Laboratorio de Microprocesadores - 86.07

Coctelera

Profesor:			Ing. Guillermo Campiglio								
Cuatrimestre/Año:			2º/2019								
Turno de las clases prácticas			Miércoles								
Jefe de trabajos prácticos:			Ing. Pedro Ignacio Martos								
Docente guía:			Ing. Fabricio Baglivo								
Autores			Seguimiento del proyecto								
Nombre	Apellido	Padrón									
Ignacio	Piperno	100 677									
Agustín	D'Amico	100 678									

Observaciones:

.....
.....
.....
.....
.....
.....
.....
.....
.....

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Introducción **1**

2. Objetivos propuestos y realizados **1**

3. Descripción del Hardware **2**

4. Descripción del Software **2**
4.1. Diagrama de flujo 3

5. Conclusiones y posibles mejoras **6**

6. Código **7**

7. Apéndice **25**

1. Introducción

El proyecto realizado consta de una máquina expendedora de líquidos, que tendrá lugar para dispensar dos fluídos. Mediante bombas sumergibles, transportará un volumen (previamente determinado por el usuario) del primer líquido dentro de un recipiente y luego, automáticamente volcará el segundo líquido hasta llenarlo. La máquina servirá un volumen total fijo de 300 ml.

Para poder determinar los límites de volumen de líquido, tanto el designado por el usuario como el del recipiente completo, se utilizó una balanza para medir el peso. Por lo tanto, cuando la balanza mida que se llego al peso deseado, dependiendo el caso, la máquina cambiará de fluido o terminará el proceso.

Se utilizó el microcontrolador **AT-Mega328p** como herramienta de control, es decir, es el encargado de: comunicarse con el usuario para saber cuánto volumen del primer líquido se verterá sobre el recipiente, comunicarse con la balanza para detectar cuando debe dejar de dispensar líquido y por lo tanto, tendrá el poder de abrir y cerrar las bombas cuando sea oportuno, entre otras más funciones.

2. Objetivos propuestos y realizados

Se buscó en este trabajo desarrollar un proyecto mediante el uso de un microcontrolador **AT-Mega328p**. Como requisitos mínimos el proyecto debía contar con las siguientes especificaciones:

- El programa debe realizar una acción a partir de información brindada por algún tipo de sensor.
- El software debe contar con algún tipo de interrupción.

Con estas consideraciones, se decidió realizar una máquina dispensadora de tragos controlada por peso.

Si bien los objetivos propuestos pudieron realizarse, una idea que no pudo llevarse a cabo es poder servir un volumen de líquido variable dependiendo del volumen del recipiente a llenar. Entre varias opciones, se propuso utilizar un sensor ultrasonido o un sensor láser para poder detectar el momento en el que el líquido conseguía cierta altura. Pero debido al tiempo que dichas implementaciones tomarían y debido a que no eran del todo confiables (el ultrasonido podía ser afectado por ruidos externos y el láser podía tener inconvenientes si se utilizaban bebidas transparentes), se decidió que la máquina dispense un volumen fijo total de 300 ml.

3. Descripción del Hardware

En la figura 1 se puede observar el diagrama en bloques del hardware utilizado.

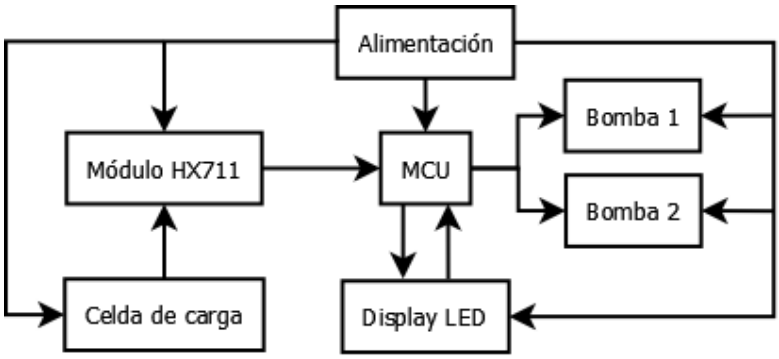


Figura 1: Diagrama en bloques de hardware

Para poder simplificar el proyecto, se utilizó una placa *Arduino UNO*, ya que dentro de ella contiene el microcontrolador **AT-Mega328p**, pero no se utilizó su IDE, se programó directamente el microcontrolador en lenguaje Assembler.

Además se utilizó el módulo *HX711* para la comunicación entre la celda de carga y el microcontrolador. El módulo consiste de un conversor analógico digital de 24 bits con una etapa amplificadora para poder transmitir la información de la celda de carga al micro mediante comunicación del tipo serie.

Entonces, la máquina consta de una placa *Arduino UNO*, de una celda de carga que cumplirá la función de balanza para poder medir el peso del recipiente y del líquido vertido, del módulo *HX711* que se utilizará para comunicar la celda de carga con el micro, de bombas de agua que serán controladas por el microcontrolador utilizando transistores **TIP41** como llaves para transportar el liquido mediante mangueras desde recipiente hacia el vaso. Para estas llaves electrónicas se utilizaron resistencias de 330 Ω de manera que las bombas tuvieran corriente suficiente para prender, y se utilizara la menor cantidad de corriente de los pines del micro.

Además, sobre la pared frontal del artefacto, se ubicará una serie de LEDs para que el usuario pueda escoger, mediante el color de los LED indicadores, el porcentaje de volumen del primer líquido a dispensar. Dicho volumen irá del 10 % al 50 % sobre el total, con saltos del 10 %.

En el apéndice se encuentra el circuito esquemático del proyecto.

4. Descripción del Software

Con el fin de poder controlar la información que recibe el **AT-Mega328p**, se utilizó un *timer* para poder obtener las mediciones de la balanza. Además de esto, el uso de interrupciones por timer permite entrar en **sleep mode** cuando el micro está inactivo, lo que permite un ahorro de energía. El microcontrolador se encuentra en modo sleep y sólo se enciende a partir de las interrupciones provocadas por el timer, por lo que se ejecuta el main con una frecuencia de $\frac{16\ MHz}{64 \cdot 65535} \simeq 4\ Hz$, lo que no interfiere con la salida de datos del conversor **HX711**, que tiene una frecuencia de salida de datos de 10 Hz. Para cada medición tomada, se procesa el dato obtenido de manera de discernir si se sensó un pulso, si se sensó un vaso, o si se sirvieron las cantidades de líquido deseadas.

Inicialmente el usuario seleccionará el volumen del primer líquido a expender dándole "toques" a la balanza, por lo que por cada golpe suave sobre la balanza, el microcontrolador prenderá LEDs contiguos hasta que se llegue al volumen deseado por el consumidor (si todos los LEDs están encendidos, el único que quedará activado es el primero). Debido a que el MC recibe muestras a una velocidad constante, dicha implementación se realizó simplemente contando la cantidad de muestras que recibía el microcontrolador y fijando un umbral de detección. Es decir, si el microcontrolador detecta una medición mayor al umbral de detección y fue de una duración entre cierta cantidad de muestras (correspondientes a un toque), quiere decir que fue un "toque" y por lo tanto debe prender el siguiente LED del display, en cambio si el micro lee más de cierta cantidad de muestras (que corresponden a la detección de un vaso) mayores al umbral, el micro considerará que se trata de un vaso, y deberá comenzar a dispensar el primer líquido.

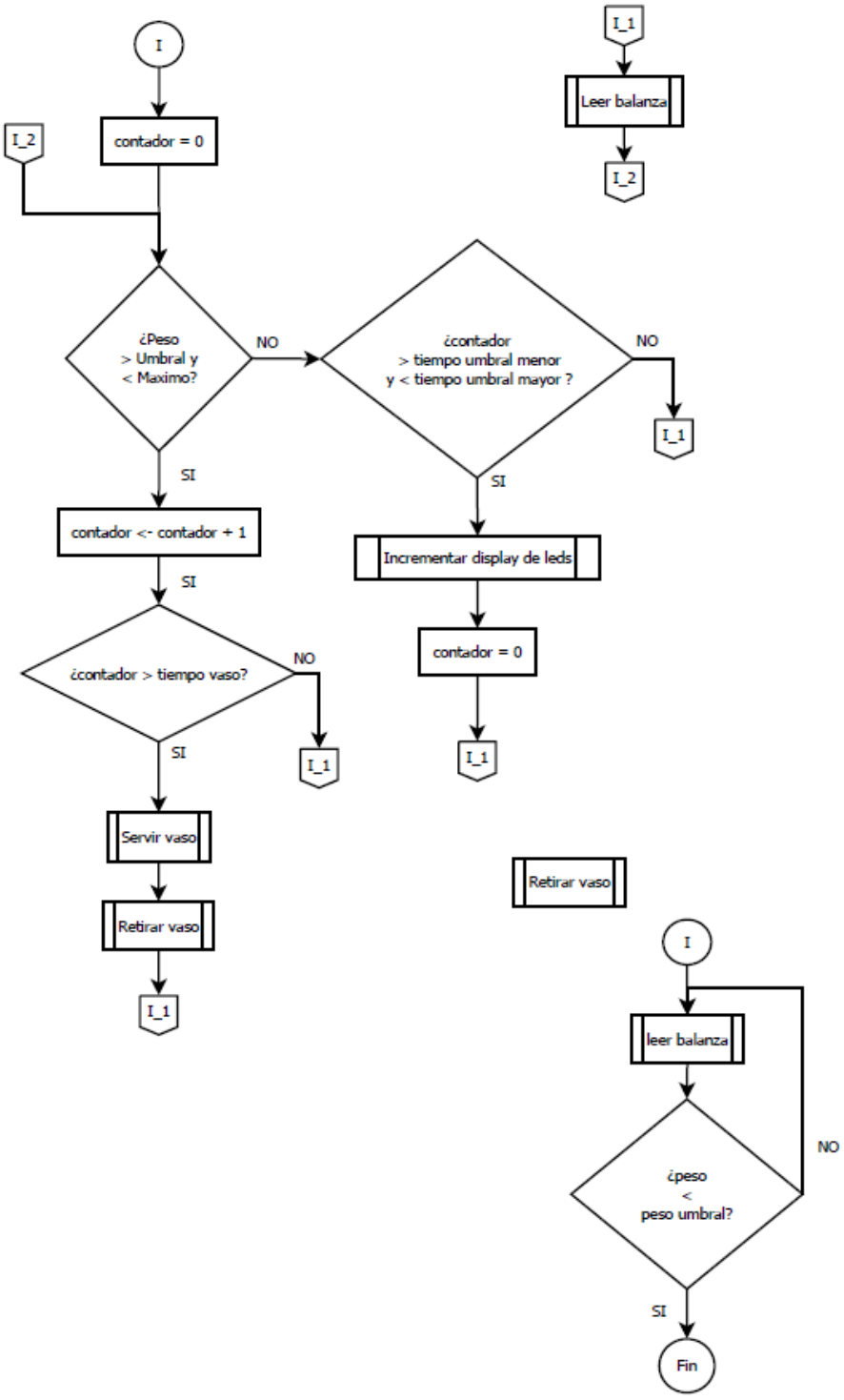
Luego de que se haya fijado el volumen a verter del primer fluido, el usuario deberá apoyar el recipiente sobre la balanza y cuando el microcontrolador detecte que se apoyó por más de la cantidad de muestras para un vaso, este activará la bomba que expenderá dicho líquido, hasta que la balanza mida un aumento de peso igual al porcentaje determinado por el usuario, por lo que apagará la bomba. Posteriormente, el microcontrolador encenderá la bomba contigua y volcará el segundo líquido hasta que el recipiente este completamente lleno, finalizando el proceso.

Algunas adiciones que se hicieron fueron:

- Si se coloca un recipiente con un peso mayor a un peso límite máximo, la máquina no expenderá ningún líquido.
- Si se retira el recipiente mientras se está vertiendo el líquido, el micro detendrá el proceso y dejará de verter líquido.

4.1. Diagrama de flujo

En las siguientes figuras se encuentra el diagrama de flujo del programa implementado.


I
I_2
contador = 0
¿Peso
> Umbral y
< Maximo?
NO
SI
contador <- contador + 1
SI
¿contador > tiempo vaso?
NO
SI
Servir vaso
Retirar vaso
I_1
I_1
Leer balanza
I_2
¿contador
> tiempo umbral menor
y < tiempo umbral mayor ?
NO
I_1
SI
Incrementar display de leds
contador = 0
I_1
Retirar vaso
I
leer balanza
¿peso
<
peso umbral?
NO
SI
Fin

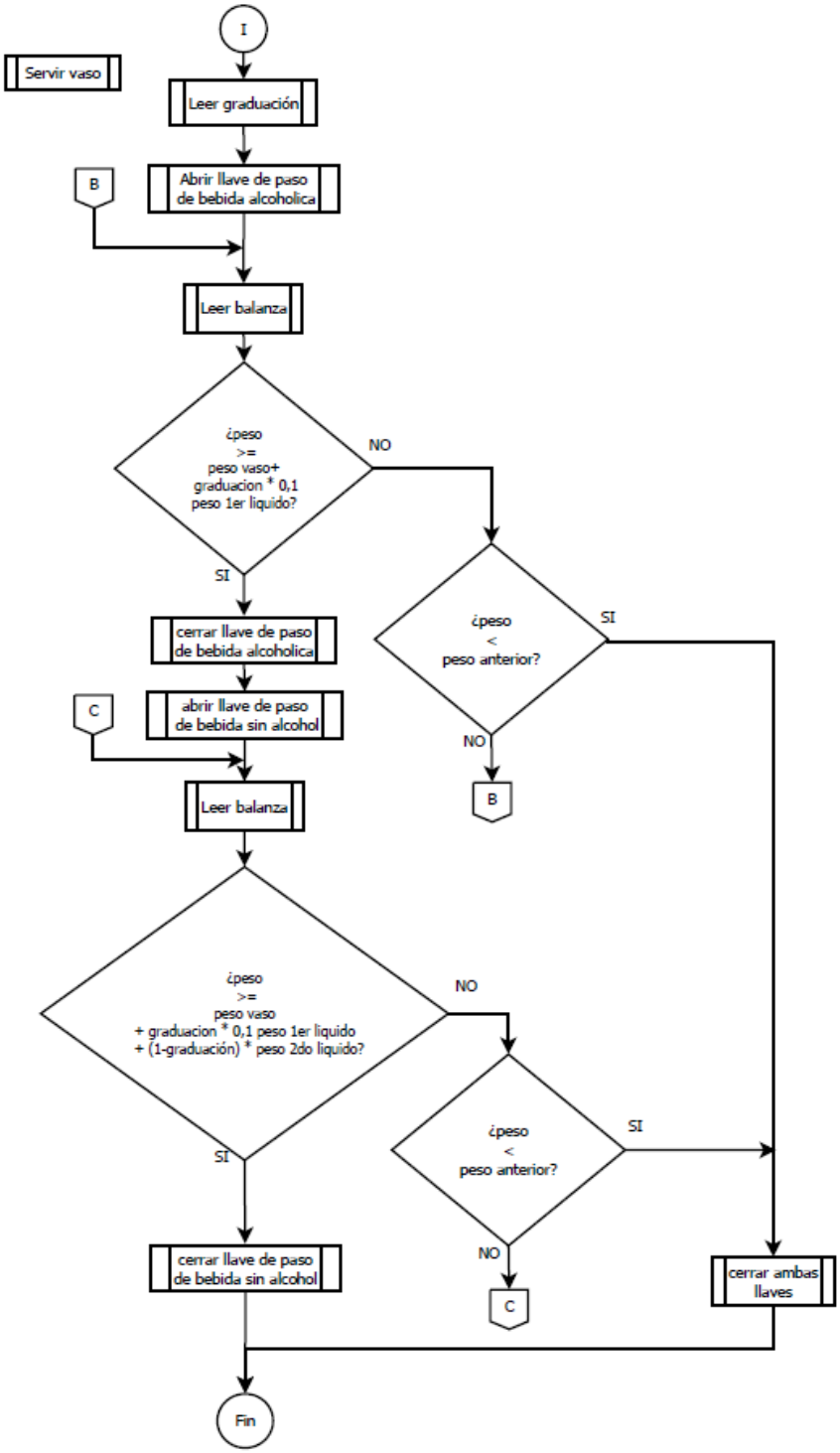


Figura 2: Diagrama de flujo del algoritmo implementado

5. Conclusiones y posibles mejoras

El proyecto pudo realizarse de manera satisfactoria, llevando a cabo casi todos objetivos propuestos sin grandes complicaciones.

El mayor obstáculo con el que se tuvo que lidiar fue el de comprender la forma en la cual la celda de carga y el módulo *hx711* enviaban los datos al MC y cómo manipular dicha información para poder realizar las acciones deseadas. Utilizando un terminal serie en la computadora se pudo conocer los valores de las mediciones de peso que arrojaba la celda de carga en formato binario y se trabajó en esa escala directamente. Si se hubiera querido mostrar el peso leído, se hubiera tenido que tarar y escalar por un factor dicho valor, pero en este proyecto no era necesario y hubiera provocado ineficiencia.

Como se mencionó anteriormente, una posible mejora sería poder servir un volumen variable dependiendo del recipiente utilizando otros sensores.

Además, se podría agregar una opción para que la máquina tenga la posibilidad de servir distintos tipos de tragos dependiendo de la preferencia del usuario.

6. Código

```
1
2  .include "m328pdef.inc"
3
4  ;--------------------- definiciones para datos SRAM ----------------------
5  ;codigo binario de peso para cota menor de deteccion de pulsos/vasos
6  .equ PESO_UMBRAL_H = 0x02
7  .equ PESO_UMBRAL_M = 0x00
8  .equ PESO_UMBRAL_L = 0x00
9
10 ;codigo binario de peso para cota mayor de deteccion de pulsos/vasos
11 .equ PESO_MAX_H = 0x05
12 .equ PESO_MAX_M = 0xFE
13 .equ PESO_MAX_L = 0x00
14
15
16 ;peso 10% fernet (de 300 ml)
17 .equ PESO_LIQUIDO_1_H = 0x00
18 .equ PESO_LIQUIDO_1_M = 0x6E
19 .equ PESO_LIQUIDO_1_L = 0x80
20
21 ;peso 10% coca (de 300 ml)
22 .equ PESO_LIQUIDO_2_H = 0x00
23 .equ PESO_LIQUIDO_2_M = 0x78
24 .equ PESO_LIQUIDO_2_L = 0x80
25 ;-------------------------------------------------------------------------
26
27 ;--------------------- definiciones  manejar_estado ----------------------
28 ;registro de estados
29 .def estado = r16                          ;ESTE REGISTRO NO PUEDE SER USADO
30                                            ;PARA OTRA COSA
31
32 ;bit de estado de deteccion de pulsos O VASO
33 .equ DT_BIT = 0
34 ;bit de estado de configurar el vaso puesto
35 .equ CV_BIT = 1
36 ;bit de estado de servido primer liquido
37 .equ S1_BIT = 2
38 ;bit de estado de servido de segundo liquido
39 .equ S2_BIT = 3
40 ;bit de estado de espera a retirar vaso lleno
41 .equ RV_BIT = 4
42
43 ;------- estado de deteccion ---------
44 .def contador = r17
45
46 .equ TIEMPO_PULSO_MENOR = 1
47 .equ TIEMPO_PULSO_MAYOR = 3
48
49 .equ TIEMPO_VASO  = 10
50
51 ;------- configurar_vaso -------------
52 .def temp_0 = r18
53 .def temp_L = r19
54 .def temp_M = r20
55 .def temp_H = r21
56
57 .def graduacion = r22
58 ;-------------------------------------------------------------------------
59
```

```
60  ;------------------------ definiciones leer_hx711 ------------------------
61  ;registros auxiliares
62  .def A=r20
63  .def NRO_BITS_HX711 = r19
64
65  ;defino los registros de I/O que voy  a usar
66  .equ DDR_ADSK = DDRB
67  .equ DDR_ADDO = DDRB
68  .equ port_ADSK = portB
69  .equ pin_ADDO = pinB
70
71  ;pines de el/los puerto/s a utilizar
72  .equ ADSK = 1
73  .equ ADDO = 0
74
75  ;registros de paso del dato leido
76  .def peso_leido_L = r16
77  .def peso_leido_M = r17
78  .def peso_leido_H = r18
79  ;-----------------------------------------------------------------------
80
81  ;---------------------- definiciones para promedio ---------------------
82  .equ LONG_TABLA = 8 ;debe ser potencia de 2 (y minimo 2)
83  .equ DIV_LONG_TABLA = 3 ;cuantas veces shiftear para dividir por N
84  .def leido_L=r16
85  .def leido_M=r17
86  .def leido_H=r18
87  ;-----------------------------------------------------------------------
88  ;------------------ definiciones para guardar_peso ---------------------
89  .def gd_temp = r16
90  .def reg_posicion_tabla = r17
91  .def gd_contador = r18
92
93  .equ TMNO_DATO = 3
94  .equ FIN_TABLA = 24
95  ;-----------------------------------------------------------------------
96
97  ;----------------- cambiar graduacion (control de display) -------------
98  .equ DDR_DISPLAY = DDRD
99  .equ PORT_DISPLAY = PORTD        ;puerto utilizado para el display
100
101 ;posiciones de los LEDs en el puerto
102 .equ LED_1 = 2
103 .equ LED_2 = 3
104 .equ LED_3 = 4
105 .equ LED_4 = 5
106 .equ LED_5 = 6
107 .def display = r16
108 ;-----------------------------------------------------------------------
109 ;------------------- definiciones obtener graduacion -------------------
110 ;deben coincidir con el display de leds
111 .equ GRAD_20 =3
112 .equ GRAD_30 =4
113 .equ GRAD_40 =5
114 .equ GRAD_50 =6
115 ;-----------------------------------------------------------------------
116 ;------------------- definiciones control de bomba ---------------------
117 .equ DDR_BOMBAS = DDRB
118 .equ PORT_BOMBAS = PORTB
119
120 ;pines donde estan las bombas
121 .equ BOMBA_1 = 3
```

```
122 .equ BOMBA_2 = 4
123 ;***************************************************************************
124
125 ;******************** definiciones para cmp24 ******************************
126 .def cp24_temp = r24
127 ;***************************************************************************
128 ;************* definiciones para comunicacion uart (USART0) ****************
129 .def  usart_leido = r23
130 .def usart_escribir = r24
131 ;***************************************************************************
132
133 .dseg
134 ;umbral de deteccion para estado activo
135 peso_umbral:
136         .byte 3
137 ;maximo peso que puede pesar un vaso
138 peso_max:
139         .byte 3
140 ;valor binario del peso de 10% de vaso de liquido 1
141 peso_liquido_1:
142         .byte 3
143 ;valor binario del peso de 10% de vaso de liquido 2
144 peso_liquido_2:
145         .byte 3
146
147 ;dato leido del modulo
148 dato_hx711:
149         .byte 3
150
151 ;tabla de 8 pesos leidos
152 tabla_pesos:
153         .byte 24
154 ;guardo la posicion de la tabla para escribirla
155 posicion_tabla:
156         .byte 1
157
158 ;promedio de valores de tabla (lo devuelve la funcion promedio)
159 promedio:
160         .byte 3
161
162 ;hasta el valor que este guardado aca va a servir el primer liquido
163 cota_1:
164         .byte 3
165 ;hasta el valor que este aca va a servir el segundo liquido
166 cota_2:
167         .byte 3
168
169 .cseg
170         .org 0x00
171         rjmp config
172
173         .org OVF1addr
174         rjmp ISR_T1_OV
175         .org int_vectors_size
176
177 config:
178
179         ;inicializo stack pointer
180         ldi r20, high(RAMEND)
181         out sph, r20
182         ldi r20, low(RAMEND)
183         out spl, r20
```

```
184
185         ;cargo el valor para el BAUD rate (BAUD = 9600) --> UBRR = 103
186         ldi r17, high(103)
187         ldi r16, low(103)
188
189         ;inicializo la comunicacion usart
190         rcall USART_Init
191         clr r16
192         clr r17
193
194 ;inicializo la posicion de tabla en 0
195         ldi XL, low(posicion_tabla)
196         ldi XH, high(posicion_tabla)
197         ldi r20, 0
198         st X, r20
199
200 ;inicializo el sleep mode
201         in r20, SMCR
202         ;limpio los bits de velocidad en el registro
203         ori r20, 1<<SE
204
205         ;seteo la modo de timer (a idle: (SM2|1|0) = 000)
206         andi r20, ~(1<<SM0|1<<SM1|1<<SM2)
207         out SMCR, R20
208
209 ; guardo los datos de umbral, peso de 10% liquido 1, peso de 10% liquido 2
210 ;en la sram
211         ldi XL, low(peso_umbral)
212         ldi XH, high(peso_umbral)
213
214         ldi r20, PESO_UMBRAL_H
215         st X+, r20
216         ldi r20, PESO_UMBRAL_M
217         st X+, r20
218         ldi r20, PESO_UMBRAL_L
219         st X, r20
220
221         ldi XL, low(peso_max)
222         ldi XH, high(peso_max)
223
224         ldi r20, PESO_MAX_H
225         st X+, r20
226         ldi r20, PESO_MAX_M
227         st X+, r20
228         ldi r20, PESO_MAX_L
229         st X, r20
230
231
232         ldi XL, low(peso_liquido_1)
233         ldi XH, high(peso_liquido_1)
234
235         ldi r20, PESO_LIQUIDO_1_H
236         st X+, r20
237         ldi r20, PESO_LIQUIDO_1_M
238         st X+, r20
239         ldi r20, PESO_LIQUIDO_1_L
240         st X, r20
241
242         ldi XL, low(peso_liquido_2)
243         ldi XH, high(peso_liquido_2)
244
245         ldi r20, PESO_LIQUIDO_2_H
```

```
246         st X+, r20
247         ldi r20, PESO_LIQUIDO_2_M
248         st X+, r20
249         ldi r20, PESO_LIQUIDO_2_L
250         st X, r20
251
252 ;configuro puertos de entrada y salida
253
254         ;inicializo los pines para lectura de hx711
255         sbi DDR_ADSK, ADSK
256         cbi DDR_ADDO, ADDO
257
258         ;seteo como salidas a los pines de los led del display
259         in r20, DDR_DISPLAY
260         ori r20, (1<<LED_1)|(1<<LED_2)|(1<<LED_3)|(1<<LED_4)|(1<<LED_5)
261         out  DDR_display, r20
262         ;siempre debe estar prendido el LED_1 (no hay opcion de graduacion 0)
263         sbi PORT_DISPLAY, LED_1
264
265         ;seteo los pines de control de bombas de liquido como salidas
266         in r20, DDR_BOMBAS
267         ori r20, (1<<BOMBA_1)|(1<<BOMBA_2)
268         out  DDR_BOMBAS, r20
269
270 ;inicializo el timer1
271
272         lds r20, TCCR1B
273         ;limpio los bits de velocidad en el registro
274         andi r20, ~(1<<CS12|1<<CS11|1<<CS10)
275         ;seteo la velocidad
276         ;velocidad (1 muestra cada 4ms)
277         ori r20, 0<<CS12|1<<CS11|1<<CS10
278         sts TCCR1B,R20
279
280         ;habilito la interrupcion
281         lds r20, TIMSK1
282         ori R20, 1<<TOIE1
283         sts TIMSK1, r20
284
285 ;setear registro de estados en 0 (chequeo de pulso/vaso)
286         clr estado
287         ori estado, 1<<DT_BIT
288
289 ;habilito interrupciones globales
290         sei
291
292 main:
293         ;entro en sleep hasta medir un peso
294         sleep
295         rcall manejo_estado
296         rjmp main
297
298 ;-----------------------------------------------------------------------
299 manejo_estado:
300
301         sbrc estado, DT_BIT
302         rcall estado_deteccion
303
304         sbrc estado, CV_BIT
305         rcall estado_configurar_vaso
306
307         sbrc estado, S1_BIT
```

```
308         rcall estado_servir_liquido_1
309
310         sbrc estado, S2_BIT
311         rcall estado_servir_liquido_2
312
313         sbrc estado, RV_BIT
314         rcall estado_retirar_vaso
315
316         ret
317
318 ;-------------------- (estado 1) ----------------------------
319 ;inputs: registro "estado"
320 ;outputs: registro "estado"
321
322 estado_deteccion:
323
324         ldi XL, low(dato_hx711)
325         ldi XH, high(dato_hx711)
326         ldi YL, low(peso_max)
327         ldi YH, high(peso_max)
328
329         ;si el dato leido es mayor al peso maximo para un vaso,
330         ;lo toma como nada y limpia el contador
331         rcall cp24
332         brts nada
333
334
335         ldi YL, low(peso_umbral)
336         ldi YH, high(peso_umbral)
337         rcall cp24
338
339         ;si el dato leido es menor al umbral y menor al mAximo,
340         ;va a validar si es un pulso
341         brtc validar
342
343         ;si es mayor a umbral, incrementa el contador
344         inc contador
345
346         ;si el contador es igual a TIEMPO_VASO, cambia los flags: CV=1 Y DT=0
347         cpi contador, TIEMPO_VASO
348         breq cambiar_a_CV
349
350         ;si el contador no es tiempo vaso, vuelve al inicio a sleep
351         rjmp ret_estado_deteccion
352
353 ;salto aca si el dato leido es menor al umbral
354 validar:
355         ;si el contador esta entre TIEMPO_PULSO_MENOR y TIMEPO_PULSO_MAYOR,
356         ;cambia la graduacion elegida
357         cpi contador, TIEMPO_PULSO_MENOR
358         brsh seguir
359         rjmp nada
360 seguir:
361         cpi contador, TIEMPO_PULSO_MAYOR
362         brlo cambio_de_graduacion
363
364
365         ;si no fue un pulso, resetea el contador
366 nada:
367         clr contador
368         rjmp ret_estado_deteccion
369
```

```
370 cambio_de_graduacion:
371         ;desactivar timer aca? (no creo que sea necesario aca)
372         rcall cambiar_graduacion
373         ;activarlo aca de nuevo?
374 
375         clr contador
376         rjmp ret_estado_deteccion
377 
378 ;salto aca si el contador llego a TIEMPO_VASO
379 cambiar_a_CV:
380         clr contador
381         clr estado
382         ori estado, 1<<CV_BIT
383 
384 ret_estado_deteccion:
385         rcall guardar_peso
386         ret
387 
388 ;-------------- (estado 2) --------------------
389 ;en este  estado configuro las cotas hasta las que va a servir cada liquido
390 ;inputs: registro "estado"
391 ;outputs: registro "estado"
392 
393 estado_configurar_vaso:
394         push graduacion
395         push XL
396         push XH
397         push temp_0
398         push temp_L
399         push temp_M
400         push temp_H
401 
402 ;desactivo el timer cuando configuro
403         rcall desactivar_timer
404 
405         ldi XL, low(dato_hx711)
406         ldi XH, high(dato_hx711)
407 
408         ;devuelve la graduacion en el registro "graduacion"
409         rcall obtener_graduacion
410 
411         ;guardo en temp el peso del vaso
412         ld temp_H, X+
413         ld temp_M, X+
414         ld temp_L, X
415 
416 ;guardo en cota 1 el valor del vaso + el peso del 10% del liquido_1
417 ;por la graduacion elegida
418         ldi XL, low(peso_liquido_1+2)
419         ldi XH, high(peso_liquido_1+2)
420         ;meto el valor en el stack para no perderlo
421         push graduacion
422 loop_cota_1:
423         ld temp_0, X
424         add temp_L, temp_0
425         ld temp_0, -X
426         adc temp_M, temp_0
427         ld temp_0, -X
428         adc temp_H, temp_0
429 
430         ;vuelvo al puntero a la posicion inicial
431         adiw XH:XL, 2
```

```
432
433         dec graduacion
434         brne loop_cota_1
435         ;obtengo de nuevo el valor de graduacion
436         pop graduacion
437
438         ldi XL, low(cota_1)
439         ldi XH, high(cota_1)
440         st X+, temp_H
441         st X+, temp_M
442         st X, temp_L
443
444 ;pone como segunda cota peso_vaso + peso_10_liquido_1 * graduacion +
445 ;peso_liquido_2 * (10-graduacion)
446         ldi XL, low(peso_liquido_2+2)
447         ldi XH, high(peso_liquido_2+2)
448         ;hago graduacion -10 y lo complemeno (ca2) (para que de positivo)
449         subi graduacion, 10
450         neg graduacion
451
452 loop_cota_2:
453         ld temp_0, X
454         add temp_L, temp_0
455         ;sbiw XH:XL, 1
456         ld temp_0, -X
457         adc temp_M, temp_0
458         ;sbiw XH:XL, 1
459         ld temp_0, -X
460         adc temp_H, temp_0
461
462         ;vuelvo al puntero a la posicion inicial
463         adiw XH:XL, 2
464
465         dec graduacion
466         brne loop_cota_2
467
468         ldi XL, low(cota_2)
469         ldi XH, high(cota_2)
470         st X+, temp_H
471         st X+, temp_M
472         st X, temp_L
473
474 ;cambio estado a servir liquido_1 y vuelvo
475         clr estado
476         ori estado, 1<<S1_BIT
477         ;activo el timer de nuevo
478         rcall activar_timer
479
480         pop temp_H
481         pop temp_M
482         pop temp_L
483         pop temp_0
484         pop XH
485         pop XL
486         pop graduacion
487         ret
488
489 ;-------------- (estado 3) -------------------------
490 ;inputs: registro "estado"
491 ;outputs: registro "estado"
492
493 estado_servir_liquido_1:
```

```
494         rcall abrir_bomba_1
495         rcall promedio_tabla
496
497         ldi XL, low(dato_hx711)
498         ldi XH, high(dato_hx711)
499
500         ldi YL, low(peso_umbral)
501         ldi YH, high(peso_umbral)
502
503         rcall cp24
504         ;si el peso leido es menor al umbral, cancelo el servido
505         brtc cancelar_s1
506
507         ldi YL, low(promedio)
508         ldi YH, high(promedio)
509
510         rcall cp24
511         ;si el peso leido es menor al promedio, cancelo el servido
512         brtc cancelar_s1
513
514         ldi YL, low(cota_1)
515         ldi YH, high(cota_1)
516         rcall cp24
517         ;si el peso leido es mayor a la cota_1, salta
518         brts terminar_s1
519         rjmp ret_servir_liquido_1
520
521 cancelar_s1:
522         rcall cerrar_bomba_1
523         ;cambio estado a RV
524         ;(que espera a que el peso leido sea menor al umbral para reiniciar)
525         clr estado
526         ori estado, 1<<RV_BIT
527         rjmp ret_servir_liquido_1
528
529 terminar_s1:
530         rcall cerrar_bomba_1
531         clr estado
532         ori estado, 1<<S2_BIT
533
534 ret_servir_liquido_1:
535         rcall guardar_peso
536         ret
537
538
539 ;-------------- (estado 4) -------------------------
540 ;inputs: registro "estado"
541 ;outputs: registro "estado"
542
543 estado_servir_liquido_2:
544
545         rcall abrir_bomba_2
546         rcall promedio_tabla
547
548         ldi XL, low(dato_hx711)
549         ldi XH, high(dato_hx711)
550
551         ldi YL, low(peso_umbral)
552         ldi YH, high(peso_umbral)
553
554         rcall cp24
555         ;si el peso leido es menor al umbral, cancelo el servido
```

```
556         brtc cancelar_s2
557
558         ldi YL, low(promedio)
559         ldi YH, high(promedio)
560
561         rcall cp24
562         ;si el peso leido es menor al promedio, salta
563         brtc cancelar_s2
564
565         ldi YL, low(cota_2)
566         ldi YH, high(cota_2)
567         rcall cp24
568         ;si el peso leido es mayor a la cota_2, salta
569         brts terminar_s2
570         rjmp ret_servir_liquido_2
571
572 cancelar_s2:
573         rcall cerrar_bomba_2
574         ;cambio estado a RV
575         ;(que espera a que el peso leido sea menor al umbral para reiniciar)
576         clr estado
577         ori estado, 1<<RV_BIT
578         rjmp ret_servir_liquido_2
579
580 terminar_s2:
581         rcall cerrar_bomba_2
582         clr estado
583         ori estado, 1<<RV_BIT
584
585 ret_servir_liquido_2:
586         rcall guardar_peso
587         ret
588
589
590
591
592 ;--------------- (estado 5) -------------------------
593 ;inputs: registro "estado"
594 ;outputs: registro "estado"
595
596 estado_retirar_vaso:
597         rcall promedio_tabla
598
599         ldi XL, low(promedio)
600         ldi XH, high(promedio)
601
602         ldi YL, low(peso_umbral)
603         ldi YH, high(peso_umbral)
604
605         rcall cp24
606         ;si el promedio es menor al umbral, cambia de estado a Deteccion
607         brtc cambiar_a_DT
608         rjmp ret_estado_retirar_vaso
609
610 cambiar_a_DT:
611         clr estado
612         ori estado, 1<<DT_BIT
613
614 ret_estado_retirar_vaso:
615         rcall guardar_peso
616         ret
617
```

```
618
619 ;-----------------------------------------------------------------
620 ;guarda el peso leido en una tabla de 8 valores de peso
621 ;inputs: -
622 ;outputs: "dato_Hx711" en sram (3 bytes)
623
624 (cada uno de 3 bytes)
625 guardar_peso:
626         push gd_contador
627         push reg_posicion_tabla
628         push gd_temp
629         push XL
630         push XH
631         push YL
632         push YH
633
634         ;pongo en gd_contador el tamano de los datos de la tabla
635         ;(cuantos registros ocupa)
636         ldi gd_contador, TMNO_DATO
637
638         ;obtengo la posicion de donde guarde el dato menos reciente
639         ldi XL, low(posicion_tabla)
640         ldi XH, high(posicion_tabla)
641         ld reg_posicion_tabla, X
642
643         ldi XL, low(dato_hx711)
644         ldi XH, high(dato_hx711)
645         ldi YL, low(tabla_pesos)
646         ldi YH, high(tabla_pesos)
647
648 ;guardo el dato leido en la posicion de la tabla que seguia de la vez anterior
649         add YL, reg_posicion_tabla
650         brcc guardar_siguiente
651         inc YH
652
653 ;guarda la cantidad de registros que ocupa un dato
654 guardar_siguiente:
655         ld gd_temp, X+
656         st Y+, gd_temp
657         dec gd_contador
658         brne guardar_siguiente
659
660         ;incremento la posicion en un dato
661         subi reg_posicion_tabla, -TMNO_DATO
662
663         ;si llega al final de la tabla lo vuelve a 0
664         cpi reg_posicion_tabla, FIN_TABLA
665         brne ret_guardar_dato
666         ldi reg_posicion_tabla, 0
667
668 ret_guardar_dato:
669         ldi XL, low(posicion_tabla)
670         ldi XH, high(posicion_tabla)
671
672         st X, reg_posicion_tabla
673
674         pop YH
675         pop YL
676         pop XH
677         pop XL
678         pop gd_temp
679         pop reg_posicion_tabla
```

```
680         pop gd_contador
681
682         ret
683
684 ;------------------------ cambiar graduacion --------------------------------
685 ;cambia los LED del display para indicar la graduacion elegida
686 ;inputs: registro PORT_DISPLAY
687 ;outputs: registro PORT_DISPLAY
688
689 cambiar_graduacion:
690         push display
691
692         ;leemos en el registro display el puerto de los leds
693
694         ;comparamos uno a uno si estan encendidos los LEDs
695         ; cuando lee uno no prendido, lo prende y sale
696         sbis PORT_DISPLAY, LED_2
697         rjmp d_graduacion_20
698         sbis PORT_DISPLAY, LED_3
699         rjmp d_graduacion_30
700         sbis PORT_DISPLAY, LED_4
701         rjmp d_graduacion_40
702         sbis PORT_DISPLAY, LED_5
703         rjmp d_graduacion_50
704
705         ;si todos los LEDs estaban prendidos, reinicia el display
706         ;dejando prendido el LED1 (y deja los pines que no son led como estaban)
707         in display, PORT_DISPLAY
708         andi display, ~((1<<LED_2)|(1<<LED_3)|(1<<LED_4)|(1<<LED_5))
709         out PORT_DISPLAY, display
710         rjmp ret_modificar_display
711
712 d_graduacion_20:
713         sbi PORT_DISPLAY, LED_2
714         rjmp ret_modificar_display
715 d_graduacion_30:
716         sbi PORT_DISPLAY, LED_3
717         rjmp ret_modificar_display
718 d_graduacion_40:
719         sbi PORT_DISPLAY, LED_4
720         rjmp ret_modificar_display
721 d_graduacion_50:
722         sbi PORT_DISPLAY, LED_5
723         rjmp ret_modificar_display
724
725 ret_modificar_display:
726         pop display
727         ret
728 ;--------------------- obtener graduacion ---------------------
729 ;obtiene la graduacion elegida a partir del display y la guarda
730 ;en el registro "graduacion"
731 ;inputs: registro PORT_DISPLAY
732 ;outputs: registro "graduacion"
733
734 obtener_graduacion:
735
736         ;la graduacion debe estar al menos en 10% (graduacion = 1)
737         ldi graduacion, 1
738
739         ;chequeo si los leds estan prendidos, y si lo estan, aumenta la graduacion
740         ;en 1, cuando ve el primero sin prender, sale
741         sbis PORT_DISPLAY, GRAD_20
```

```
742         rjmp ret_obtener_graduacion
743         inc graduacion
744
745         sbis PORT_DISPLAY, GRAD_30
746         rjmp ret_obtener_graduacion
747         inc graduacion
748
749         sbis PORT_DISPLAY, GRAD_40
750         rjmp ret_obtener_graduacion
751         inc graduacion
752
753         sbis PORT_DISPLAY, GRAD_50
754         rjmp ret_obtener_graduacion
755         inc graduacion
756
757 ret_obtener_graduacion:
758         ret
759 ;------------------------ rutinas de control de timer ------------------------------
760 ;inputs: -
761 ;outputs: -
762
763 activar_timer:
764         push r20
765         lds r20, TIMSK1
766         ori R20, 1<<TOIE1
767         sts TIMSK1, r20
768         pop r20
769         ret
770
771 desactivar_timer:
772         push r20
773         lds r20, TIMSK1
774         andi r20, ~(1<<TOIE1)
775         sts TIMSK1, r20
776         pop r20
777         ret
778 ;------------------------ rutinas de control de bombas ------------------------------
779 ;inputs: registro PORT_BOMBAS
780 ;outputs: registro PORT_BOMBAS
781
782 abrir_bomba_1:
783         sbi PORT_BOMBAS, BOMBA_1
784         ret
785 cerrar_bomba_1:
786         cbi PORT_BOMBAS, BOMBA_1
787         ret
788 abrir_bomba_2:
789         sbi PORT_BOMBAS, BOMBA_2
790         ret
791 cerrar_bomba_2:
792         cbi PORT_BOMBAS, BOMBA_2
793         ret
794 ;-------------------------- leer_hx711 ----------------------------------------------
795 ;lee el peso obtenido por el modulo amplificador conversor
796 ;y lo guarda en la ram en la posicion "dato_hx711"
797 leer_hx711:
798         push NRO_BITS_HX711
799         push A
800         push peso_leido_L
801         push peso_leido_M
802         push peso_leido_H
803         push XL
```

```
804         push XH
805         ;limpio el carry porque lo voy a usar
806         clc
807         ;limpio los registros que voy a usar
808         clr peso_leido_L
809         clr peso_leido_M
810         clr peso_leido_H
811
812         ;habilito la conversion de datos si no estaba activada
813         cbi port_ADSK, ADSK
814
815         ;si no termino la conversion vuelve a chequear ADDO
816 AD_not_finished:
817         sbic pin_ADDO, ADDO
818         rjmp AD_not_finished
819
820         ;cargo el contador r19 con 24 para pasar 24 bits
821         ldi NRO_BITS_HX711, 24
822
823 ShiftOut:
824         ;mando un pulso de clock
825         sbi port_ADSK, ADSK
826         ;se necesita delay de 1us aproximadamente,
827         ;usamos 18 ciclos de maquina (con freq=16MHz), por lo que tarda 1,125us
828         rcall T_high
829         cbi port_ADSK, ADSK
830
831         ;guarda el dato leido en el carry
832         sbic pin_ADDO, ADDO
833         sec
834
835         ;guarda el bit leido en los registros
836         mov A, peso_leido_L
837         rol A
838         mov peso_leido_L ,A
839         mov A, peso_leido_M
840         rol A
841         mov peso_leido_M ,A
842         mov A, peso_leido_H
843         rol A
844         mov peso_leido_H ,A
845         ;chequeo si movio los 24 bits
846         dec r19
847         brne ShiftOut
848
849         ;vuelvo a poner el clock en 1 cuando termina y asi
850         ;pone a DOUT en alto nuevamente
851         sbi port_ADSK, ADSK
852         rcall T_high
853
854         ;el clock debe terminar en bajo
855         ;para no entrar en modo de bajo consumo del hx711
856         cbi port_ADSK, ADSK
857
858         ldi XL, low(dato_hx711)
859         ldi XH, high(dato_hx711)
860
861 ;guardo el dato en la SRAM
862         st X+, peso_leido_H
863         st X+, peso_leido_M
864         st X, peso_leido_L
865
```

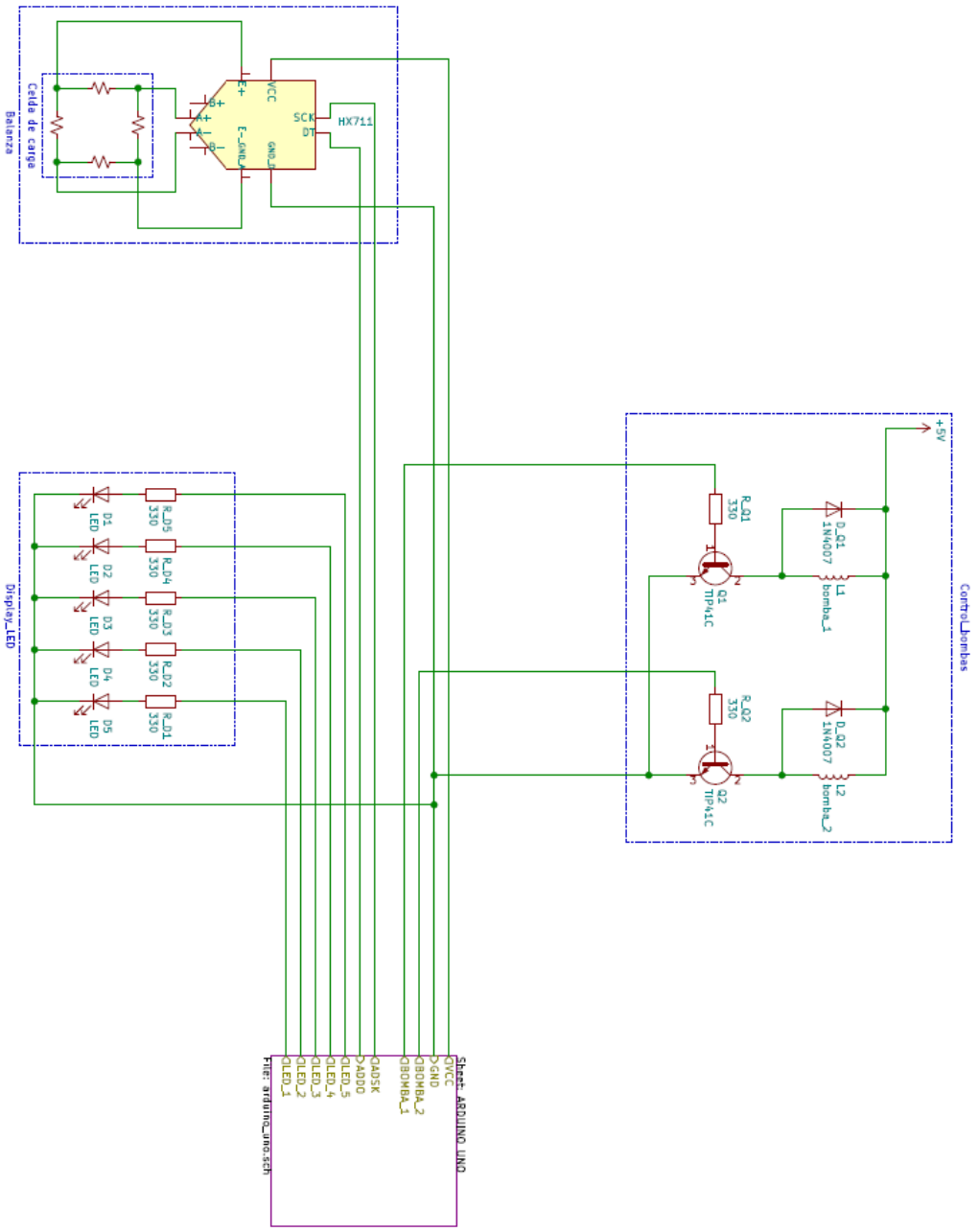
```
866         pop XH
867         pop XL
868         pop peso_leido_H
869         pop peso_leido_M
870         pop peso_leido_L
871         pop A
872         pop NRO_BITS_HX711
873         ret
874
875 ;este delay dura 15 ciclos de maquina, sin contar el rcall
876 T_high:
877         push r16
878         ldi r16, 3
879 T_h_loop:
880         dec r16
881         brne T_h_loop
882
883         pop r16
884         ret
885 ;------------------------------ promedio de tabla ----------------------------------------
886 ;hace el promedio de los datos almacenados en la tabla
887 ;devuelve el resultado en la posicion "promedio" en sram
888 ;inputs: "tabla_pesos" en sram (24 bytes)
889 ;outputs: "promedio" en sram (3 bytes)
890
891 promedio_tabla:
892         push leido_L
893         push leido_M
894         push leido_H
895         push r19
896         push r20
897         push r21
898         push r22
899         push r23
900         push XL
901         push XH
902
903 ;limpio los registros acumuladores
904         clr r20
905         clr r21
906         clr r22
907         clr r23
908 ;puntero para obtener los datos que leo
909         ldi XL, LOW(tabla_pesos)
910         ldi XH, HIGH(tabla_pesos)
911
912         ;hace el promedio de LONG_TABLA pesos leidos
913         ldi r19, LONG_TABLA
914 sumar:
915
916         ld leido_H, X+
917         ld leido_M, X+
918         ld leido_L, X+
919
920         ;sumo de a 4 registros y acumulo el resultado en r23:r22:r21:r20
921         add r20, leido_L
922         adc r21, leido_M
923         adc     r22, leido_H
924         ;si hubo carry incrementa el registro mas significativo
925         brcc skip
926         inc r23
927 skip:
```

```
928         dec r19
929         brne sumar
930
931         ;divide por LONG_TABLA (tiene que ser pot de 2) el resultado
932         ldi r19, DIV_LONG_TABLA
933 dividir:
934         lsr r23
935         ror r22
936         ror r21
937         ror r20
938         dec r19
939         brne dividir
940
941         ;resto un valor pequeno para permitir un margen de tolerancia al servir
942         subi r20, 0x00
943         sbci r21, 0x19
944         sbci r22, 0x00
945         ;una vez obtenido el resultado del promedio de 4 medidas de peso
946         ;lo guarda en la sram (resultado en primeros 24 bits)
947         ldi XL, low(promedio)
948         ldi XH, high(promedio)
949         st X+, r22 ;peso_H
950         st X+, r21 ;peso_M
951         st X, r20  ;peso_L
952
953         pop XH
954         pop XL
955         pop r23
956         pop r22
957         pop r21
958         pop r20
959         pop r19
960         pop leido_H
961         pop leido_M
962         pop leido_L
963         ret
964
965 ;-----------------------------------------------------------------------
966 ;compara a los numeros de 3 bytes ubicados en los punteros X e Y
967 ;devuelve el resultado en el bit T del SREG
968 ;X<Y T = 0, X>=Y T = 1
969 ;inputs: punteros X e Y
970 ;outputs: bit T de SREG
971
972 cp24:
973         push r17
974         push r18
975         push r24
976         push XL
977         push XH
978         push YL
979         push YH
980         push cp24_temp
981         ;contador para comparar 3 registros
982         ldi cp24_temp, 3
983 loop_cpi24:
984         ld r17, X+
985         ld r18, Y+
986         cp r17, r18
987         brlo X_menor_a_Y
988         cp r18, r17
989         brlo X_mayor_o_igual_a_Y
```

```
990             dec cp24_temp
991             brne loop_cpi24
992
993     X_mayor_o_igual_a_Y:
994             set
995             rjmp ret_cp24
996
997     X_menor_a_Y:
998             clt
999             rjmp ret_cp24
1000
1001    ret_cp24:
1002
1003            pop cp24_temp
1004            pop YH
1005            pop YL
1006            pop XH
1007            pop XL
1008            pop r24
1009            pop r18
1010            pop r17
1011            ret
1012    ;-------------------------------------------------------------------------
1013
1014
1015
1016
1017    ;---------- interrupcion por timer overflow ----------------------
1018    ;lee el peso medido en el instante de la interrupcion
1019    ;y lo manda por comunicaci[U+FFFD]n UART al terminal serie
1020    ;inputs: -
1021    ;outputs: -
1022
1023    ISR_T1_OV:
1024            push XL
1025            push XH
1026            push usart_escribir
1027
1028            rcall leer_hx711
1029
1030    ;transmito el resultado por el puerto serie para leerlo en la pantalla
1031            ldi XL, low(dato_hx711)
1032            ldi XH, high(dato_hx711)
1033
1034            ld usart_escribir, X+
1035            rcall USART_Transmit
1036
1037            ld usart_escribir, X+
1038            rcall USART_Transmit
1039
1040            ld usart_escribir, X
1041            rcall USART_Transmit
1042
1043            pop usart_escribir
1044            pop XH
1045            pop XL
1046            reti
1047
1048    ;-----------------------------USART0-------------------------------------
1049    ;inputs: r16 y r17 (con BAUD rate)
1050    ;outputs: -
1051
```

```
1052 ;inicializar la comunicacion USART asincronica normal
1053 USART_Init:
1054         ; Setea baud rate (asume que el UBRR esta en R17(H):R16(L))
1055         sts UBRR0H, r17
1056         sts UBRR0L, r16
1057         ; habilita transmision y recepcion
1058         ldi r16, (1<<RXEN0)|(1<<TXEN0)
1059         sts UCSR0B,r16
1060         ; Setea formato de "frame"(bits de la comunicacion): 8data, 2stop bit
1061         ldi r16, (1<<USBS0)|(3<<UCSZ00)
1062         sts UCSR0C,r16
1063         ret
1064 ;-----------------------------------------------------------
1065 ;recibir datos de 5 a 8 bits en Usart_leido
1066 ;inputs: -
1067 ;outputs: registro "usart_leido"
1068
1069 USART_Receive:
1070         push r21
1071         ; Espera a recibir dato
1072 loop_r:
1073         lds R21, UCSR0A
1074         sbrs R21, RXC0
1075         rjmp loop_r
1076         ; recibe los datos del buffer UDR0
1077         lds usart_leido, UDR0
1078
1079         pop r21
1080         ret
1081 ;-----------------------------------------------------------
1082 ;transmitir 5 a 8 bits por usart_escribir
1083 ;inputs: -
1084 ;outputs: usart_escribir
1085
1086 USART_Transmit:
1087         push r20
1088         ; Espera a que el buffer de transmision este vacio
1089 loop_t:
1090         lds R20,UCSR0A
1091         sbrs R20,UDRE0
1092         rjmp loop_t
1093         ; pone el dato de R16 en el buffer de transmision y lo envia
1094         sts UDR0, usart_escribir
1095         pop r20
1096         ret
1097 ;-----------------------------------------------------------
```

7. Apéndice


Celda de carga
Balanza
HX711
VCC
SCK
DT
GND
Display_LED
LED
330
Control_bombas
+5V
TIP41C
1N4007
bomba_1
bomba_2
Sheet: ARDUINO_UNO
File: arduino_uno.sch

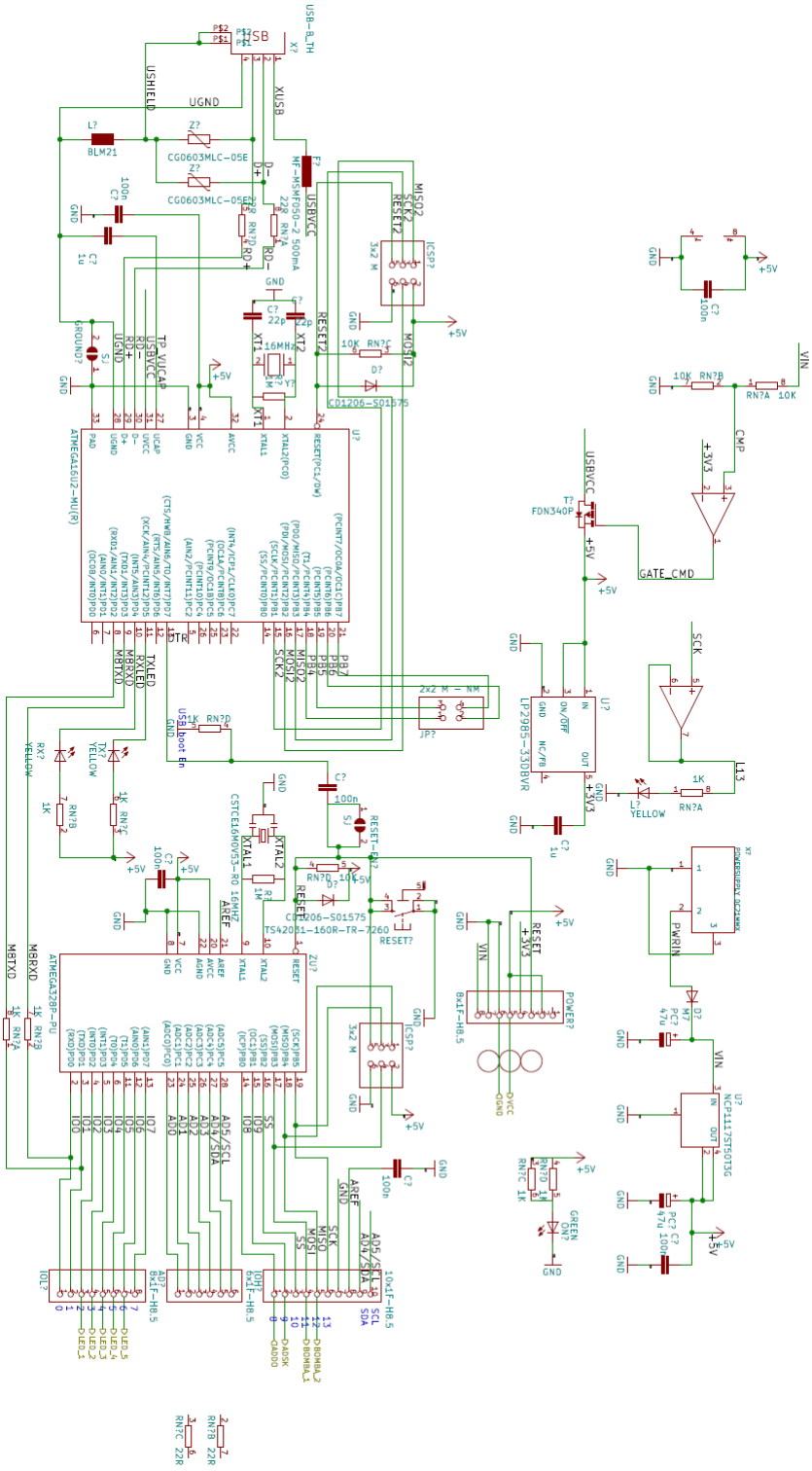


Figura 3: Esquemático del circuito implementado

Referencias

[1] M.A. Mazidi, S. Naimi y S. Naimi, *The AVR microcontroller and embedded system: Using Assembly and C.*

[2] Hoja de datos del microcontrolador **AT-Mega328p**: *http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf*

[3] Hoja de datos del módulo **HX711**: *https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf*

[4] Hoja de datos del transistor **TIP41C**: *https://www.st.com/resource/en/datasheet/tip41c.pdf*

[5] Hoja de datos del diodo **1N4007**: *https://www.diodes.com/assets/Datasheets/ds28002.pdf*