
****
46.  * USCI_A0 RECEIVE INTERRUPT SERVICE ROUTINE
47.  *************************************************************
****/
48.  #pragma vector = USCIAB0RX_VECTOR
49.  __interrupt void USCI0RX_ISR(void)
50.  {
51.  uart_gets(uart_rev.rev_buf);
52.  }
```

❖ ***ADC.h***

```
1.  #ifndef __ADC__H__
2.  #define __ADC__H__
3.
4.  #include <stdint.h>
5.
6.  //Chu y thach anh ngoai khong mo phong bang protues duoc
7.  typedef enum {   ON2_5V,          //Dien ap tham chieu noi 2.5V
8.                   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;
```

```
13.     while(ADC10CTL1 & ADC10BUSY);
14.     ADC10AE0=PORT_ADC;
15.     ADC10CTL0 = ADC10SHT_3 + ADC10ON;
16.     switch(V_tham_chieu)
17.     {
18.     case(ON2_5V):
19.     {
20.     ADC10CTL0|= SREF_1 + REFON + REF2_5V ;
21.     break;
22.     }
23.     case(ON1_5V):
24.     {
25.     ADC10CTL0|= SREF_1 + REFON ;
26.     break;
27.     }
28.     case(VCC):
29.     {
30.     ADC10CTL0|= SREF_0;
31.     break;
32.     }
33.     case(VeREF):
34.     {
35.     ADC10CTL0|= SREF_2 + REFOUT ;
36.     break;
37.     }
38.     }
39.
```

```
40.     //Bien doi tung kenh
41.     ADC10CTL1  = CONSEQ_0;
42.     ADC10DTC0=0;     //Tat DTC
43.     ADC10DTC1=0;
44.     // Cho phep chuyen doi
45.     ADC10CTL0 |= ENC;
46.     // Bat dau chuyen doi
47.     ADC10CTL0 |= ADC10SC;
48.     }
49.
50.     unsigned int ADC10_Read_Channel(unsigned char channel)
51.     {
52.     //Neu kenh doc khac lan cuoi cung doc
53.     if(channel != Reading_channel)
54.     {
55.     //Disable chuyen doi
56.     ADC10CTL0&=~( ADC10SC + ENC );
57.     //Thay doi kenh can chuyen doi-chi dung trong che do don kenh
58.     ADC10CTL1 &= 0x0FFF;
59.     ADC10CTL1|=(channel<<12);
60.     Reading_channel = channel;
61.
62.     ADC10CTL0 &= ~ADC10IFG;              //Tat co ngat
63.     ADC10CTL0|= (ADC10SC + ENC);
64.     //__bis_SR_register(CPUOFF + GIE); // Ngat CPU cho den khi chuyen doi
xong
65.     while(!(ADC10CTL0 & ADC10IFG));    //Cho den khi chuyen doi xong
```

```
66.     return ADC10MEM;                //Lay gia tri ADC
67.     }
68.     else    //Neu channel giong channel da doc lan truoc
69.     {
70.     ADC10CTL0 &= ~ADC10IFG;         //Tat co ngat
71.     ADC10CTL0|= (ADC10SC + ENC);    //Cho phep chuyen doi
72.     //__bis_SR_register(CPUOFF  + GIE); // Ngat CPU cho den khi chuyen doi xong
73.     while(!(ADC10CTL0 & ADC10IFG));         //Cho den khi chuyen doi xong
74.     return ADC10MEM;                //Lay gia tri ADC
75.     }
76.     }
77.
78.     #pragma vector=ADC10_VECTOR
79.     __interrupt void ADC10_ISR(void)
80.     {
81.      __bic_SR_register_on_exit(CPUOFF);        // Clear CPUOFF bit from 0(SR)
82.     }
```

❖ ***Basic_config.h***

```
1.  #ifndef BASIC_CONFIG_H_
2.  #define BASIC_CONFIG_H_
3.
```

```
4.
/*********************************************************************
5.      * DEFINITIONS
6.      *********************************************************************
*********/
7.
8.      // These definitions help you to notice the system clocks
9.      #define MCLK_F 16 // frequency of Master Clock in MHz
10.     #define SMCLK_F 16000000 // frequency of Sub-System Master Clock in
Hz
11.     #define ACLK_F 12000 // frequency of Auxiliary Clock in Hz
12.
/*********************************************************************
13.     * FUNCTIONS 'S PROTOTYPES
14. *********************************************************************
    **********/
15.
16.     void Config_stop_WDT (void);
17.     void Config_Clocks (void);
18.     void delay_us (int t);
19.     void delay_ms (int t);
20.
21.     #endif /* BASIC_CONFIG_H_ */
```

❖ ***Basic_config.c***

```
1.      #include <msp430.h>
2.      #include "Basic_config.h"
```

```
3.
4.
5.  //*************************************************************
*********
6.  // Stop Watch-dog Timer
7.  //*************************************************************
*********
8.  void Config_stop_WDT(void)
9.  {
10.         WDTCTL = WDTPW + WDTHOLD;       // Stop watchdog timer
11. }
12.
13. //*************************************************************
*********
14. // Clocks Configurations
15. //*************************************************************
*********
16. void Config_Clocks(void)
17. {
18. if (CALBC1_16MHZ ==0xFF || CALDCO_16MHZ == 0xFF) // Check if
constants cleared
19. {
20.  while(1);          // If cal constants erased, trap CPU!!
21. }
22.
23. BCSCTL1 = CALBC1_16MHZ;  // Set DCO range & ACLK prescaler
24. DCOCTL = CALDCO_16MHZ;  // Set DCO step + modulation
```

```
25.     BCSCTL3 |= LFXT1S_2 + XCAP_3; // configure ACLK Source
26.     while(IFG1 & OFIFG)                 // wait for OSCFault to clear
27.     {
28.     IFG1 &= ~OFIFG;
29.     __delay_cycles(100000);
30.             }
31.             BCSCTL2 |= SELM_0 + DIVM_0;          // select MCLK,
SMCLK clock and prescaler
32.     //*************************************************************
*********
33.     // Delay for t microseconds
34.     // The delay is not exact when t is too small
35.     //*************************************************************
*********
36.     void delay_us (int t)
37.     {
38.             int i;
39.             for (i = 0; i<t; i++ )
40.                     _delay_cycles(MCLK_F);
41.     }
42.
43.     //*************************************************************
*********
44.     // Delay for t milliseconds
45.     //*************************************************************
*********
46.     void delay_ms (int t)
```

```
47.     {
48.     int i;
49.     for (i = 0; i<t; i++ )
50.     _delay_cycles(MCLK_F*1000);
51.     }
52.
```

2.3.3. Thiết kế bộ điều khiển PID

❖ ***PID.h***

```
1.      #ifndef PID_H_
2.      #define PID_H_
3.
4.      typedef
5.      struct {
6.      float Ki;
7.      float Kp;
8.      float Kd;
9.      float D_part;
10.     float I_part;
11.     float P_part;
12.     float T;
13.     float pre_Out;
14.     float Out;
15.     loat pre_pre_Error;
16.     float pre_Error;
17.     float saturate_h;
18.     float saturate_l;
```

```
19.     float I_part_sat;
20.     float Error;
21.     }
22.     pidparams;
23.
24.     void PID_process(volatile pidparams *pid, float error);
25.     void PID_init(volatile pidparams *pid, float Kp,float Ki,float Kd,
26.                   float saturate_l, float saturate_h,float t,float
I_part_sat);
27.
28.     #endif /* LIB_PID_H_ */
```

❖ ***PID.c***

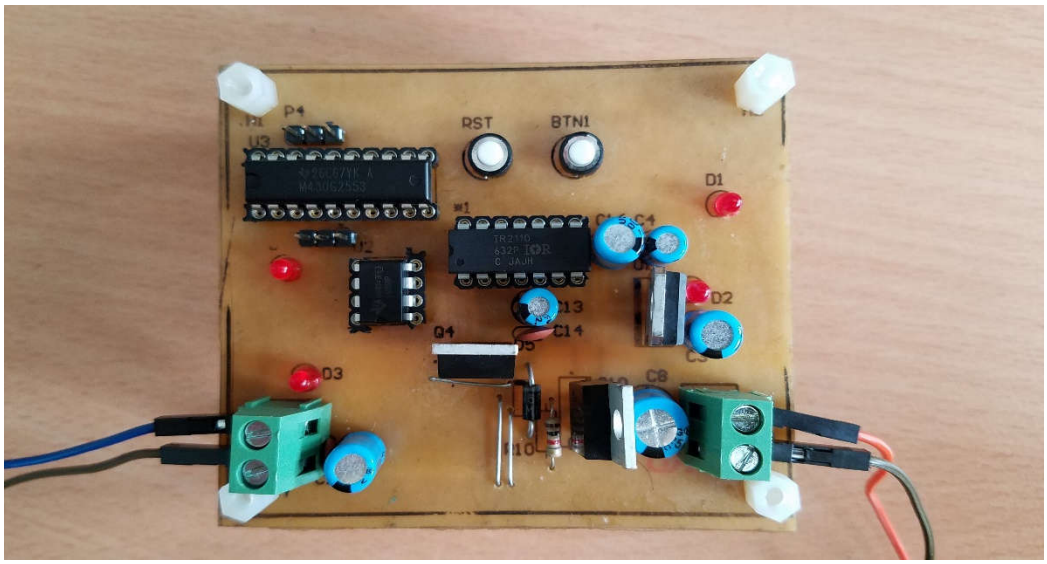
```
1.     #include "pid.h"
2.
3.     void PID_process(volatile pidparams *pid, float error)
4.     {
5.     pid->P_part = pid->Kp * pid->Error;
6.     pid->I_part += pid->Ki * pid->Error / pid->T;
7.     pid->I_part = pid->I_part < pid->I_part_sat ? pid->I_part : pid->I_part_sat;
8.     pid->I_part = pid->I_part > -pid->I_part_sat ? pid->I_part : -pid-
>I_part_sat;
9.     pid->D_part = pid->Kd * (pid->Error - pid->pre_Error) * pid->T;
10.    pid->Out += pid->P_part + pid->I_part + pid->D_part;
11.    pid->pre_Error = pid->Error;
12.
13.    if (pid->Out < pid->saturate_l)
```

```
14.     pid->Out = pid->saturate_l;
15.     else if(pid->Out > pid->saturate_h)
16.     pid->Out = pid->saturate_h;
17.     }
18.     void PID_init(volatile pidparams *pid, float _Kp,float _Ki,float _Kd,
19.     float _saturate_l, float _saturate_h,float _t,float _I_part_sat)
20.     {
21.     pid->Kp = _Kp;
22.     pid->Ki = _Ki;
23.     pid->Kd = _Kd;
24.     pid->saturate_l = _saturate_l;
25.     pid->saturate_h = _saturate_h;
26.     pid->T = _t;
27.     pid->I_part_sat = _I_part_sat;
28.     }
```

CHƯƠNG 3: THÍ NGHIỆM VÀ ĐÁNH GIÁ

3.1. Thí nghiệm đáp ứng ngỏ ra

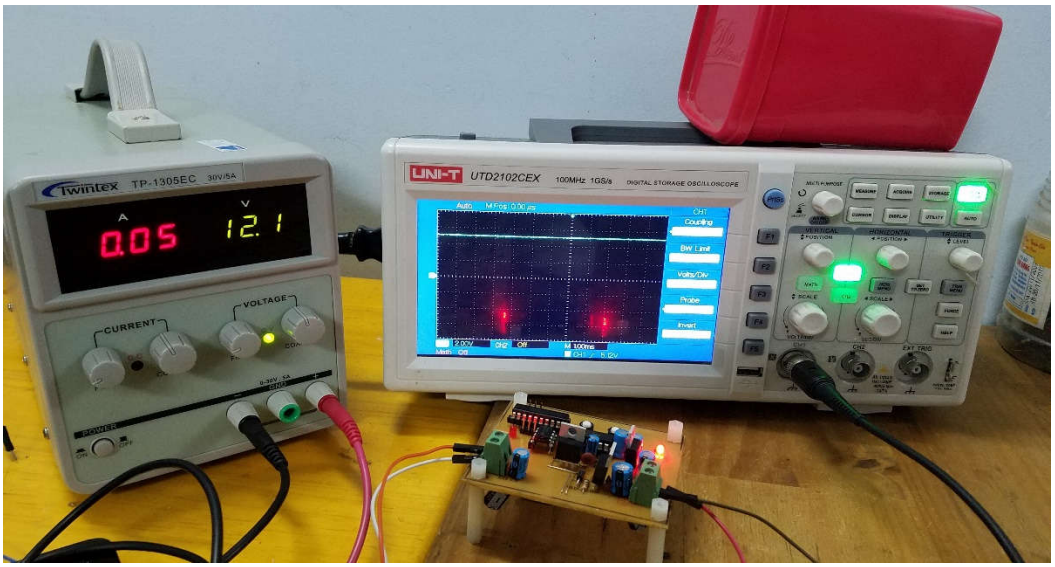
3.1.1. Mô hình mạch BUCK hoàn chỉnh



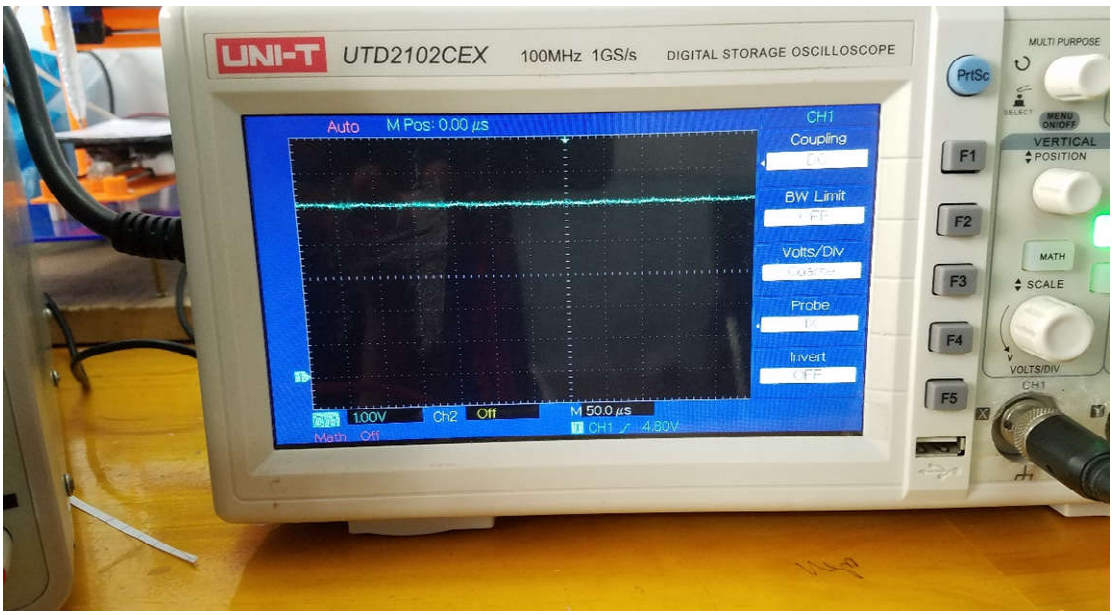
***Hình 3.1**: Mạch hoàn chỉnh*

3.1.2. Kết quả thí nghiệm

- ***Thí nghiệm không tải, V_{in}=12V***

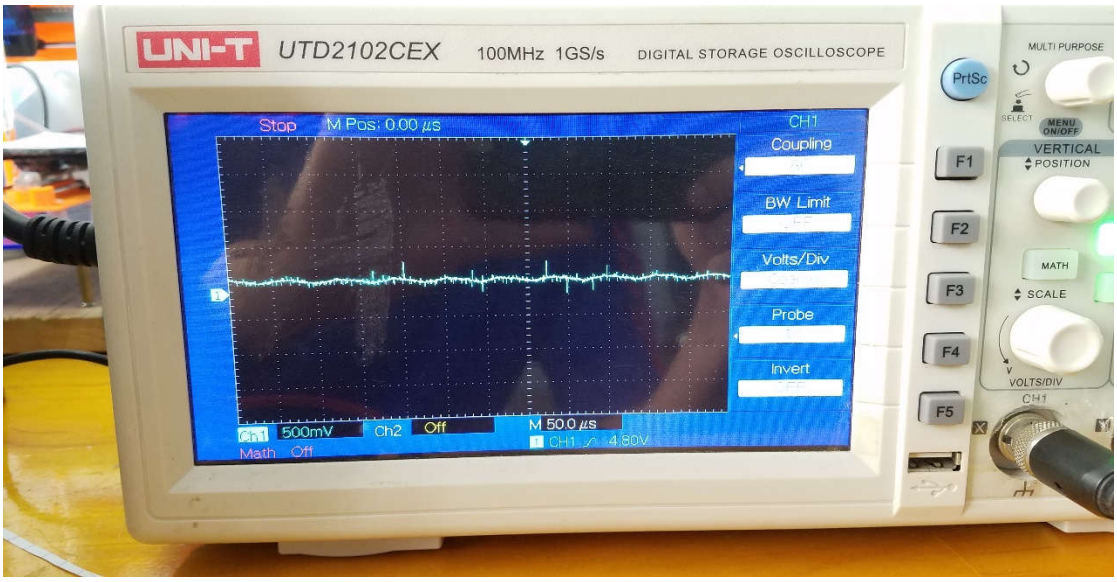


***Hình 3.2**: Thí nghiệm không tải, đo áp ngỏ ra*



***Hình 3.3:** Thí nghiệm không ải, đo áp ngỏ ra V_{out}*

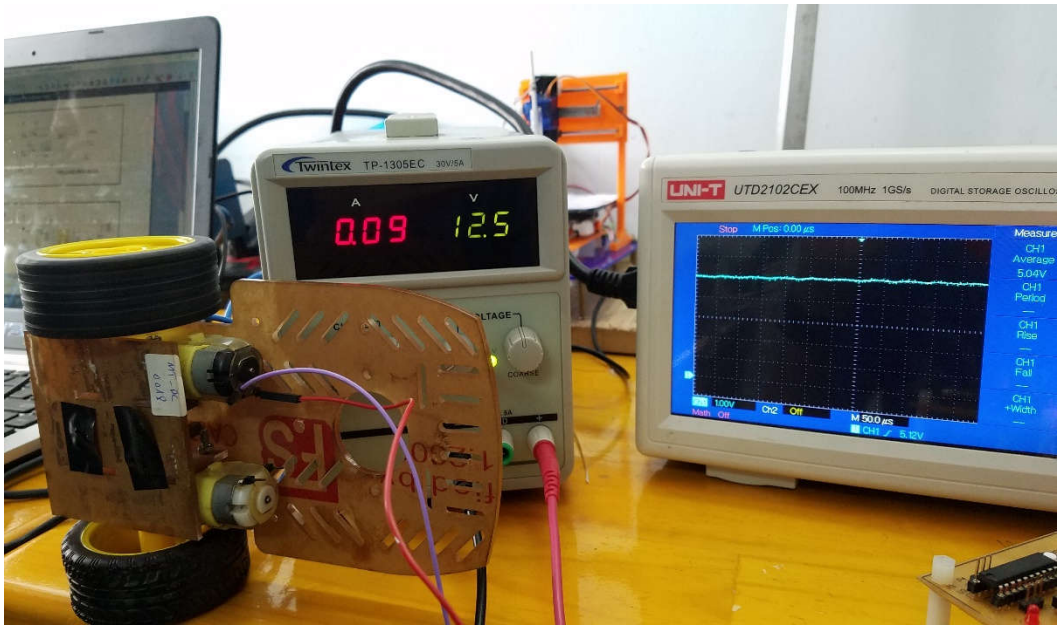
Hình 3.4: *Thí nghiệm không tải, đo các thông số của* V_{out}



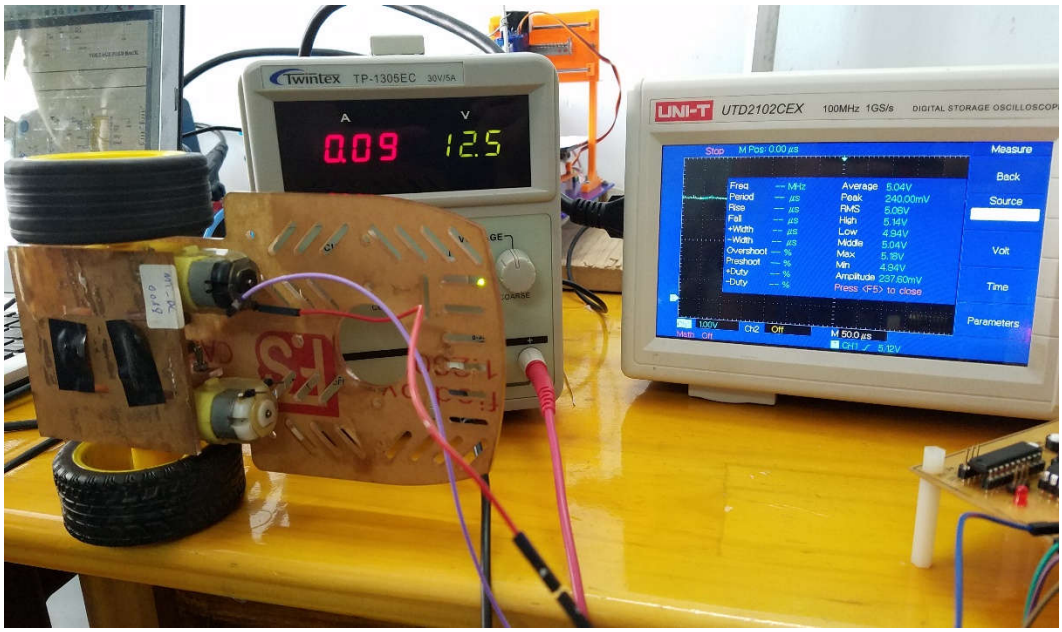
Hình 3.5: *Thí nghiệm không tải, đo thành phần AC của áp ra*

❖ ***Thí nghiệm với tải là động cơ DC***

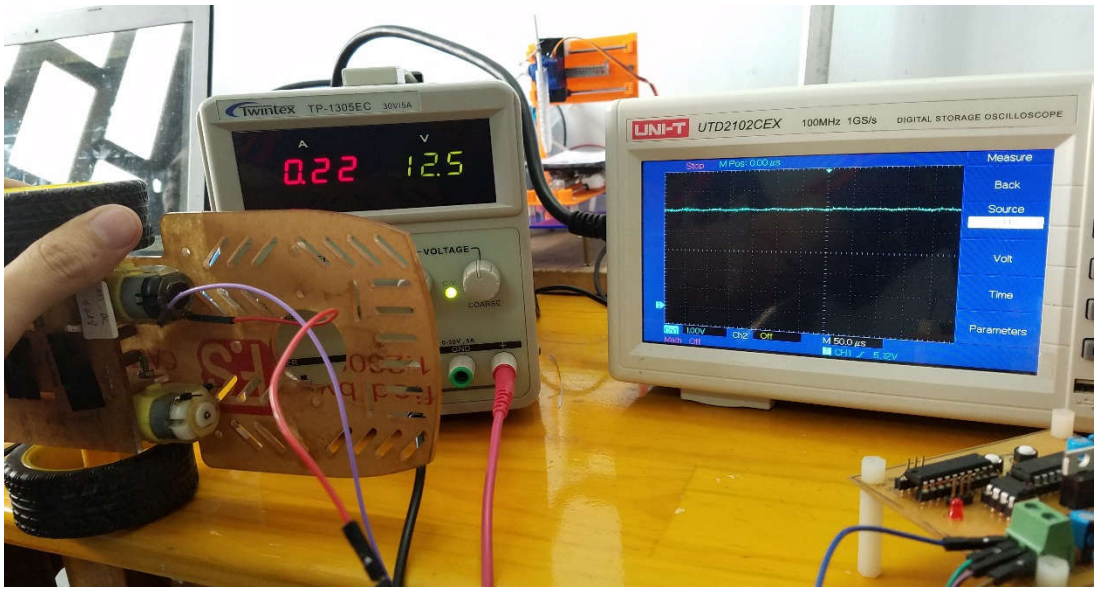
Video kết quả thí nghiệm: https://youtu.be/-o3ZBtGTLWE



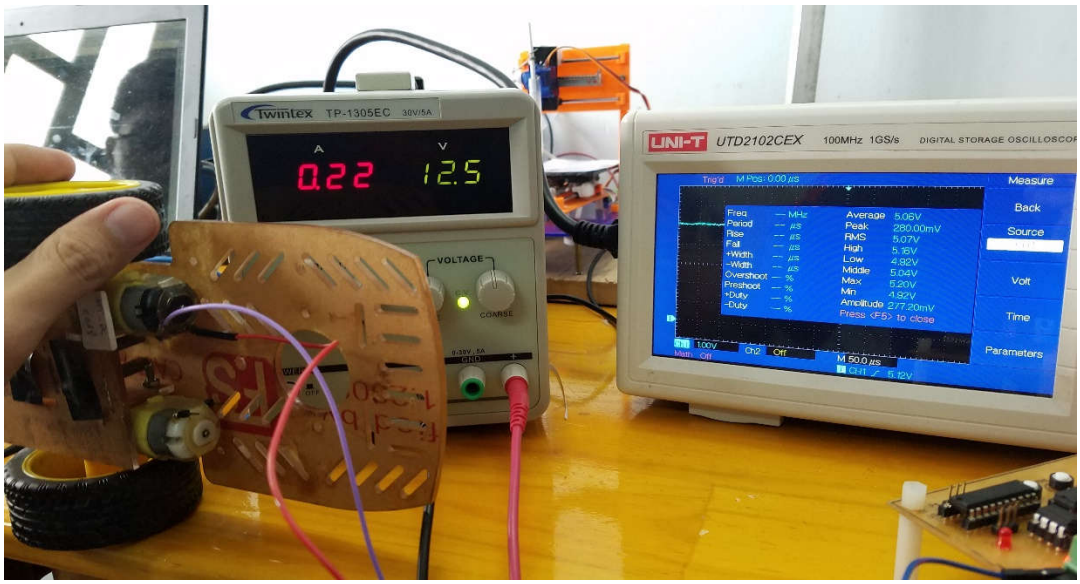
Hình 3.6: Thí nghiệm động cơ không tải, đo áp ngỏ ra



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

3.2. Đánh giá và nhận xét

3.2.1. Tính toán độ nhấp nhô áp ra

- Trường hợp không tải: $\frac{\Delta V_0}{V_0} = \frac{V_{0\_max} - V_{0\_min}}{V_0} = \frac{5.16 - 4.94}{5.04} = 4.37\%$
- Trường hợp động cơ DC không tải: $\frac{\Delta V_0}{V_0} = \frac{5.18 - 4.94}{5.04} = 4.76\%$
- Trường hợp động cơ DC full tải: $\frac{\Delta V_0}{V_0} = \frac{5.20 - 4.92}{5.06} = 5.53\%$

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