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

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.

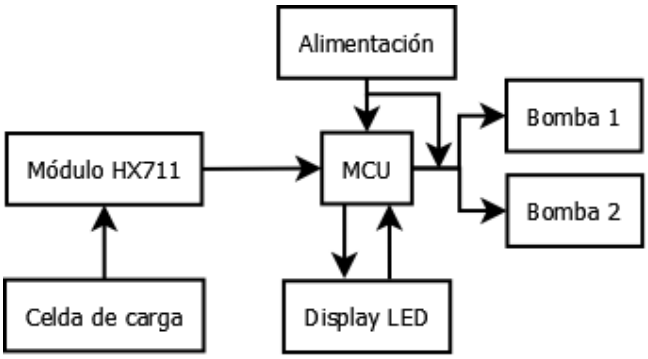


Figura 1: Diagrama en bloques de hardware

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 con una velocidad de $\frac{f_{cl}}{64}$. El microcontrolador se encuentra en modo sleep y sólo se enciende a partir de las interrupciones provocadas por el timer. 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.

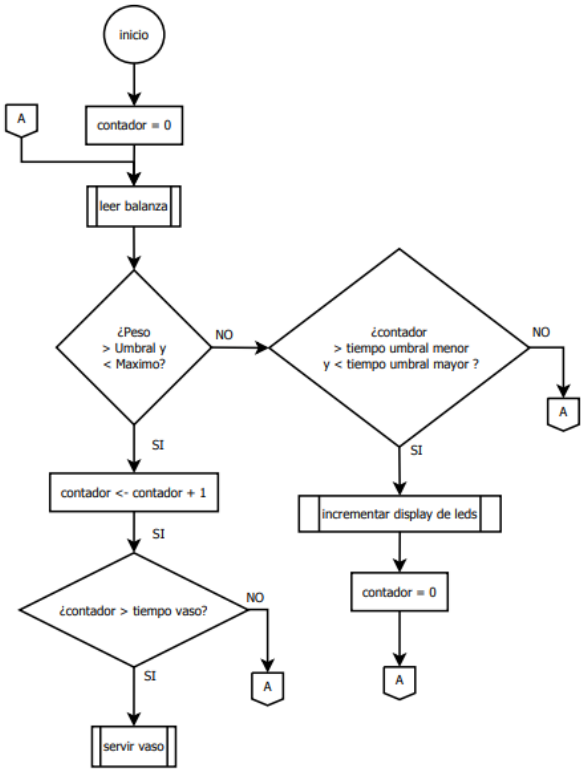
Luego de que se haya fijado el volumen a verter del primer fluido, el usuario deberá apoyar el recipiente sobre la balanza y cuando el microcontrolador detecte que se apoyó por más de la cantidad de muestras para un vaso, este activará la bomba que expenderá dicho líquido, hasta que la balanza mida un 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.



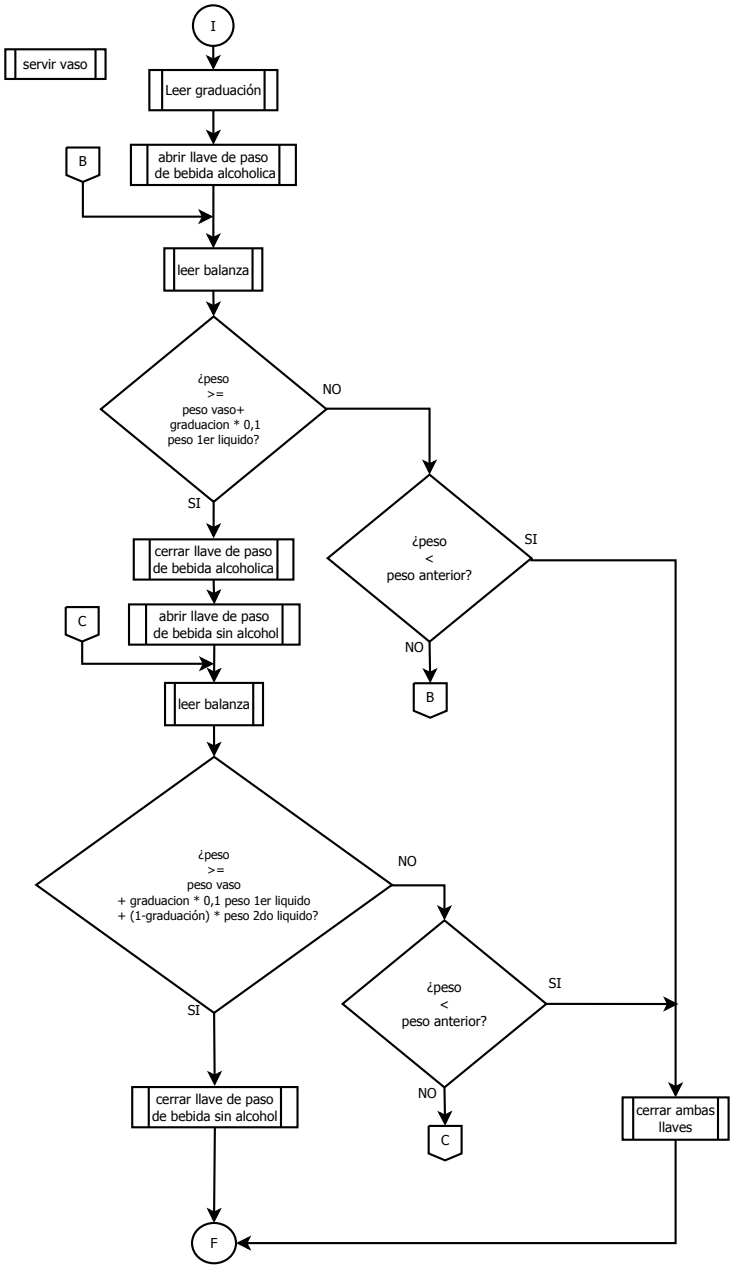


Figura 2: Diagrama de flujo del algoritmo implementado

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 main
172
173         .org OVF1addr
174         rjmp ISR_T1_OV
175         .org int_vectors_size
176
177 main:
178
179 ;configuracion
180         ;inicializo stack pointer
181         ldi r20, high(RAMEND)
182         out sph, r20
183         ldi r20, low(RAMEND)
```

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

```
246         ldi r20, PESO_LIQUIDO_2_H
247         st X+, r20
248         ldi r20, PESO_LIQUIDO_2_M
249         st X+, r20
250         ldi r20, PESO_LIQUIDO_2_L
251         st X, r20
252
253 ;configuro puertos de entrada y salida
254
255         ;inicializo los pines para lectura de hx711
256         sbi DDR_ADSK, ADSK
257         cbi DDR_ADDO, ADDO
258
259         ;seteo como salidas a los pines de los led del display
260         in r20, DDR_DISPLAY
261         ori r20, (1<<LED_1)|(1<<LED_2)|(1<<LED_3)|(1<<LED_4)|(1<<LED_5)
262         out  DDR_display, r20
263         ;siempre debe estar prendido el LED_1 (no hay opcion de graduacion 0)
264         sbi PORT_DISPLAY, LED_1
265
266         ;seteo los pines de control de bombas de liquido como salidas
267         in r20, DDR_BOMBAS
268         ori r20, (1<<BOMBA_1)|(1<<BOMBA_2)
269         out  DDR_BOMBAS, r20
270
271 ;inicializo el timer1
272
273         lds r20, TCCR1B
274         ;limpio los bits de velocidad en el registro
275         andi r20, ~(1<<CS12|1<<CS11|1<<CS10)
276         ;seteo la velocidad
277         ;velocidad (1 muestra cada 4ms)(sin prescaler)
278         ori r20, 0<<CS12|1<<CS11|1<<CS10
279         sts TCCR1B,R20
280
281         ;habilito la interrupcion
282         lds r20, TIMSK1
283         ori R20, 1<<TOIE1
284         sts TIMSK1, r20
285
286 ;setear registro de estados en 0 (chequeo de pulso/vaso)
287         clr estado
288         ori estado, 1<<DT_BIT
289
290 ;habilito interrupciones globales
291         sei
292
293 main_loop:
294         ;entro en sleep hasta medir un peso
295         sleep
296         rcall manejo_estado
297         rjmp main_loop
298
299 ;-------------------------------------------------------------------------
300 manejo_estado:
301
302         sbrc estado, DT_BIT
303         rcall estado_deteccion
304
305         sbrc estado, CV_BIT
306         rcall estado_configurar_vaso
307
```

```
308         sbrc estado, S1_BIT
309         rcall estado_servir_liquido_1
310
311         sbrc estado, S2_BIT
312         rcall estado_servir_liquido_2
313
314         sbrc estado, RV_BIT
315         rcall estado_retirar_vaso
316
317         ret
318
319 ;-------------------- (estado 1) ----------------------------
320 estado_deteccion:
321
322         ldi XL, low(dato_hx711)
323         ldi XH, high(dato_hx711)
324         ldi YL, low(peso_max)
325         ldi YH, high(peso_max)
326
327         ;si el dato leido es mayor al peso maximo para un vaso,
328         ;lo toma como nada y limpia el contador
329         rcall cp24
330         brts nada
331
332
333         ldi YL, low(peso_umbral)
334         ldi YH, high(peso_umbral)
335         rcall cp24
336
337         ;si el dato leido es menor al umbral y menor al mAximo,
338         ;va a validar si es un pulso
339         brtc validar
340
341         ;si es mayor a umbral, incrementa el contador
342         inc contador
343
344         ;si el contador es igual a TIEMPO_VASO, cambia los flags: CV=1 Y DT=0
345         cpi contador, TIEMPO_VASO
346         breq cambiar_a_CV
347
348         ;si el contador no es tiempo vaso, vuelve al inicio a sleep
349         rjmp ret_estado_deteccion
350
351 ;salto aca si el dato leido es menor al umbral
352 validar:
353         ;si el contador esta entre TIEMPO_PULSO_MENOR y TIMEPO_PULSO_MAYOR,
354         ;cambia la graduacion elegida
355         cpi contador, TIEMPO_PULSO_MENOR
356         brsh seguir
357         rjmp nada
358 seguir:
359         cpi contador, TIEMPO_PULSO_MAYOR
360         brlo cambio_de_graduacion
361
362
363         ;si no fue un pulso, resetea el contador
364 nada:
365         clr contador
366         rjmp ret_estado_deteccion
367
368 cambio_de_graduacion:
369         ;desactivar timer aca? (no creo que sea necesario aca)
```

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

```
432
433         ldi XL, low(cota_1)
434         ldi XH, high(cota_1)
435         st X+, temp_H
436         st X+, temp_M
437         st X, temp_L
438
439 ;pone como segunda cota peso_vaso + peso_10_liquido_1 * graduacion +
440 ;peso_liquido_2 * (10-graduacion)
441         ldi XL, low(peso_liquido_2+2)
442         ldi XH, high(peso_liquido_2+2)
443         ;hago graduacion -10 y lo complemeno (ca2) (para que de positivo)
444         subi graduacion, 10
445         neg graduacion
446
447 loop_cota_2:
448         ld temp_0, X
449         add temp_L, temp_0
450         ;sbiw XH:XL, 1
451         ld temp_0, -X
452         adc temp_M, temp_0
453         ;sbiw XH:XL, 1
454         ld temp_0, -X
455         adc temp_H, temp_0
456
457         ;vuelvo al puntero a la posicion inicial
458         adiw XH:XL, 2
459
460         dec graduacion
461         brne loop_cota_2
462
463         ldi XL, low(cota_2)
464         ldi XH, high(cota_2)
465         st X+, temp_H
466         st X+, temp_M
467         st X, temp_L
468
469 ;cambio estado a servir liquido_1 y vuelvo
470         clr estado
471         ori estado, 1<<S1_BIT
472         ;activo el timer de nuevo
473         rcall activar_timer
474
475         pop temp_H
476         pop temp_M
477         pop temp_L
478         pop temp_0
479         pop XH
480         pop XL
481         pop graduacion
482         ret
483
484 ;-------------- (estado 3) -------------------------
485 estado_servir_liquido_1:
486         rcall abrir_bomba_1
487         rcall promedio_tabla
488
489         ldi XL, low(dato_hx711)
490         ldi XH, high(dato_hx711)
491
492         ldi YL, low(peso_umbral)
493         ldi YH, high(peso_umbral)
```

```
494
495         rcall cp24
496         ;si el peso leido es menor al umbral, cancelo el servido
497         brtc cancelar_s1
498
499         ldi YL, low(promedio)
500         ldi YH, high(promedio)
501
502         rcall cp24
503         ;si el peso leido es menor al promedio, cancelo el servido
504         brtc cancelar_s1
505
506         ldi YL, low(cota_1)
507         ldi YH, high(cota_1)
508         rcall cp24
509         ;si el peso leido es mayor a la cota_1, salta
510         brts terminar_s1
511         rjmp ret_servir_liquido_1
512
513 cancelar_s1:
514         rcall cerrar_bomba_1
515         ;cambio estado a RV
516         ;(que espera a que el peso leido sea menor al umbral para reiniciar)
517         clr estado
518         ori estado, 1<<RV_BIT
519         rjmp ret_servir_liquido_1
520
521 terminar_s1:
522         rcall cerrar_bomba_1
523         clr estado
524         ori estado, 1<<S2_BIT
525
526 ret_servir_liquido_1:
527         rcall guardar_peso
528         ret
529
530
531 ;-------------- (estado 4) -------------------------
532 estado_servir_liquido_2:
533
534         rcall abrir_bomba_2
535         rcall promedio_tabla
536
537         ldi XL, low(dato_hx711)
538         ldi XH, high(dato_hx711)
539
540         ldi YL, low(peso_umbral)
541         ldi YH, high(peso_umbral)
542
543         rcall cp24
544         ;si el peso leido es menor al umbral, cancelo el servido
545         brtc cancelar_s2
546
547         ldi YL, low(promedio)
548         ldi YH, high(promedio)
549
550         rcall cp24
551         ;si el peso leido es menor al promedio, salta
552         brtc cancelar_s2
553
554         ldi YL, low(cota_2)
555         ldi YH, high(cota_2)
```

```
556         rcall cp24
557         ;si el peso leido es mayor a la cota_2, salta
558         brts terminar_s2
559         rjmp ret_servir_liquido_2
560
561 cancelar_s2:
562         rcall cerrar_bomba_2
563         ;cambio estado a RV
564         ;(que espera a que el peso leido sea menor al umbral para reiniciar)
565         clr estado
566         ori estado, 1<<RV_BIT
567         rjmp ret_servir_liquido_2
568
569 terminar_s2:
570         rcall cerrar_bomba_2
571         clr estado
572         ori estado, 1<<RV_BIT
573
574 ret_servir_liquido_2:
575         rcall guardar_peso
576         ret
577
578
579
580
581 ;-------------- (estado 5) ------------------------
582 estado_retirar_vaso:
583         rcall promedio_tabla
584
585         ldi XL, low(promedio)
586         ldi XH, high(promedio)
587
588         ldi YL, low(peso_umbral)
589         ldi YH, high(peso_umbral)
590
591         rcall cp24
592         ;si el promedio es menor al umbral, cambia de estado a Deteccion
593         brtc cambiar_a_DT
594         rjmp ret_estado_retirar_vaso
595
596 cambiar_a_DT:
597         clr estado
598         ori estado, 1<<DT_BIT
599
600 ret_estado_retirar_vaso:
601         rcall guardar_peso
602         ret
603
604
605 ;-----------------------------------------------------------------------
606 ;guarda el peso leido en una tabla de 8 valores de peso (cada uno de 3 bytes)
607 guardar_peso:
608         push gd_contador
609         push reg_posicion_tabla
610         push gd_temp
611         push XL
612         push XH
613         push YL
614         push YH
615
616         ;pongo en gd_contador el tamano de los datos de la tabla
617         ;(cuantos registros ocupa)
```

```
618         ldi gd_contador, TMNO_DATO
619
620         ;obtengo la posicion de donde guarde el dato menos reciente
621         ldi XL, low(posicion_tabla)
622         ldi XH, high(posicion_tabla)
623         ld reg_posicion_tabla, X
624
625         ldi XL, low(dato_hx711)
626         ldi XH, high(dato_hx711)
627         ldi YL, low(tabla_pesos)
628         ldi YH, high(tabla_pesos)
629
630 ;guardo el dato leido en la posicion de la tabla que seguia de la vez anterior
631         add YL, reg_posicion_tabla
632         brcc guardar_siguiente
633         inc YH
634
635 ;guarda la cantidad de registros que ocupa un dato
636 guardar_siguiente:
637         ld gd_temp, X+
638         st Y+, gd_temp
639         dec gd_contador
640         brne guardar_siguiente
641
642         ;incremento la posicion en un dato
643         subi reg_posicion_tabla, -TMNO_DATO
644
645         ;si llega al final de la tabla lo vuelve a 0
646         cpi reg_posicion_tabla, FIN_TABLA
647         brne ret_guardar_dato
648         ldi reg_posicion_tabla, 0
649
650 ret_guardar_dato:
651         ldi XL, low(posicion_tabla)
652         ldi XH, high(posicion_tabla)
653
654         st X, reg_posicion_tabla
655
656         pop YH
657         pop YL
658         pop XH
659         pop XL
660         pop gd_temp
661         pop reg_posicion_tabla
662         pop gd_contador
663
664         ret
665
666 ;------------------------ cambiar graduacion ------------------------------
667 ;cambia los LED del display para indicar la graduacion elegida
668 cambiar_graduacion:
669         push display
670
671         ;leemos en el registro display el puerto de los leds
672
673         ;comparamos uno a uno si estan encendidos los LEDs
674         ; cuando lee uno no prendido, lo prende y sale
675         sbis PORT_DISPLAY, LED_2
676         rjmp d_graduacion_20
677         sbis PORT_DISPLAY, LED_3
678         rjmp d_graduacion_30
679         sbis PORT_DISPLAY, LED_4
```

```
680         rjmp d_graduacion_40
681         sbis PORT_DISPLAY, LED_5
682         rjmp d_graduacion_50
683
684         ;si todos los LEDs estaban prendidos, reinicia el display
685         ;dejando prendido el LED1 (y deja los pines que no son led como estaban)
686         in display, PORT_DISPLAY
687         andi display, ~((1<<LED_2)|(1<<LED_3)|(1<<LED_4)|(1<<LED_5))
688         out PORT_DISPLAY, display
689         rjmp ret_modificar_display
690
691 d_graduacion_20:
692         sbi PORT_DISPLAY, LED_2
693         rjmp ret_modificar_display
694 d_graduacion_30:
695         sbi PORT_DISPLAY, LED_3
696         rjmp ret_modificar_display
697 d_graduacion_40:
698         sbi PORT_DISPLAY, LED_4
699         rjmp ret_modificar_display
700 d_graduacion_50:
701         sbi PORT_DISPLAY, LED_5
702         rjmp ret_modificar_display
703
704 ret_modificar_display:
705         pop display
706         ret
707 ;-------------------- obtener graduacion ----------------------
708 ;obtiene la graduacion elegida a partir del display y la guarda
709 ;en el registro "graduacion"
710 obtener_graduacion:
711
712         ;la graduacion debe estar al menos en 10% (graduacion = 1)
713         ldi graduacion, 1
714
715         ;chequeo si los leds estan prendidos, y si lo estan, aumenta la graduacion
716         ;en 1, cuando ve el primero sin prender, sale
717         sbis PORT_DISPLAY, GRAD_20
718         rjmp ret_obtener_graduacion
719         inc graduacion
720
721         sbis PORT_DISPLAY, GRAD_30
722         rjmp ret_obtener_graduacion
723         inc graduacion
724
725         sbis PORT_DISPLAY, GRAD_40
726         rjmp ret_obtener_graduacion
727         inc graduacion
728
729         sbis PORT_DISPLAY, GRAD_50
730         rjmp ret_obtener_graduacion
731         inc graduacion
732
733 ret_obtener_graduacion:
734         ret
735 ;----------------------------------------------------------
736
737 activar_timer:
738         push r20
739         lds r20, TIMSK1
740         ori R20, 1<<TOIE1
741         sts TIMSK1, r20
```

```
742         pop r20
743         ret
744
745 desactivar_timer:
746         push r20
747         lds r20, TIMSK1
748         andi r20, ~(1<<TOIE1)
749         sts TIMSK1, r20
750         pop r20
751         ret
752
753 abrir_bomba_1:
754         sbi PORT_BOMBAS, BOMBA_1
755         ret
756 cerrar_bomba_1:
757         cbi PORT_BOMBAS, BOMBA_1
758         ret
759 abrir_bomba_2:
760         sbi PORT_BOMBAS, BOMBA_2
761         ret
762 cerrar_bomba_2:
763         cbi PORT_BOMBAS, BOMBA_2
764         ret
765 ;--------------------------- leer_hx711 ----------------------------------------
766 ;lee el peso obtenido por el modulo amplificador conversor
767 ;y lo guarda en la ram en la posicion "dato_hx711"
768 leer_hx711:
769         push NRO_BITS_HX711
770         push A
771         push peso_leido_L
772         push peso_leido_M
773         push peso_leido_H
774         push XL
775         push XH
776         ;limpio el carry porque lo voy a usar
777         clc
778         ;limpio los registros que voy a usar
779         clr peso_leido_L
780         clr peso_leido_M
781         clr peso_leido_H
782
783         ;habilito la conversion de datos si no estaba activada
784         cbi port_ADSK, ADSK
785
786         ;si no termino la conversion vuelve a chequear ADDO
787 AD_not_finished:
788         sbic pin_ADDO, ADDO
789         rjmp AD_not_finished
790
791         ;cargo el contador r19 con 24 para pasar 24 bits
792         ldi NRO_BITS_HX711, 24
793
794 ShiftOut:
795         ;mando un pulso de clock
796         sbi port_ADSK, ADSK
797         ;se necesita delay de 1us aproximadamente,
798         ;usamos 18 ciclos de maquina (con freq=16MHz), por lo que tarda 1,125us
799         rcall T_high
800         cbi port_ADSK, ADSK
801
802         ;guarda el dato leido en el carry
803         sbic pin_ADDO, ADDO
```

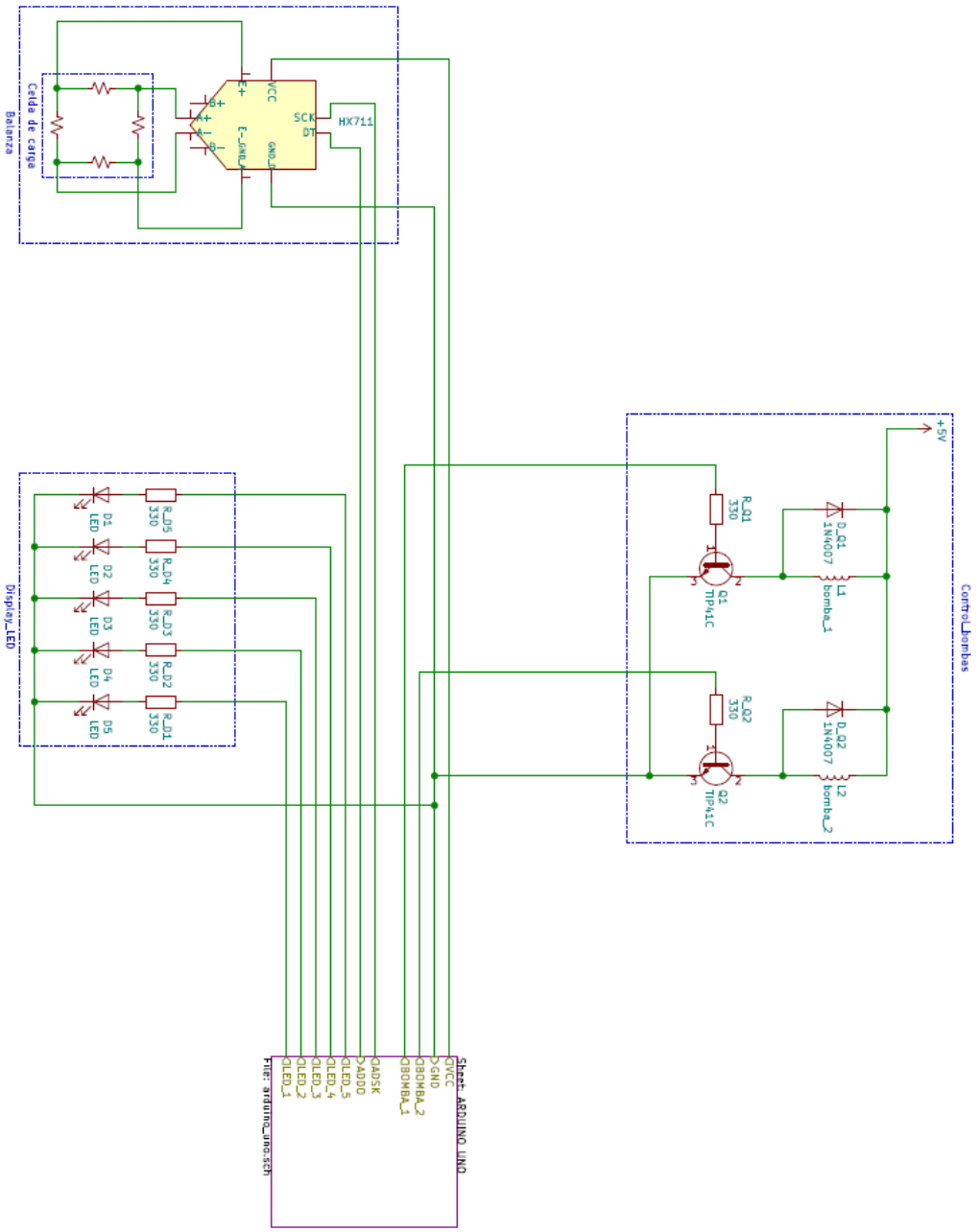
```
804         sec
805
806         ;guarda el bit leido en los registros
807         mov A,peso_leido_L
808         rol A
809         mov peso_leido_L ,A
810         mov A,peso_leido_M
811         rol A
812         mov peso_leido_M ,A
813         mov A,peso_leido_H
814         rol A
815         mov peso_leido_H ,A
816         ;chequeo si movio los 24 bits
817         dec r19
818         brne ShiftOut
819
820         ;vuelvo a poner el clock en 1 cuando termina y asi
821         ;pone a DOUT en alto nuevamente
822         sbi port_ADSK, ADSK
823         rcall T_high
824
825         ;el clock debe terminar en bajo
826         ;para no entrar en modo de bajo consumo del hx711
827         cbi port_ADSK, ADSK
828
829         ldi XL, low(dato_hx711)
830         ldi XH, high(dato_hx711)
831
832 ;guardo el dato en la SRAM
833         st X+, peso_leido_H
834         st X+, peso_leido_M
835         st X, peso_leido_L
836
837         pop XH
838         pop XL
839         pop peso_leido_H
840         pop peso_leido_M
841         pop peso_leido_L
842         pop A
843         pop NRO_BITS_HX711
844         ret
845
846 ;este delay dura 15 ciclos de maquina, sin contar el rcall
847 T_high:
848         push r16
849         ldi r16, 3
850 T_h_loop:
851         dec r16
852         brne T_h_loop
853
854         pop r16
855         ret
856 ;------------------------- promedio de tabla ----------------------------------
857 ;hace el promedio de los datos almacenados en la tabla
858 ;devuelve el resultado en la posicion "promedio" en sram
859 promedio_tabla:
860         push leido_L
861         push leido_M
862         push leido_H
863         push r19
864         push r20
865         push r21
```

```
866         push r22
867         push r23
868         push XL
869         push XH
870
871 ;limpio los registros acumuladores
872         clr r20
873         clr r21
874         clr r22
875         clr r23
876 ;puntero para obtener los datos que leo
877         ldi XL, LOW(tabla_pesos)
878         ldi XH, HIGH(tabla_pesos)
879
880         ;hace el promedio de LONG_TABLA pesos leidos
881         ldi r19, LONG_TABLA
882 sumar:
883
884         ld leido_H, X+
885         ld leido_M, X+
886         ld leido_L, X+
887
888         ;sumo de a 4 registros y acumulo el resultado en r23:r22:r21:r20
889         add r20, leido_L
890         adc r21, leido_M
891         adc     r22, leido_H
892         ;si hubo carry incrementa el registro mas significativo
893         brcc skip
894         inc r23
895 skip:
896         dec r19
897         brne sumar
898
899         ;divide por LONG_TABLA (tiene que ser pot de 2) el resultado
900         ldi r19, DIV_LONG_TABLA
901 dividir:
902         lsr r23
903         ror r22
904         ror r21
905         ror r20
906         dec r19
907         brne dividir
908
909         ;resto un valor pequeno para permitir un margen de tolerancia al servir
910         subi r20, 0x00
911         sbci r21, 0x19
912         sbci r22, 0x00
913         ;una vez obtenido el resultado del promedio de 4 medidas de peso
914         ;lo guarda en la sram (resultado en primeros 24 bits)
915         ldi XL, low(promedio)
916         ldi XH, high(promedio)
917         st X+, r22 ;peso_H
918         st X+, r21 ;peso_M
919         st X, r20  ;peso_L
920
921         pop XH
922         pop XL
923         pop r23
924         pop r22
925         pop r21
926         pop r20
927         pop r19
```

```
928         pop leido_H
929         pop leido_M
930         pop leido_L
931         ret
932
933 ;-----------------------------------------------------------------------
934 ;compara a los numeros de 3 bytes ubicados en los punteros X e Y
935 ;devuelve el resultado en el bit T del SREG
936 ;X<Y T = 0, X>=Y T = 1
937 cp24:
938         push r17
939         push r18
940         push r24
941         push XL
942         push XH
943         push YL
944         push YH
945         push cp24_temp
946         ;contador para comparar 3 registros
947         ldi cp24_temp, 3
948 loop_cpi24:
949         ld r17, X+
950         ld r18, Y+
951         cp r17, r18
952         brlo X_menor_a_Y
953         cp r18, r17
954         brlo X_mayor_o_igual_a_Y
955         dec cp24_temp
956         brne loop_cpi24
957
958 X_mayor_o_igual_a_Y:
959         set
960         rjmp ret_cp24
961
962 X_menor_a_Y:
963         clt
964         rjmp ret_cp24
965
966 ret_cp24:
967
968         pop cp24_temp
969         pop YH
970         pop YL
971         pop XH
972         pop XL
973         pop r24
974         pop r18
975         pop r17
976         ret
977 ;-----------------------------------------------------------------------
978
979
980
981
982 ;---------- interrupcion por timer overflow ---------------------
983 ISR_T1_OV:
984         push XL
985         push XH
986         push usart_escribir
987
988         rcall leer_hx711
989
```

```
990 ;transmito el resultado por el puerto serie para leerlo en la pantalla
991         ldi XL, low(dato_hx711)
992         ldi XH, high(dato_hx711)
993 
994         ld usart_escribir, X+
995         rcall USART_Transmit
996 
997         ld usart_escribir, X+
998         rcall USART_Transmit
999 
1000        ld usart_escribir, X
1001        rcall USART_Transmit
1002 
1003        pop usart_escribir
1004        pop XH
1005        pop XL
1006        reti
1007 
1008 ;-----------------------------USART0-----------------------------------------
1009 
1010 ;inicializar la comunicacion USART asincronica normal
1011 USART_Init:
1012        ; Setea baud rate (asume que el UBRR esta en R17(H):R16(L))
1013        sts UBRR0H, r17
1014        sts UBRR0L, r16
1015        ; habilita transmision y recepcion
1016        ldi r16, (1<<RXEN0)|(1<<TXEN0)
1017        sts UCSR0B,r16
1018        ; Setea formato de "frame"(bits de la comunicacion): 8data, 2stop bit
1019        ldi r16, (1<<USBS0)|(3<<UCSZ00)
1020        sts UCSR0C,r16
1021        ret
1022 ;----------------------------------------------------------------
1023 ;recibir datos de 5 a 8 bits en Usart_leido
1024 USART_Receive:
1025        push r21
1026        ; Espera a recibir dato
1027 loop_r:
1028        lds R21, UCSR0A
1029        sbrs R21, RXC0
1030        rjmp loop_r
1031        ; recibe los datos del buffer UDR0
1032        lds usart_leido, UDR0
1033 
1034        pop r21
1035        ret
1036 ;----------------------------------------------------------------
1037 ;transmitir 5 a 8 bits por usart_escribir
1038 USART_Transmit:
1039        push r20
1040        ; Espera a que el buffer de transmision este vacio
1041 loop_t:
1042        lds R20,UCSR0A
1043        sbrs R20,UDRE0
1044        rjmp loop_t
1045        ; pone el dato de R16 en el buffer de transmision y lo envia
1046        sts UDR0,usart_escribir
1047        pop r20
1048        ret
1049 ;----------------------------------------------------------------
```

7. Apéndice


Celda de carga
Balanza
HX711
VCC
SCK
DT
GND
Display_LED
D1
D2
D3
D4
D5
LED
R_D5
R_D4
R_D3
R_D2
R_D1
330
Control_bombas
+5V
R_Q1
R_Q2
Q1
Q2
TIP41C
D_Q1
D_Q2
1N4007
L1
L2
bomba_1
bomba_2
Sheet: ARDUINO_UNO
File: arduino_uno.sch
LED_1
LED_2
LED_3
LED_4
LED_5
A0SK
A0D0
BOMBA_2
BOMBA_1
GND
VCC

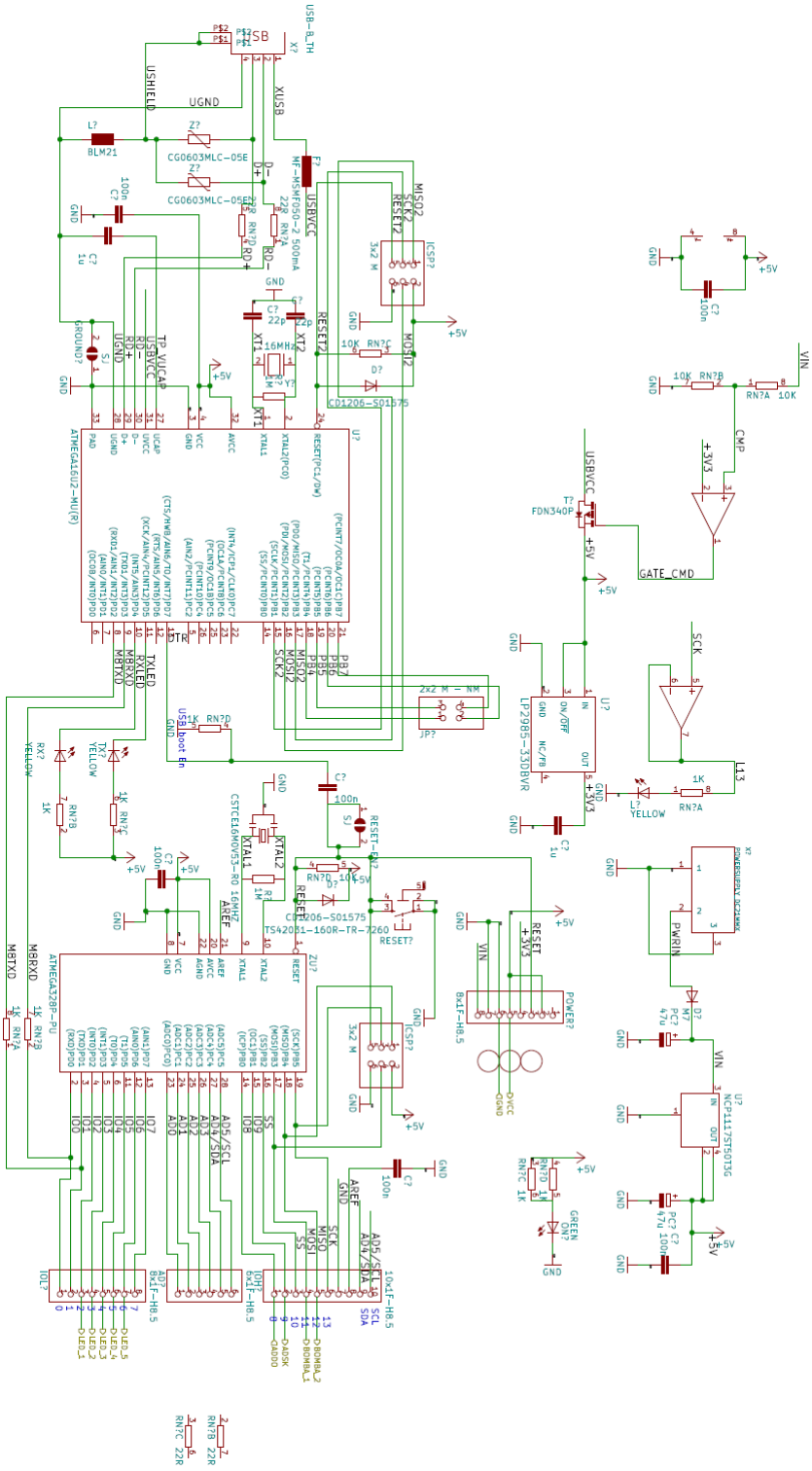


Figura 3: Esquemático del circuito implementado

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*