HACKADAY.IO

8051 V/f inverter of induction motor

Back to overview

Simple scalar (V/f) inverter of induction motor, using the 8051 in PWM technique


Ezequiel

Back to overview Files 4 Components 0 Logs 5 Instructions 0 Discussion 0

« Back to project details

SORT BY:

Newest

The V/f Strategy

01/05/2017 at 16:30 • 0 comments

Some good articles explaining the speed control of induction motors via V / f strategy are available in:

http://www.electrical4u.com/variable-frequency-drive/

http://controltrends.org/wp-content/uploads/2010/10/VFFundamentals.pdf

In this project, I tried to prevent the microcontroller from performing calculations for voltage and frequency variables. It uses pre-calculated values to load the registers used in the routines, according to the desired output frequency. I will show how these values were obtained (see the spreadsheet with the calculations on the "v-f" tab).

First the frequency. To change the output frequency simply adjust the sine wave refresh rate, ie change the reload value of the timer 0 interrupt counter (variables T_SIN_H and T_SIN_L, which represent the 16-bit word). For more details, refer to the spreadsheet and the 8051 manual, remembering that the timer is in mode 1. The calculated values are shown below.

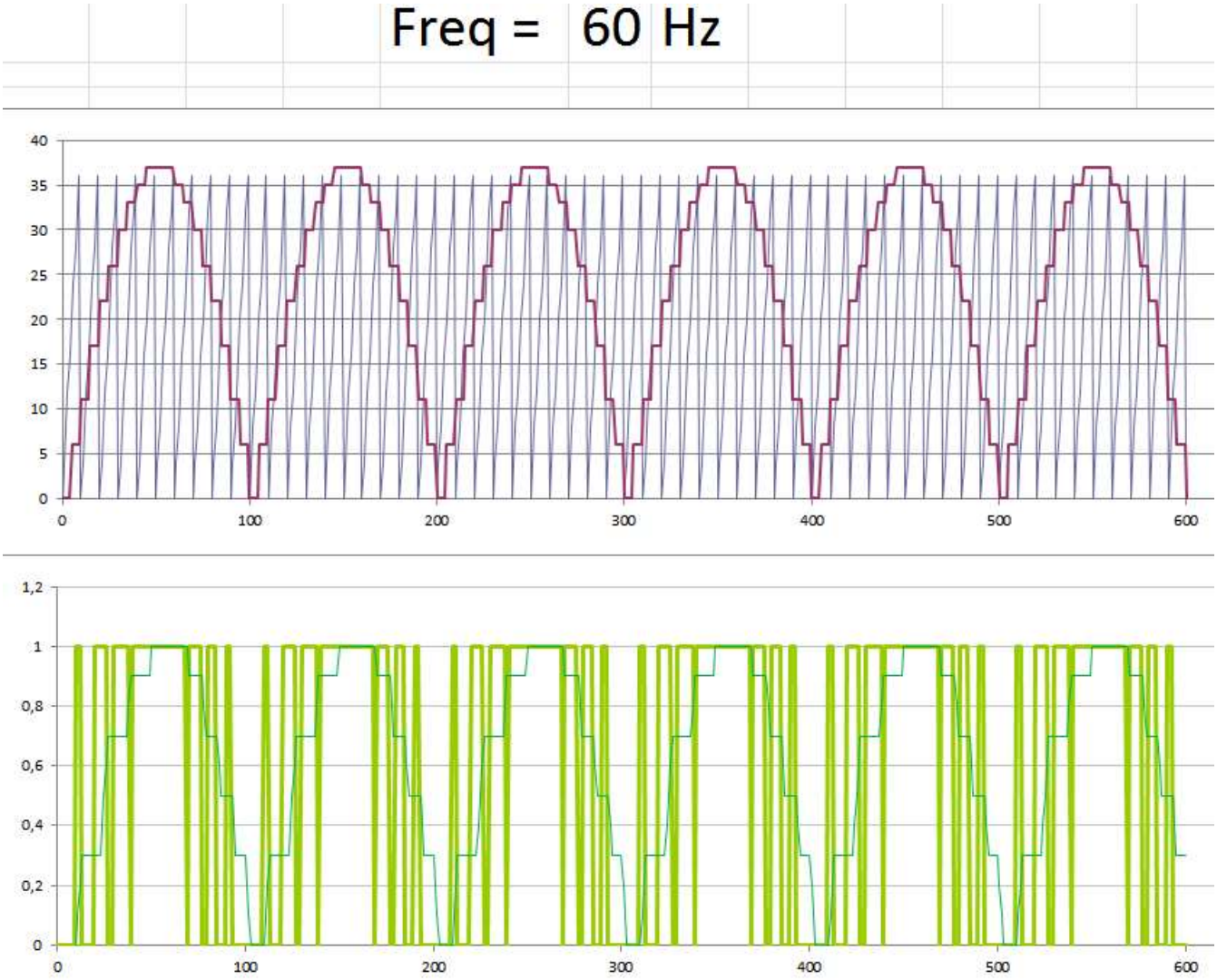
```
;VALUES TO BE RELOADED IN SINE´S TIMER, ACCORDING TO THE FREQUENCY
    T_60 EQU 65152D
    T_50 EQU 65075D
    T_40 EQU 64960D
    T_30 EQU 64768D
    T_20 EQU 64384D
    T_10 EQU 63232D
```

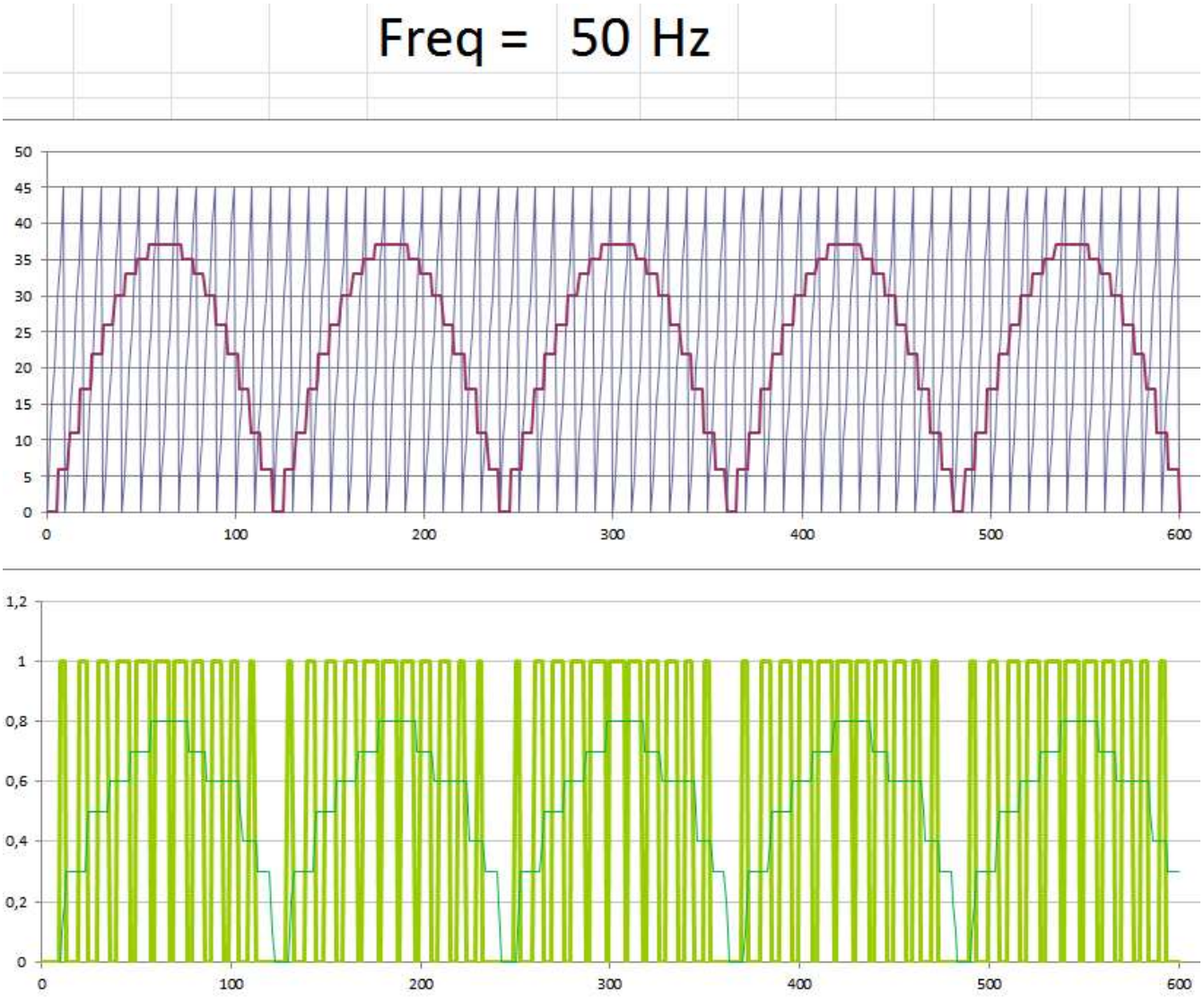
To change the voltage, the option with less calculations in the microcontroller is to vary the amplitude of the sawtooth wave. This is done by changing the step of increasing the timer interrupt 1 (variable STEP_SAW). The variation behavior is inverse, that is, by increasing the wave amplitude, it will be reducing the output voltage.

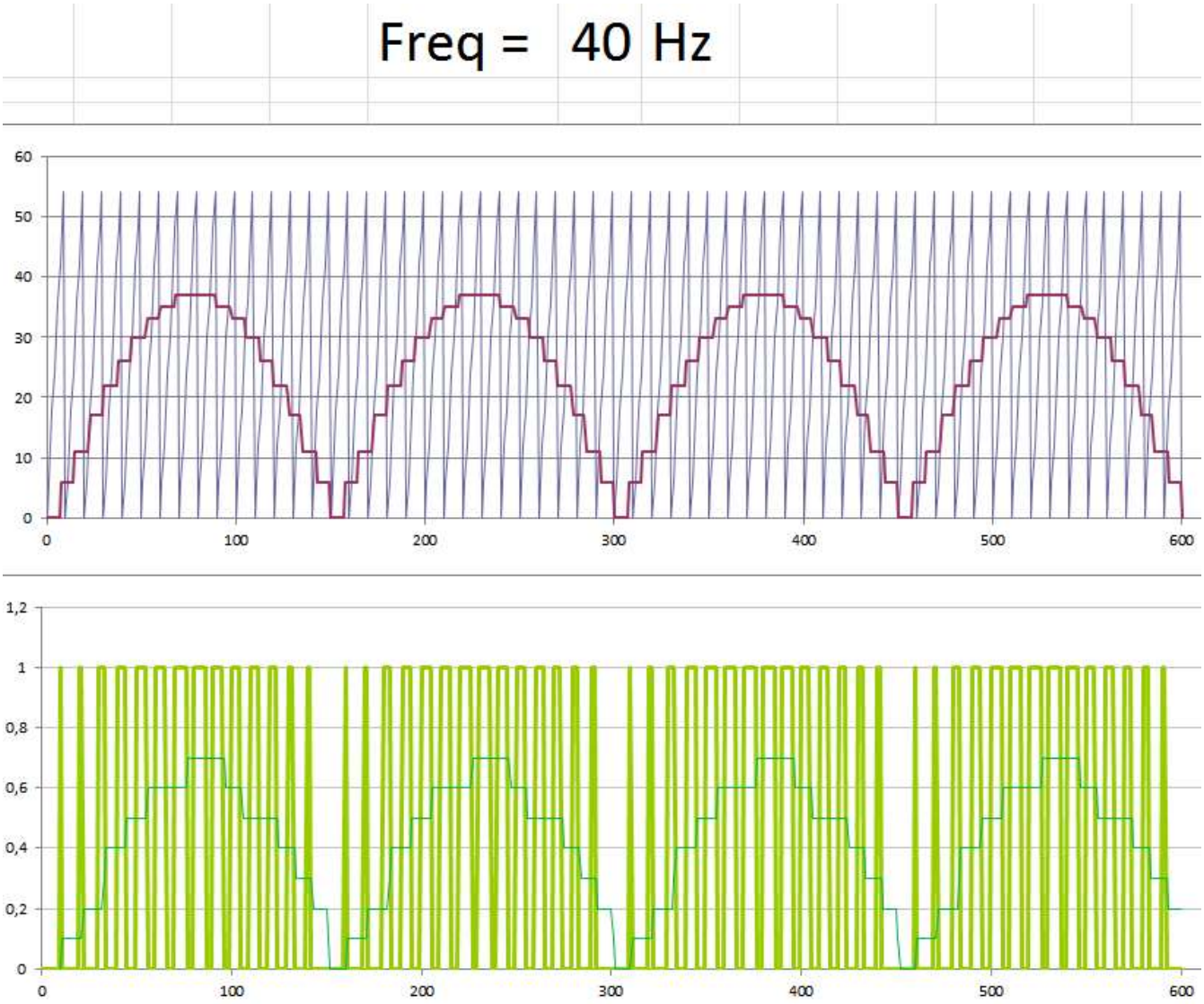
In this case, the challenges were: to find integer increments that produced good resolution in the tension variation; And ensure that the amplitude of the saw is less than or equal to 255 (to fit into an 8-bit word) at all output frequencies. By trial and error method, all parameters had to be adjusted to respect these conditions (including the sine amplitude, 37, remember?). The calculated values are shown below.

```
;VALUES OF THE AMPLITUDE STEP OF THE SAWTOOTH WAVE, ACCORDING TO THE FREQUENCY
        STEP_60 EQU 4D
        STEP_50 EQU 5D
        STEP_40 EQU 6D
        STEP_30 EQU 8D
        STEP_20 EQU 11D
        STEP_10 EQU 23D
```

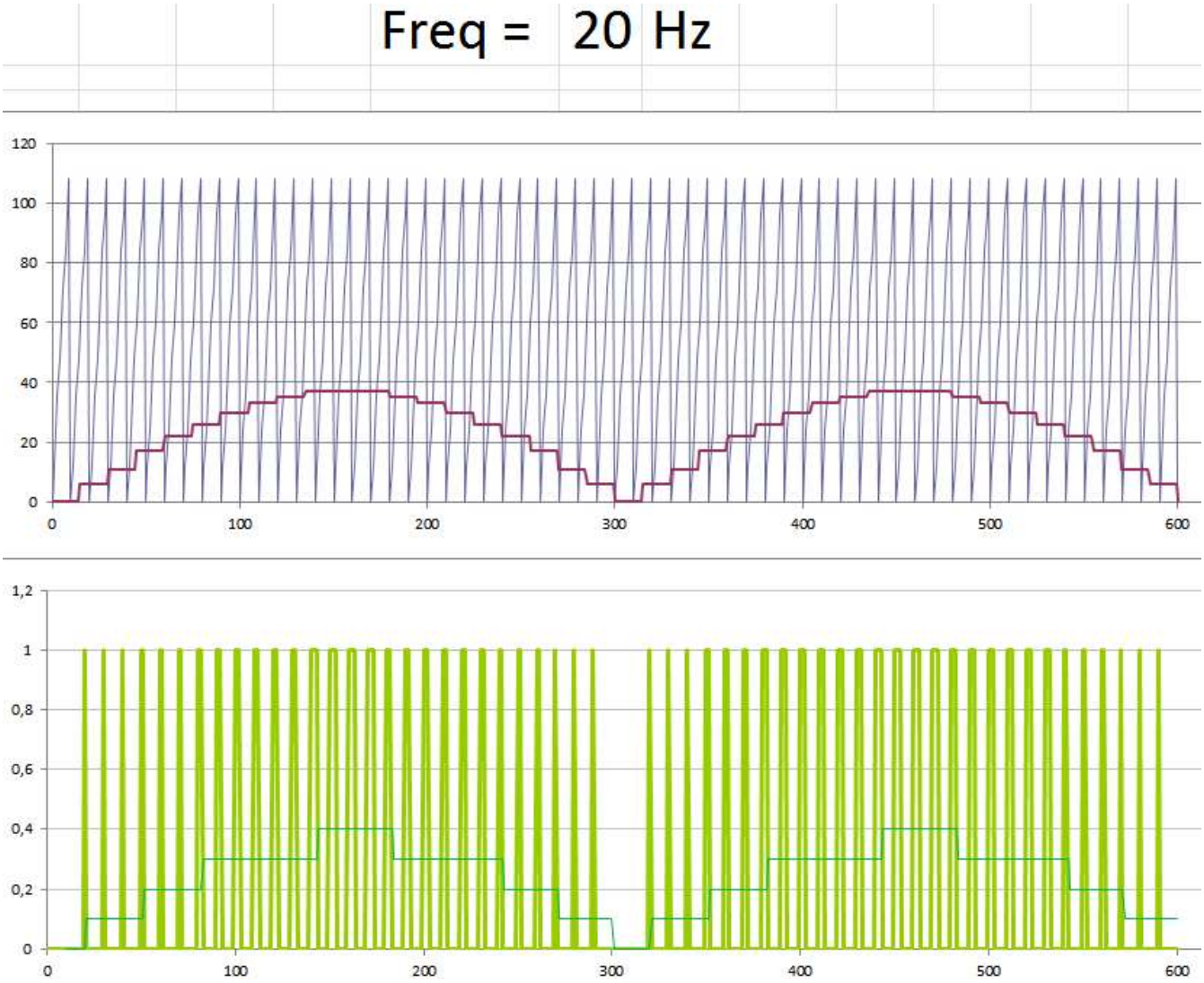
A simulation of the full PWM output can be seen in the "pwm" tab of the worksheet. The following figures show the results (at 10Hz is poor, but acceptable)

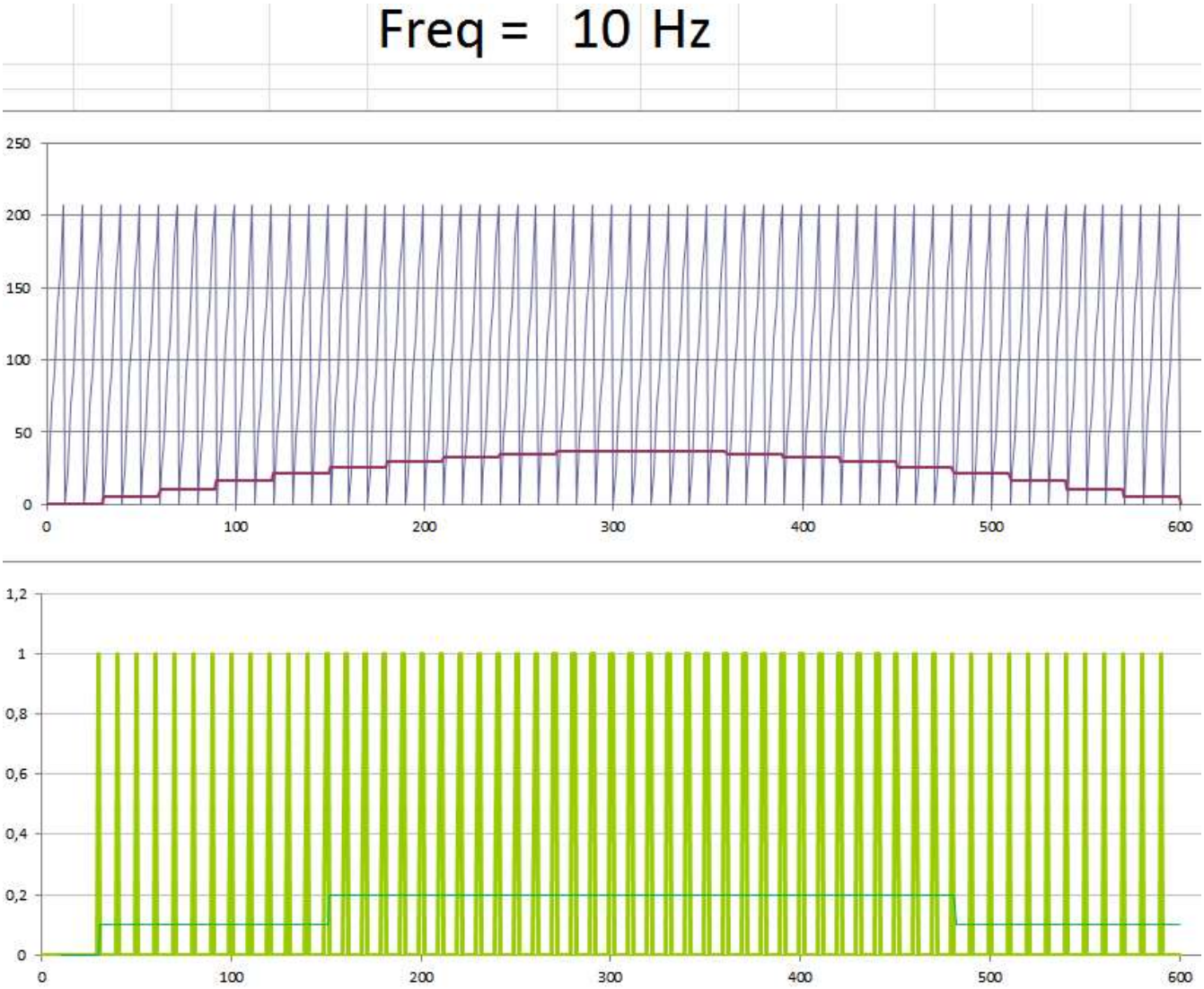

Freq = 60 Hz
40
35
30
25
20
15
10
5
0
0
100
200
300
400
500
600
1,2
1
0,8
0,6
0,4
0,2
0
0
100
200
300
400
500
600


Freq = 50 Hz
50
45
40
35
30
25
20
15
10
5
0
0
100
200
300
400
500
600
1,2
1
0,8
0,6
0,4
0,2
0
0
100
200
300
400
500
600


Freq = 40 Hz
60
50
40
30
20
10
0
0
100
200
300
400
500
600
1,2
1
0,8
0,6
0,4
0,2
0
0
100
200
300
400
500
600


Freq = 30 Hz
80
70
60
50
40
30
20
10
0
0
100
200
300
400
500
600
1,2
1
0,8
0,6
0,4
0,2
0
0
100
200
300
400
500
600


Freq = 20 Hz
120
100
80
60
40
20
0
0
100
200
300
400
500
600
1,2
1
0,8
0,6
0,4
0,2
0
0
100
200
300
400
500
600



The Sine Table

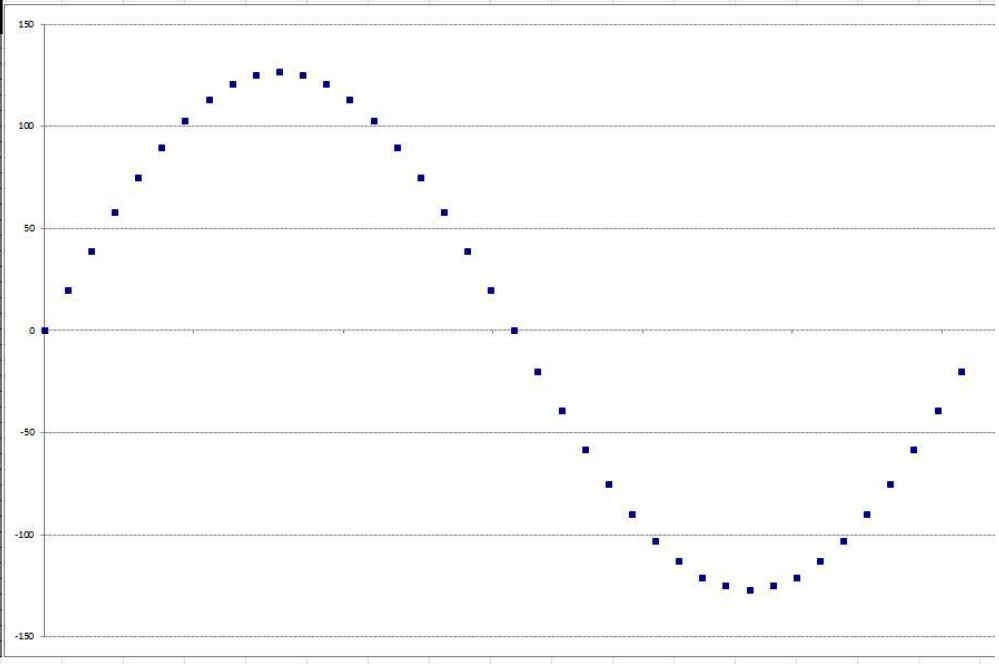
01/04/2017 at 10:54 • 0 comments

To make the sine wave table used in the algorithm shown in the last post, a spreadsheet was used to help calculate the values (this is in the file area, see "sine" tab in calc.xls).

Each half-cycle of the wave was sampled at 20 points, and the amplitude was limited so that the values fit into 8-bit words. The higher bit in the word is the polarity and the other 7 is the value itself, which determines that the amplitude is 127 max. This resulted in a good waveform even with integer values. See the following figure:

points	20
amplitude	127

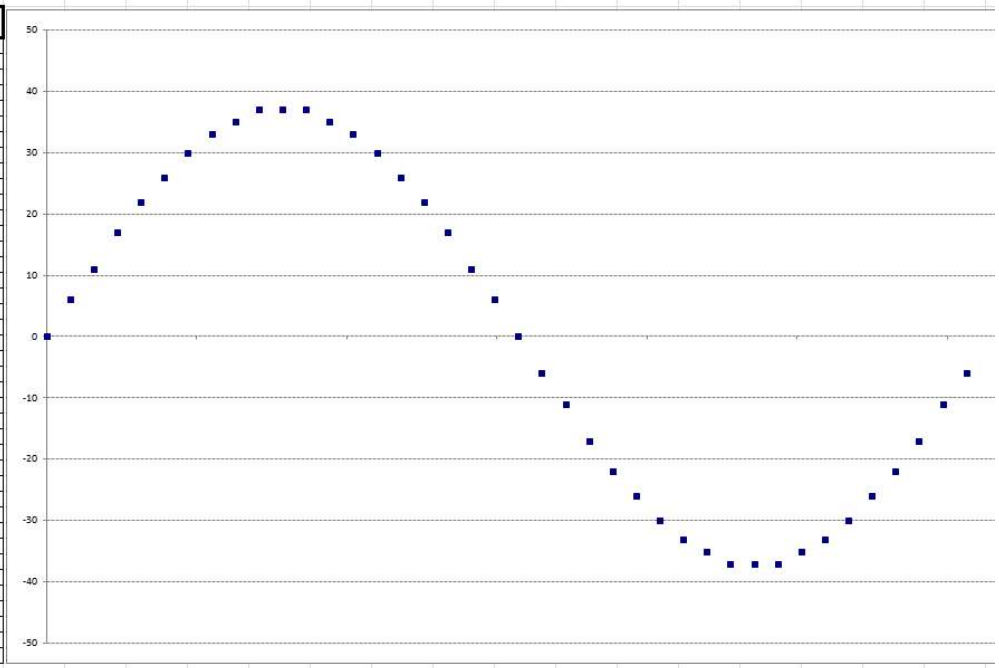
x (pi/points increment)	sin(x)	round to integer	bin	hex
0	0	0	00000000	0
0,157079633	19,86718	20	00010100	14
0,314159265	39,24516	39	00100111	27
0,471238898	57,65679	58	00111010	3A
0,628318531	74,64873	75	01001011	4B
0,785398163	89,80256	90	01011010	5A
0,942477796	102,7452	103	01100111	67
1,099557429	113,1578	113	01110001	71
1,256637061	120,7842	121	01111001	79
1,413716694	125,4364	125	01111101	7D
1,570796327	127	127	01111111	7F
1,727875959	125,4364	125	01111101	7D
1,884955592	120,7842	121	01111001	79
2,042035225	113,1578	113	01110001	71
2,199114858	102,7452	103	01100111	67
2,35619449	89,80256	90	01011010	5A
2,513274123	74,64873	75	01001011	4B
2,670353756	57,65679	58	00111010	3A
2,827433388	39,24516	39	00100111	27
2,984513021	19,86718	20	00010100	14
3,141592654	1,56E-14	0	00000000	0
3,298672286	-19,8672	-20	10010100	94
3,455751919	-39,2452	-39	10100111	A7
3,612831552	-57,6568	-58	10111010	BA
3,769911184	-74,6487	-75	11001011	CB
3,926990817	-89,8026	-90	11011010	DA
4,08407045	-102,745	-103	11100111	E7
4,241150082	-113,158	-113	11110001	F1
4,398229715	-120,784	-121	11111001	F9
4,555309348	-125,436	-125	11111101	FD
4,71238898	-127	-127	11111111	FF
4,869468613	-125,436	-125	11111101	FD
5,026548246	-120,784	-121	11111001	F9
5,183627878	-113,158	-113	11110001	F1
5,340707511	-102,745	-103	11100111	E7
5,497787144	-89,8026	-90	11011010	DA
5,654866776	-74,6487	-75	11001011	CB
5,811946409	-57,6568	-58	10111010	BA
5,969026042	-39,2452	-39	10100111	A7
6,126105675	-19,8672	-20	10010100	94



To implement the V/f control strategy (we'll see later), the wave amplitude had to be set to 37 max (trial and error). But this did not significantly impair the waveform. See the following figure:

points	20
amplitude	37

x (pi/points increment)	sin(x)	round to integer	bin	hex
0	0	0	00000000	0
0,157079633	5,788075	6	00000110	6
0,314159265	11,43363	11	00001011	B
0,471238898	16,79765	17	00010001	11
0,628318531	21,74805	22	00010110	16
0,785398163	26,16295	26	00011010	1A
0,942477796	29,93363	30	00011110	1E
1,099557429	32,96724	33	00100001	21
1,256637061	35,18909	35	00100011	23
1,413716694	36,54447	37	00100101	25
1,570796327	37	37	00100101	25
1,727875959	36,54447	37	00100101	25
1,884955592	35,18909	35	00100011	23
2,042035225	32,96724	33	00100001	21
2,199114858	29,93363	30	00011110	1E
2,35619449	26,16295	26	00011010	1A
2,513274123	21,74805	22	00010110	16
2,670353756	16,79765	17	00010001	11
2,827433388	11,43363	11	00001011	B
2,984513021	5,788075	6	00000110	6
3,141592654	4,53E-15	0	00000000	0
3,298672286	-5,78808	-6	10000110	86
3,455751919	-11,4336	-11	10001011	8B
3,612831552	-16,7976	-17	10010001	91
3,769911184	-21,7481	-22	10010110	96
3,926990817	-26,163	-26	10011010	9A
4,08407045	-29,9336	-30	10011110	9E
4,241150082	-32,9672	-33	10100001	A1
4,398229715	-35,1891	-35	10100011	A3
4,555309348	-36,5445	-37	10100101	A5
4,71238898	-37	-37	10100101	A5
4,869468613	-36,5445	-37	10100101	A5
5,026548246	-35,1891	-35	10100011	A3
5,183627878	-32,9672	-33	10100001	A1
5,340707511	-29,9336	-30	10011110	9E
5,497787144	-26,163	-26	10011010	9A
5,654866776	-21,7481	-22	10010110	96
5,811946409	-16,7976	-17	10010001	91
5,969026042	-11,4336	-11	10001011	8B
6,126105675	-5,78808	-6	10000110	86



Then the final table in the code:

```
;======== SINE WAVE TABLE ==================
;FULL CYCLE OF SINE WAVE, SAMPLED AT 40 POINTS
;MOST SIGNIFICANT BIT INDICATES NEGATIVE POLARITY
```

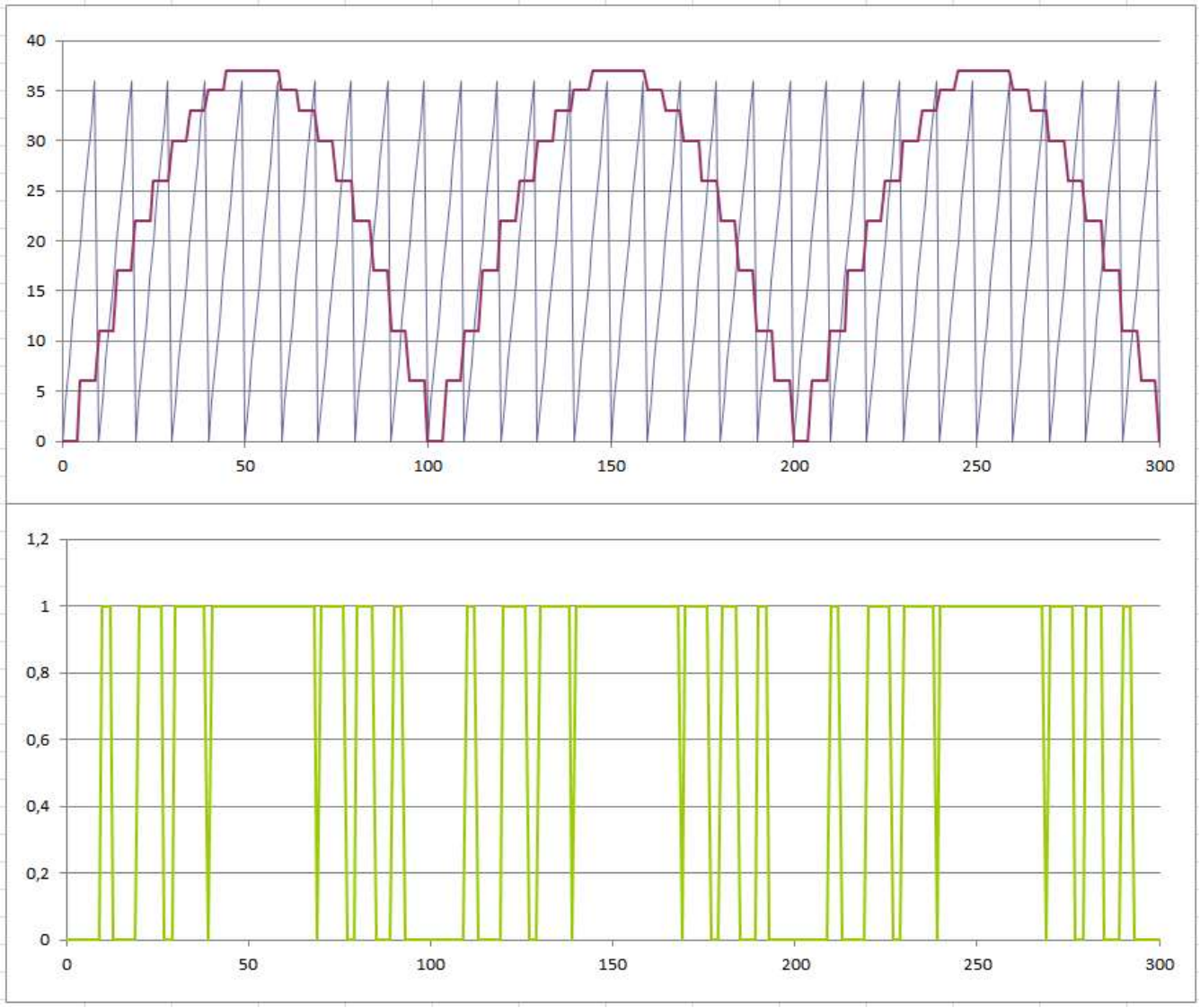
```
SINE_WAVE:
        DB 0,6,11,17,22,26,30,33,35,37,37,37,35,33,30,26,22,17,11,6,0,134,1

;=====================================
```

The PWM

01/02/2017 at 16:07 • 0 comments

A good article that explains in detail the PWM strategy is available on Wikipedia at https://en.wikipedia.org/wiki/Pulse-width_modulation.

For now, the strategy adopted in this project is to compare the desired output signal (sine wave) with a reference signal, a sawtooth wave. When the latter is below the generator signal, the PWM is in high state (1). Otherwise it is in the low state (0). See the following figure.



It is not necessary to do this for the complete sine wave, just a half-cycle because of its symmetry. To achieve the negative half-cycle, one bit will toggle the switching sequence of H-bridge.

The 8051 does not have any facility to generate PWM outputs, so the strategy described should be made entirely in the code. To generate the sine and sawtooth waves, the two 8051 timers and their interrupts will be used.

The sine wave uses the lookup table technique. Let's see the code:

```
T0_INT:
        CLR TR0                 ;STOP THE TIMER
        MOV DPTR, #SINE_WAVE
        MOV A, COUNT_SIN
        MOVC A, @A+DPTR         ;SEE THE VALUE OF COUNT_SIN INDEX IN THE SI
        MOV SINE, A             ;AND STORE IN SINE VARIABLE
```

```
        MOV C, SINE.7           ;HIGH BIT OF VALUE IS THE POLARITY
        MOV P1.1,C              ;SEND IT TO THE OWN OUTPUT
        CLR SINE.7              ;AND CLEAR IT

        INC COUNT_SIN           ;GO TO NEXT VALUE
        MOV A, COUNT_SIN

        CJNE A, #40, QUIT_T0    ;CHECK IF TABLE ARE AT THE END
        MOV COUNT_SIN, #0       ;THEN, RESTART THE TABLE COUNT

QUIT_T0:
        MOV     TL0, T_SIN_L    ;RELOAD THE TIMER COUNTER
        MOV     TH0, T_SIN_H
        SETB TR0                ;RESTART THE TIMER
        RETI
```

Each time the interrupt is called, the current value of the wave is updated by querying a pre-calculated table. Considering that the time interval is equal, the wave is generated by the sequencing of values.

For the sawtooth wave the procedure is simpler: just increment the current value with constant step until it reaches the value of its period (10 steps), restarting the sequence. As the frequency is higher, it is possible to use the 8051 timer in 8-bit auto reload mode, simplifying the code. See the code:

```
T1_INT:
        ;THE SAWTOOTH WAVE GENERATION
        MOV A, SAWTOOTH
        ADD A, STEP_SAW         ;INCREMENT THE SAW WIHT CURRENT STEP
        MOV SAWTOOTH, A

        DJNZ COUNT_SAW, QUIT_T1 ;CHECK IT'S DONE
        MOV SAWTOOTH, #0        ;THEN, RESTART THE WAVE
        MOV COUNT_SAW, #10D     ;AND RESET ITS COUNTER
QUIT_T1:
        RETI
```

Finally, the PWM output compares the two signals and the result will be the carry bit:

```
        MOV A,SAWTOOTH          ;COMPARES THE SAW WAVE
        SUBB A,SINE             ;VERSUS THE SINE WAVE
```

```
        MOV P1.0,C              ;<==== PMW OUTPUT
```

Putting everything together (PWM shares the sawtooth timer):

```
;===== SINE WAVE GENERATION =======
;THE SINE WAVE IS USED AS A COMPARATIVE REFERENCE IN THE PWM OUTPUT.
;THE SINE VALUE IS PERIODICALLY UPDATED BY T0 INTERRUPTION CALL,
;VIA LOOKUP TABLE TECHNIQUE. THE TIME OF INTERRUPTION VARIES,
;DEPENDING ON THE SELECTED OUTPUT FREQ.

T0_INT:
        CLR TR0                 ;STOP THE TIMER
        MOV DPTR, #SINE_WAVE
        MOV A, COUNT_SIN
        MOVC A, @A+DPTR         ;SEE THE VALUE OF COUNT_SIN INDEX IN THE SI
        MOV SINE, A             ;AND STORE IN SINE VARIABLE

        MOV C, SINE.7           ;HIGH BIT OF VALUE IS THE POLARITY
        MOV P1.1,C              ;SEND IT TO THE OWN OUTPUT
        CLR SINE.7              ;AND CLEAR IT

        INC COUNT_SIN           ;GO TO NEXT VALUE
        MOV A, COUNT_SIN

        CJNE A, #40, QUIT_T0    ;CHECK IF TABLE ARE AT THE END
        MOV COUNT_SIN, #0       ;THEN, RESTART THE TABLE COUNT

QUIT_T0:
        MOV     TL0, T_SIN_L    ;RELOAD THE TIMER COUNTER
        MOV     TH0, T_SIN_H
        SETB TR0                ;RESTART THE TIMER
        RETI

;===== SAW WAVE GENERATION  AND PWM OUTPUT =======
;THE FREQUENCY OF SAWTHOOTH WAVE IS FIXED AT 1200Hz,
;AND THIS SIGNAL IS GENERATE WITH 10 STEPS.

T1_INT:
        ;FIRST, THE PWM!
        MOV A,SAWTOOTH          ;COMPARES THE SAW WAVE
```

```
        SUBB A,SINE             ;VERSUS THE SINE WAVE
        MOV P1.0,C              ;<==== PMW OUTPUT

        ;CONTINUE THE SAWTOOTH WAVE GENERATION
        MOV A, SAWTOOTH
        ADD A, STEP_SAW         ;INCREMENT THE SAW WIHT CURRENT STEP
        MOV SAWTOOTH, A

        DJNZ COUNT_SAW, QUIT_T1 ;CHECK IT'S DONE
        MOV SAWTOOTH, #0        ;THEN, RESTART THE WAVE
        MOV COUNT_SAW, #10D     ;AND RESET ITS COUNTER
QUIT_T1:
        DEC R0                  ;R0 FOR BUTTON DEBOUNCE ROUTINE
        RETI
```

Code and Schematics

12/31/2016 at 13:30 • 0 comments

The full source is available in the file area and also in GitHub, and is well commented (the translation may not be very good). The code is fully functional, but may change in the future.

Circuit schematics are also available. The control circuit is based on the AT89C2051 which for me is quite easy to use and assemble. But the code is fully compatible with the 8051 architecture and can be used on other platforms and manufacturers.

The drive circuit is based on H-bridge with MOSFETs. I have not yet built it, so I think the components may vary a bit.

Starting

12/31/2016 at 13:10 • 0 comments

Hi,

I will try to explain the project in detail in these next logs.

First, for this project you do not need complex tools, just a text editor and an 8051 assembler (there are several available on the internet). I chose to use a complete IDE that has these tools and also a useful simulator. It is the 8051 IDE MCU, which is free, available at http://mcu8051ide.sourceforge.net

Going up?