Universidad de Buenos Aires
Facultad de Ingeniería

86.07 - Laboratorio de microprocesadores

TP1: Manejo de puertos

Berard, Lucia Magdalena
101213 - lberard@fi.uba.ar

19 de mayo de 2021

En el siguiente trabajo práctico se analiza el procedimiento de un parpadeo de un LED utilizando un microcontrolador ATmega328P. Se pretende entender el lenguaje de programación Assembly mediante el uso de los registros de los puertos y la utilidad de la resistencia de pullup.

Índice

1. Introducción

1.1. LED

LED acrónimo de Light Emitting Diode o Diodo Emisor de Luz, es un dispositivo semiconductor que emite luz al circular a través de él una corriente eléctrica.

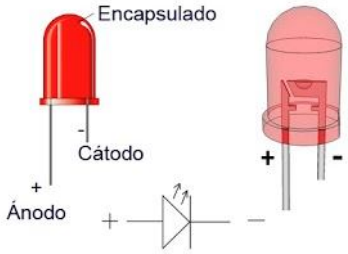

Figura 1: Diodo LED

Para calcular las resistencias exactas y obtener un desempeño óptimo se realiza a través de la siguiente fórmula:

$$R = \frac{V_{CC} - V_{LED}}{I_{LED}} \tag{1}$$

donde V_{CC} es la tensión de alimentación y V_{LED} es el voltaje necesario para el funcionamiento del LED, generalmente está entre 1.7 y 3.3 V, depende del color del diodo y de la composición de metales.

1.2. Microcontrolador

Para el trabajo práctico se utilizó un Arduino UNO, el cual sirvió como programador para el microcontrolador ATmega328P:

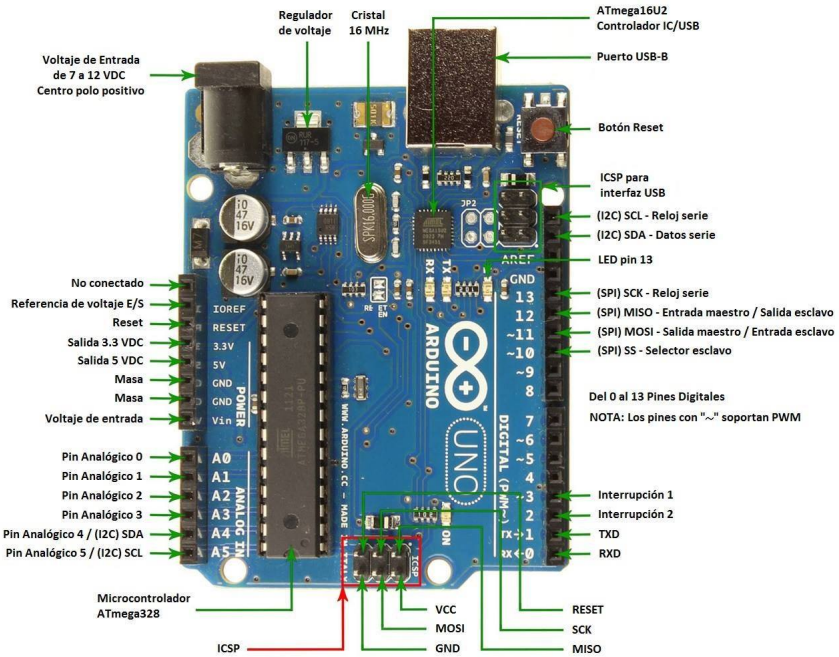

Figura 2: Arduino UNO

El microcontrolador tiene varios puertos de los cuales se puede hacer uso, estos puertos se pueden configurar como entrada o como salida. Para poder hacer esto es necesario escribir en los registros del puerto para darle las instrucciones necesarias.

Existen tres principales formas de controlar los puertos:

The Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Figura 3: Formas de controlar los puertos. Ejemplo con PORT B

1.2.1. DDR

Para hacer que un puerto se comporte como entrada o salida, es necesario setear el DDR, este registro no activará ni desactivará ningún pin del microcontrolador, simplemente le indicará al puerto si este será entrada o salida.

Para indicarle al DDR si el puerto será de entrada o salida, el 1 indica salida, y el 0 entrada, se le puede escribir como hexadecimal, decimal, o binario.

1.2.2. PORT

El PORT controla la salida del puerto, este se usa en caso de que el DDR haya sido seleccionado como salida, un 1 en el PORT indica un nivel alto en el puerto como salida, un 0 indica que el pin estará en nivel bajo.

1.2.3. PIN

El PIN es un registro de lectura, este registro da un 1 si el pin del microcontrolador tiene una tensión mayor a V_{ih} (limite de tensión de entrada por arriba del cual se considera 1 lógico) , y un cero si el pin presenta una tensión menor a V_{iL} (la tensión de entrada low por debajo de la cual se considera 0 lógico.

En este caso el valor del PIN se le puede asignar a una variable la cual guardará el valor del mismo, al momento de ejecutar la instrucción.

1.3. Materiales y entorno de desarrollo

La programación del microcontrolador se implementó con código en `Assembly` con la ayuda de la plataforma de desarrollo de Atmel, Atmel Studio 7.0. En cuanto la parte práctica se precisaron los siguientes materiales:

- 1 Arduino UNO
- Cable USB Arduino
- 1 protoboard
- 2 resistencias 10KΩ
- 1 resistencia 220 Ω
- 2 pulsadores
- Cables

2. Desarrollo

2.1. Parpadeo de un LED

Como en cualquier aproximación a un nuevo lenguaje de programación se pretende realizar un "Hola Mundo", que en el caso del Arduino es realizar un programa que haga parpadear un LED.

Figura 4: Diagrama en bloques

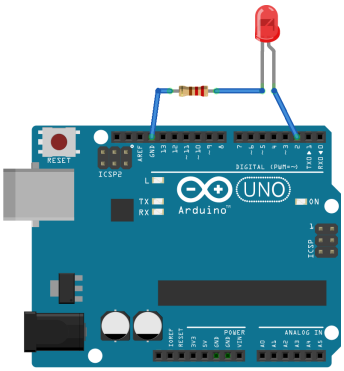

Figura 5: Diagrama esquemático

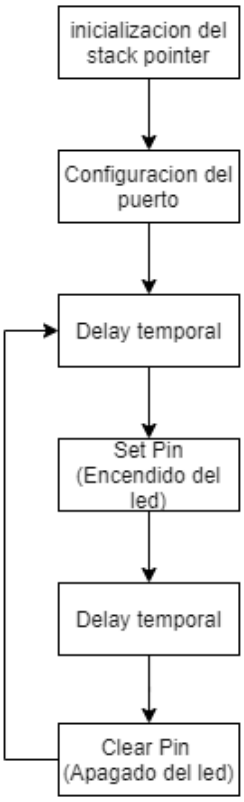

Figura 6: Diagrama de flujo

```
.include "m328pdef.inc"
;---------------------------------------------------
.equ LED_PIN        = 2
;---------------------------------------------------
.DSEG
.ORG SRAM_START
;---------------------------------------------------
.cseg
.org 0x0000
        rjmp main
.org INT_VECTORS_SIZE
;---------------------------------------------------
main:
        ;Inicializacion del stack pointer
        LDI             R16,HIGH(RAMEND)
    OUT             SPH,R16
    LDI             R16,LOW(RAMEND)
    OUT             SPL,R16
        ; Configuro puerto B
        ldi             r20,0xff
        out             DDRD,r20

;---------------------------------------------------
blink:
        sbi PORTD,LED_PIN ; encendido del led
        rcall delay
        cbi PORTD,LED_PIN ; apagado del led
        rcall delay
        rjmp blink
;---------------------------------------------------
delay:
        ldi             r20, 100
        loop3:
                ldi             r21, 100
        loop2:
                ldi             r22, 100
        loop1:
                dec             r22
                brne        loop1
                dec             r21
                brne        loop2
                dec             r20
                brne        loop3
                ret
```

2.2. Encendido de un LED con pulsadores

Se modificó el programa para que prenda un LED cuando se presiona el pulsador 1 y quede parpadeando hasta que se apague cuando se presiona el botón 2. El LED está conectado a un pin del microcontrolador y los pulsadores a otros dos pines.

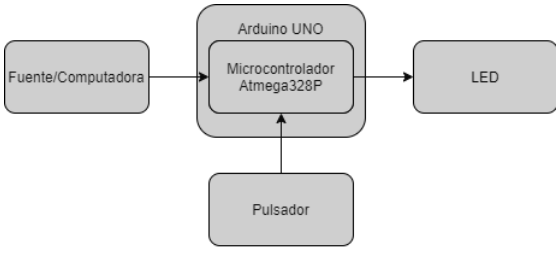

Figura 7: Diagrama en bloques

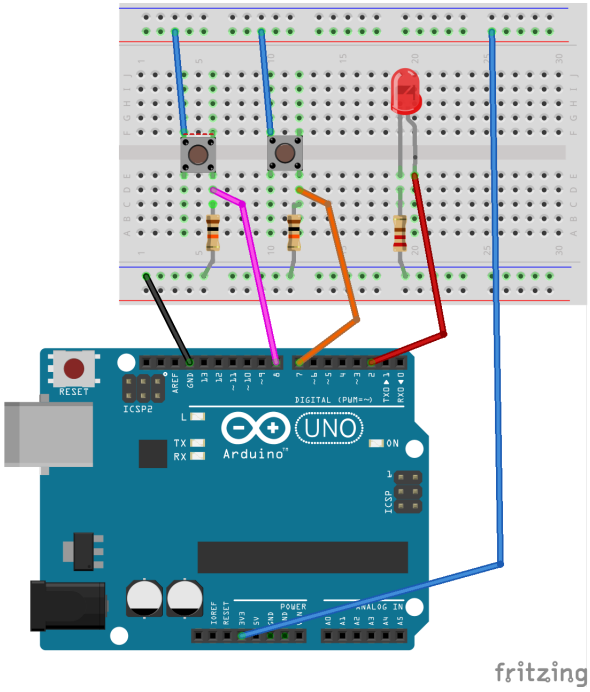

Figura 8: Diagrama esquemático

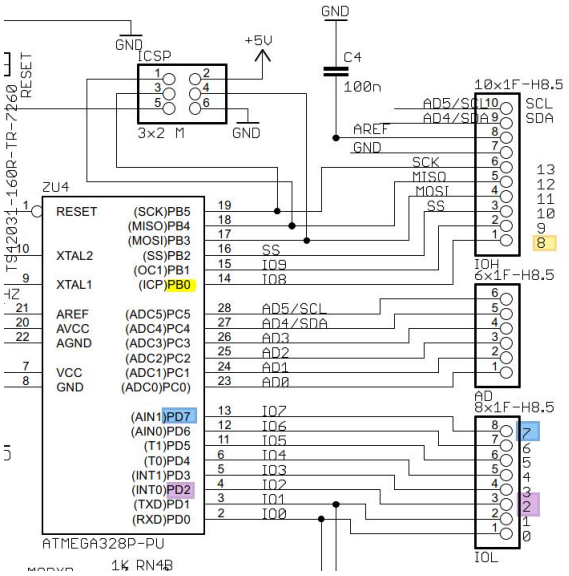

Figura 9: Puertos utilizados - Zoom del circuito interno del Arduino UNO. Conexiones del microcontrolador Atmega328p con los pines del Arduino

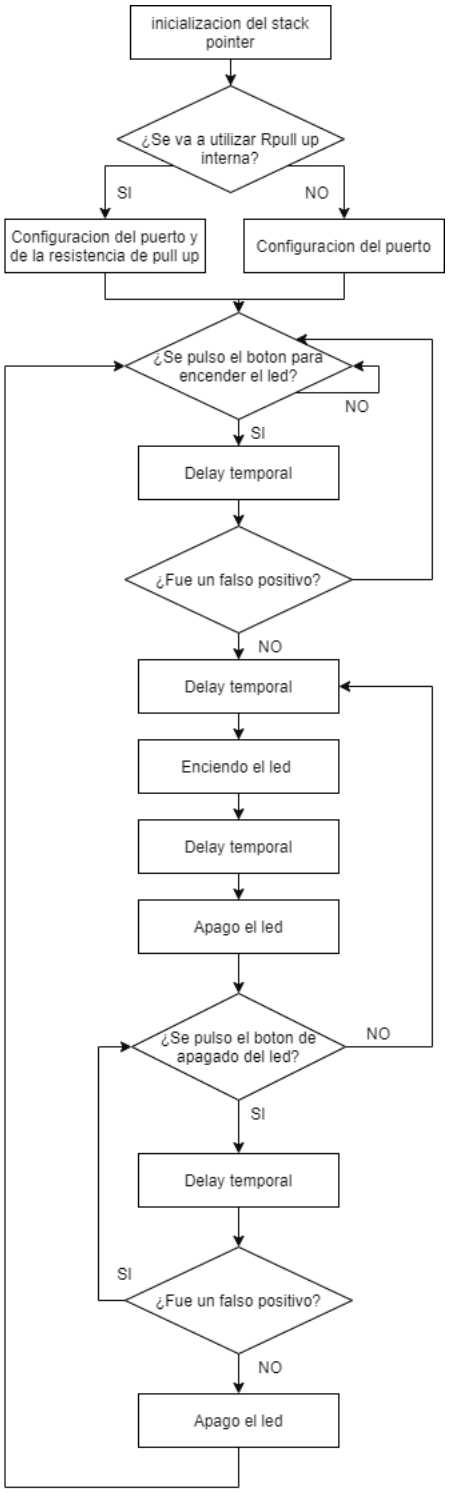

Figura 10: Diagrama de flujo

```
;-----------------------------------------------------------------
.include "m328pdef.inc"

.equ PBUTTON_1          = 0             ;PortB
.equ PBUTTON_2          = 7             ;PortD
.equ LED_PIN            = 2             ;PortD

;-----------------------------------------------------------------
.DSEG
.ORG SRAM_START

;-----------------------------------------------------------------
```

```
13  .cseg
14  .org 0x0000
15  rjmp config
16
17  .org INT_VECTORS_SIZE
18
19  config:
20          ;Inicializacion del stack pointer
21          LDI             R16,HIGH(RAMEND)
22      OUT             SPH,R16
23      LDI             R16,LOW(RAMEND)
24      OUT             SPL,R16
25          ;Configuracion del puerto
26          ldi             r20,0xFF
27          out             DDRD,r20
28          cbi             PORTD,LED_PIN
29          ldi             r20,0x00
30          out             DDRB,r20
31          ;Configuracion R pull up
32          ;ldi             r20,0x00
33          ;out             PORTD,r20
34
35  main:
36          rcall       pbutton_input
37          rcall       mainloop
38          rjmp        main
39
40  ;-----------------------------------------------------------------------
41  ;Verifico si se presiono el pulsador de encendido
42  pbutton_input:
43          sbis        PINB, PBUTTON_1
44          rjmp        pbutton_input
45          rcall       delay
46          sbis        PINB, PBUTTON_1
47          rjmp        pbutton_input
48          ret
49          rcall       delay
50
51  ;Parpadeo-------------------------------------------------------------
52
53  mainloop:
54          ;Blink:
55          rcall       delay
56          sbi             PORTD,LED_PIN                   ;prendo
57          rcall       delay
58          cbi             PORTD,LED_PIN
59          ;Espero el botón de apagado
60          sbis        PIND, PBUTTON_2
61          rjmp        mainloop
62          rcall       delay
63          sbis        PIND, PBUTTON_2
64          rjmp        mainloop
65          cbi             PORTD,LED_PIN                   ;apago el led
66  ret
67
68  ;Delay----------------------------------------------------------------
69  delay:
70                  ldi             r20, 100
71          loop3:
72                  ldi             r21, 100
73          loop2:
74                  ldi             r22, 100
75          loop1:
76                  dec             r22
77                  brne        loop1
78                  dec             r21
79                  brne        loop2
80                  dec             r20
81                  brne        loop3
82                  ret
```

Los valores utilizados en el delay en las líneas 70,72 y 74 pueden ser modificados para obtener una frecuencia de parpadeo mayor o menor segun como se prefiera. Para la resolución del ejercicio se planteo un valor intermedio.

2.3. Resistencia de pullup

Para evitar tener que utilizar las resistencias externas de $10K\Omega$:

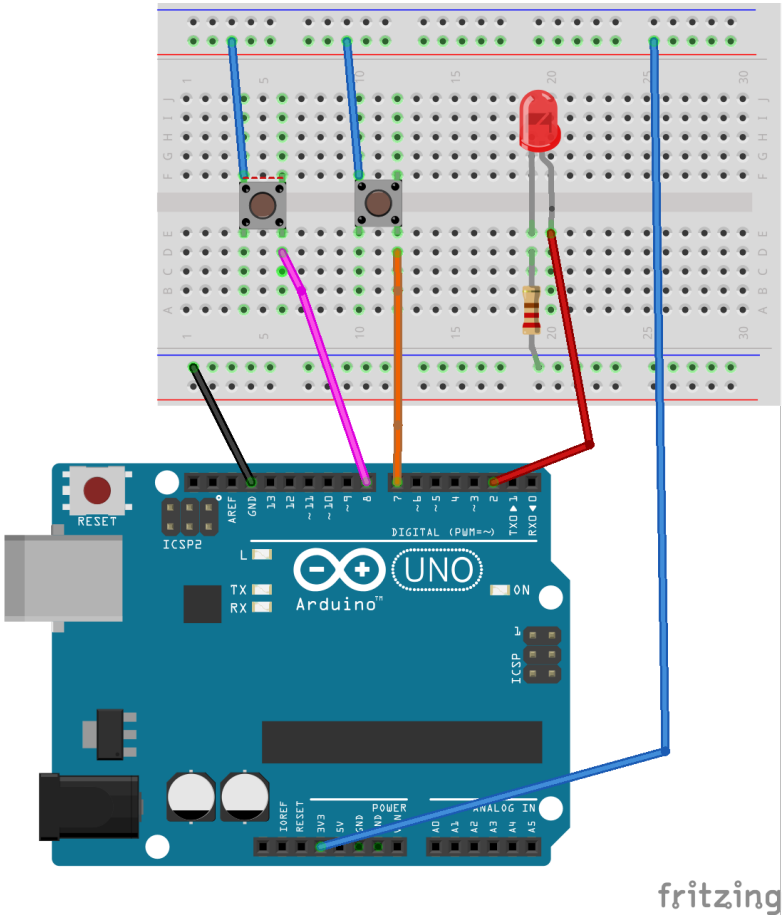

Figura 11: Diagrama esquemático

se puede utilizar la resistencia interna del Arduino cambiando la configuración de los puertos. Esto se logra descomentando las líneas 32 y 33 del ejercicio anterior obteniendo el siguiente código para la configuración:

```
1  config:
2          ;Inicializacion del stack pointer
3          LDI             R16,HIGH(RAMEND)
4      OUT             SPH,R16
5      LDI             R16,LOW(RAMEND)
6      OUT             SPL,R16
7          ;Configuracion del puerto
8          ldi             r20,0xFF
9          out             DDRD,r20
10         cbi             PORTD,LED_PIN
11         ldi             r20,0x00
12         out             DDRB,r20
13         ;Configuracion R pull up
14         ldi             r20,0x00
15         out             PORTD,r20
```

El resto del código no es necesario modificarlo.

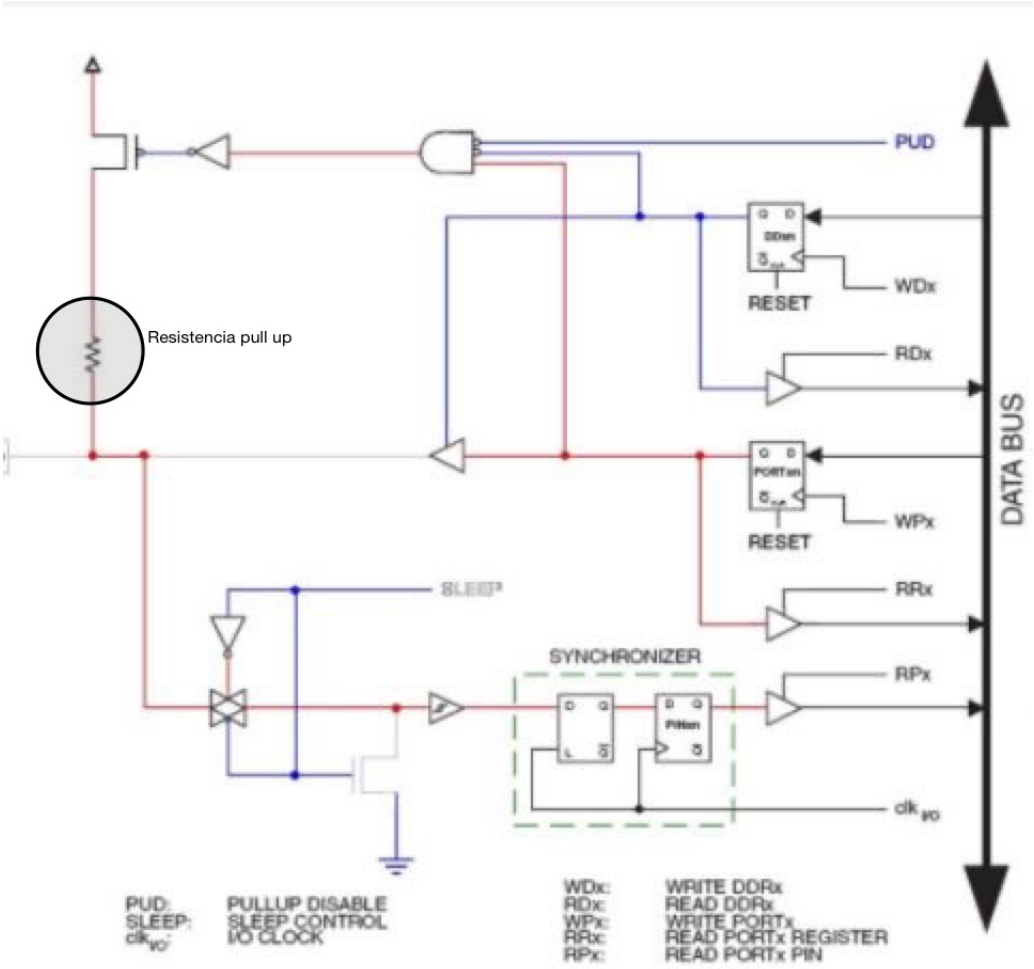

Figura 12: Parte interna del microcontrolador

3. Conclusiones

Durante el desarrollo del trabajo práctico se logró una mejor comprensión del comportamiento de los puertos y su manejo mediante el lenguaje de programación Assembler. Esto se logró afianzando lo visto en clase y con la ayuda de la documentación. (Ver Anexo)

Se tuvo mayor dificultad y demora en la creación de una rutina de retardo eficiente. En un principio se le habia seteado valores incorrectos por lo que no se podía apreciar el parpadeo. Resultó útil el entorno de desarrollo Atmel Studio 7 para poder debuggear y entender mejor el problema.

4. Anexo

4.1. Documentación

- AVR® Instruction Set Manual
- megaAVR® Data Sheet
- Circuito interno Arduino:

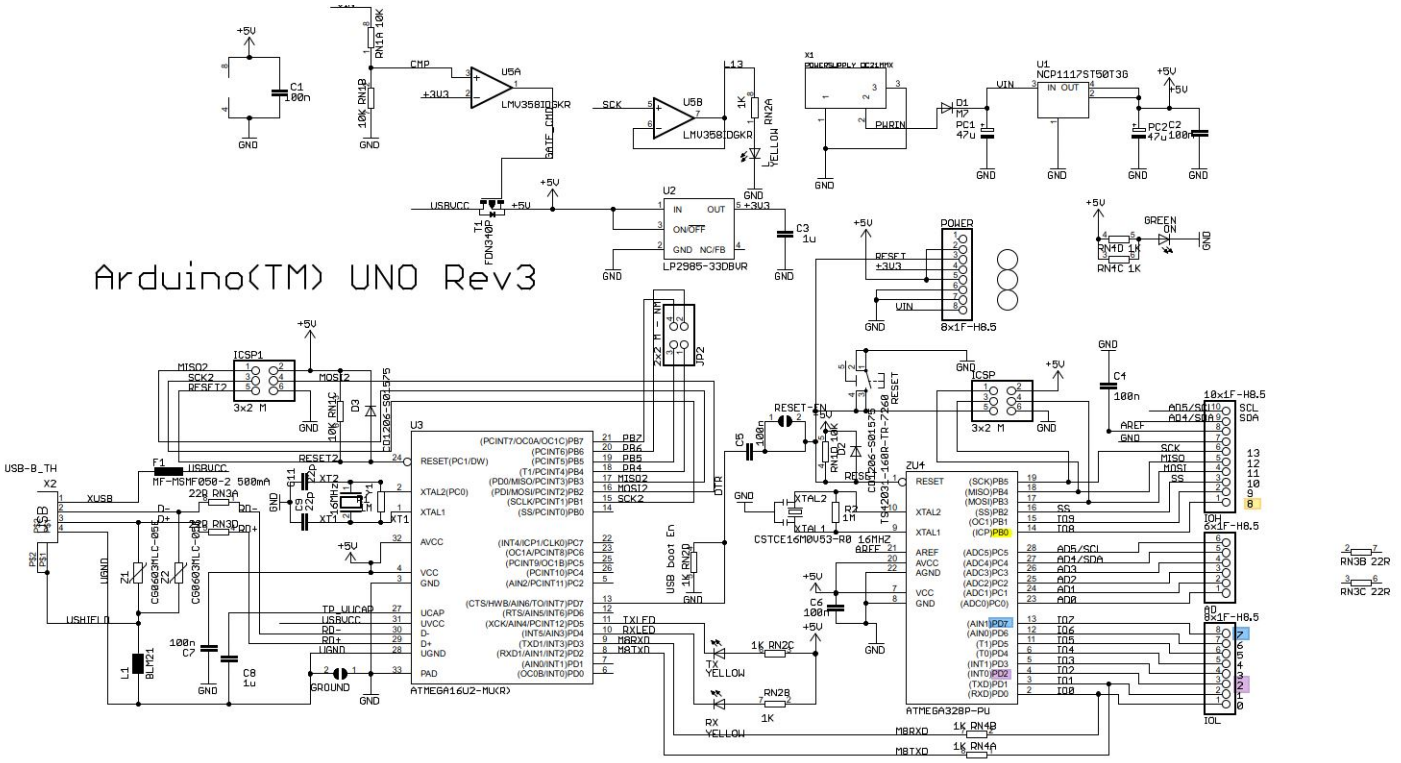

Figura 13: Circuito interno Arduino UNO

4.2. Repositorio en Github

hyperref `https://github.com/fiuba-labo-de-micro-miercoles/2021_1c_trabajos_practicos-lmberard/tree/master/TP1`

4.3. Vídeos de los ejercicios en funcionamiento

`https://drive.google.com/drive/folders/16qcHbCewz10kwfn_KdnIPQjYj25sXht_?usp=sharing`