

# Practica 3 - Memoria

Henry José, Chacón Barillas, 202002535  
*Escuela de Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos de Guatemala*

*La práctica consiste en realizar un programa en el cual debe seguir una secuencia respectiva utilizando botones, por lo que cada botón tiene a su disposición diferentes secuencias que están guardadas en los registros respectivos a una memoria*

## I. OBJETIVOS

- Utilizar los registros de memoria del microcontrolador.
- Comprender el funcionamiento de lectura y escritura de la memoria.
- Aplicar los conocimientos adquiridos en las prácticas anteriores.
- Calcular los valores necesarios de acuerdo al reloj del microcontrolador.

## II. MARCO TEÓRICO

### A. Keil μVision

Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) ampliamente utilizado para el desarrollo de software embebido, especialmente en microcontroladores basados en la arquitectura ARM. Fue desarrollado por Keil, una empresa que ahora es parte de ARM Holdings, y es uno de los entornos de desarrollo más populares para programar y depurar microcontroladores.

#### Características:

- Soporte de Microcontroladores:** Keil uVision es compatible con una amplia gama de microcontroladores basados en ARM, como los de la serie Cortex-M, y también soporta otros microcontroladores como los basados en 8051.
- Compilador ARM (ARMCC):** Incluye el compilador de ARM, que optimiza el código para la arquitectura ARM, permitiendo un rendimiento eficiente y un uso reducido de memoria.
- Depuración y Simulación:** Ofrece herramientas avanzadas de depuración y simulación que permiten al usuario probar y depurar el código en tiempo real o a través de simulaciones de hardware.

■ **Editor de Código:** Tiene un editor de código integrado con soporte para sintaxis destacada, autocompletado, y herramientas de búsqueda que facilitan la escritura y edición de código.

■ **Gestión de Proyectos:** Permite la gestión de proyectos con múltiples archivos, automatización de tareas de compilación y enlace, y configuración de opciones de compilación.

■ **Configuración de Periféricos:** Para microcontroladores compatibles, uVision incluye herramientas gráficas que permiten configurar fácilmente los periféricos, pines, y otros recursos del microcontrolador.

■ **Soporte para Lenguaje Ensamblador y C/C++:** uVision permite escribir código en ensamblador, C y C++, lo que brinda flexibilidad en la programación y optimización del código para aplicaciones embebidas.

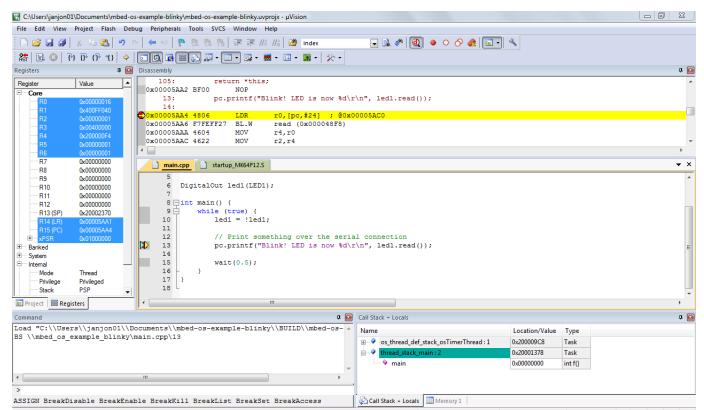


Figura 1: Entorno de Keil μVision

### III. DIAGRAMA DE FLUJO

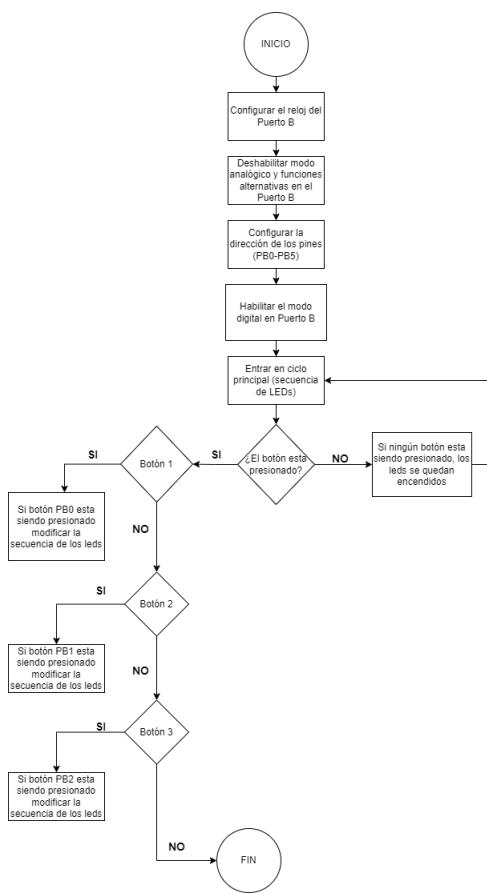


Figura 2: Algoritmo del programa realizado

### IV. DISEÑO EXPERIMENTAL

La siguiente práctica consiste en realizar un programa el cual genere 3 patrones diferentes de 5 secuencias utilizando 3 leds. Al iniciar el programa, este mostrara el primer patrón en los leds. Se deberá ingresar el mismo patrón utilizando 3 pulsadores que corresponderán a cada led. Si el patrón es correcto, se debe pasar al siguiente patrón de lo contrario se debe iniciar nuevamente con el primer patrón.

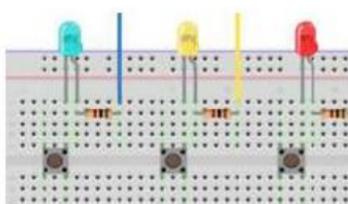


Figura 3: Circuito propuesto por el ingeniero

### A. Materiales

- Laptop
- Keil  $\mu$ Vision

### B. Procedimiento

#### 1. Configuración del Puerto B

- Activación del reloj para Puerto B: Se habilita el reloj para el puerto B mediante la manipulación del registro SYSCTL\_RCGCGPIO\_R.

#### 2. Deshabilitar el modo analógico y las funciones alternativas

- Se desactiva el modo analógico y las funciones alternativas en el puerto B (pines PB0-PB5) configurando los registros GPIO\_PORTB\_AMSEL\_R y GPIO\_PORTB\_PCTL\_R.

#### 3. Configuración de entradas y salidas

- Se especifica que los pines PB0, PB1, y PB2 funcionarán como entradas (botones), mientras que PB3, PB4, y PB5 funcionarán como salidas (LEDs). Esto se realiza mediante el registro GPIO\_PORTB\_DIR\_R.

#### 4. Habilitar el modo digital

- Se habilita el modo digital para los pines PB0-PB5 utilizando el registro GPIO\_PORTB\_DEN\_R.

#### 5. Ciclo Principal (Secuencias de LEDs)

##### Secuencias de LEDs:

- El programa entra en un ciclo principal donde se alternan diferentes patrones de encendido y apagado de los LEDs conectados a PB3, PB4 y PB5. Dependiendo del estado de los botones conectados a PB0, PB1 y PB2, se alterna entre distintas secuencias.

##### Control de botones:

- Se monitorea el estado de los botones (PB0, PB1, PB2) y, según el botón que esté presionado, se ajusta la secuencia de encendido de los LEDs.

##### Retorno a secuencia principal:

- Si no se detecta ningún cambio en los botones, se repiten las secuencias predeterminadas de LEDs.

## V. CÓDIGOS

### A. Código para el calculo del seno

```

1   SYSCTL_RCGCGPIO_R    EQU 0x400FE608 ;Reloj de
2   tiva
3
4   GPIO_PORTB_AMSEL_R    EQU 0x40005528 ;Modo
5   analogico
6   GPIO_PORTB_PCTL_R    EQU 0x4000552C ;
7   Desactivar funcion alternativa
8   GPIO_PORTB_DIR_R     EQU 0x40005400 ;
9   Especificacion de direccion
10  GPIO_PORTB_AFSEL_R   EQU 0x40005420 ;Funciones
11  alternativas
12  GPIO_PORTB_DEN_R    EQU 0x4000551C ;Modo
13  digital
14
15  PB0 EQU 0x40005004 ; PUSH 1
16  PB1 EQU 0x40005008 ; PUSH 2
17  PB2 EQU 0x40005010 ; PUSH 3
18  PB3 EQU 0x40005020 ; LED 1
19  PB4 EQU 0x40005040 ; LED 2
20  PB5 EQU 0x40005080 ; LED 3
21  ;tressegundos EQU 6000000
22  tressegundos EQU 1000000
23
24
25  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
26  ;;;;;;;;;;;;;;;;;;; CONFIGURACION PUERTO B
27  ;;;;;;;;;;;;;;;;;;;;;
28
29  LDR R1, =SYSCTL_RCGCGPIO_R ;activa el reloj
30  para el puerto B
31  LDR R0, [R1] ; carga en R0 el valor
32  almacenado al que apunta R1
33  ORR R0, R0, #2_00000010 ; escribir bit 0
34  para activar el reloj del puerto B
35  STR R0, [R1]
36  NOP ;Espera tiempo para que el
37  reloj se establezca
38  NOP
39
40  LDR R1, =GPIO_PORTB_AMSEL_R ; deshabilita
41  modo analogico
42  LDR R0, [R1] ; carga en R0 el valor
43  almacenado al que apunta R1
44  ;BIC R0, #0xOF ; deshabilita
45  analogico para PBO - PB3
46  BIC R0, #2_00000000 ; deshabilita
47  analogico para PBO - PB3
48  STR R0, [R1] ; carga en R0 el valor
49  almacenado al que apunta R1
50
51  LDR R1, =GPIO_PORTB_PCTL_R ; configura
52  puertos como GPIO
53  LDR R0, [R1] ; carga en R0 el valor
54  almacenado al que apunta R1
55  ;BIC R0, R0, #0x00000FF ; realiza un AND
56  entre el primer numero y el complemento del
57
58  ;do
59  ;BIC R0, R0, #0x0000FF00
60  BIC R0, #2_11111111 ; desactiva
61  funciones alternativas
62  STR R0, [R1] ; carga en R0 el valor
63  almacenado al que apunta R1
64
65  ;LDR R1, =GPIO_PORTB_DIR_R ; configura
66  registro de direccion
67  LDR R0, [R1] ; carga en R0 el valor
68  almacenado al que apunta R1
69  ;BIC R0, R0, #0x07 ; realiza un AND entre
70  el primer numero y el complemento del 2do
71  ;STR R0, [R1] ; carga en R0 el valor
72  almacenado al que apunta R1
73
74  LDR R1, =GPIO_PORTB_DIR_R ; configura
75  registro de direccion
76  LDR R0, [R1] ; carga en R0 el valor
77  almacenado al que apunta R1
78  ;ORR R0, R0, #0x08
79  ORR R0, R0, #2_11111000
80  STR R0, [R1] ; carga en R0 el valor
81  almacenado al que apunta R1
82
83  LDR R1, =GPIO_PORTB_AFSEL_R ; configura como
84  funcion regular
85  LDR R0, [R1] ; carga en R0 el valor
86  almacenado al que apunta R1
87  ;BIC R0, R0, #0x0F
88  BIC R0, R0, #2_00111111
89
90  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
91  ;STR R0, [R1] ; carga en R0 el valor
92  almacenado al que apunta R1
93  LDR R1, =GPIO_PORTB_DEN_R ; habilita el
94  puerto B como puerto digital
95  LDR R0, [R1] ; carga en R0 el valor
96  almacenado al que apunta R1
97  ;ORR R0, R0, #0x0F
98  ORR R0, R0, #2_00111111
99
100 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
101
102 ;;;;;;;;;;;;;;;;;;; Secuencia 1
103 BL Led1
104 BL Led2
105 BL Led3
106 BL Ledapagados
107
108 ;BL Lecturaa
109 ;BL ledencendidos
110 ;BL Ledapagados
111 ;BL Lecturab
112 ;BL ledencendidos
113 ;BL Ledapagados
114 ;BL Lecturac
115 ;BL ledencendidos
116 ;BL Ledapagados
117
118 ;;;;;;;;;;;;;;;;;;; Secuencia 2
119 BL Led3

```

```

95  BL Led2
96  BL Led1
97  BL Ledapagados
98
99  BL Lecturac
100 BL ledencendidos
101 BL Ledapagados
102 BL Lecturab
103 BL ledencendidos
104 BL Ledapagados
105 BL Lecturaa
106 BL ledencendidos
107 BL Ledapagados
108 ;;;;;;;;;;; Secuencia 3
109 BL Led2
110 BL Led1
111 BL Led3
112 BL Ledapagados
113
114 BL Lecturab
115 BL ledencendidos
116 BL Ledapagados
117 BL Lecturaa
118 BL ledencendidos
119 BL Ledapagados
120 BL Lecturac
121 BL ledencendidos
122 BL Ledapagados
123 ;;;;;;;;;;; Secuencia 4
124 BL Led3
125 BL Led1
126 BL Led2
127 BL Ledapagados
128
129 BL Lecturac
130 BL ledencendidos
131 BL Ledapagados
132 BL Lecturaa
133 BL ledencendidos
134 BL Ledapagados
135 BL Lecturab
136 BL ledencendidos
137 BL Ledapagados
138 ;;;;;;;;;;; Secuencia 5
139 BL Led2
140 BL Led3
141 BL Led1
142 BL Ledapagados
143
144 BL Lecturab
145 BL ledencendidos
146 BL Ledapagados
147 BL Lecturac
148 BL ledencendidos
149 BL Ledapagados
150 BL Lecturaa
151 BL ledencendidos
152 BL Ledapagados
153
154 BL ledencendidos
155 BL Ledapagados
156
157 BL ledencendidos
158 BL Ledapagados
159 BL ledencendidos
160 BL Ledapagados
161
162 B Principal
163
164

165 Delay
166 ADD R2, #1 ; suma #1 a R2
167 NOP ; Espera tiempo para que el
        reloj se estabilice
168 NOP
169 NOP
170 NOP
171 CMP R2, R4 ; compara R2 y R4
172 BNE Delay
173 BXEQ LR
174
175 Led1
176 LDR R1, =PB3 ; Enciende LED 1
177 LDR R0, =0x00
178 STR R0, [R1]
179
180 LDR R1, =PB4 ; Apagar LED 2
181 LDR R0, =0x10
182 STR R0, [R1]
183
184 LDR R1, =PB5 ; Apagar LED 3
185 LDR R0, =0x20
186 STR R0, [R1]
187
188 LDR R2, =0 ; resetea R2
189 B Delay
190
191 Led2
192 LDR R1, =PB3 ; Apagar LED 1
193 LDR R0, =0x08
194 STR R0, [R1]
195
196 LDR R1, =PB4 ; Enciende LED 2
197 LDR R0, =0x00
198 STR R0, [R1]
199
200 LDR R1, =PB5 ; Apagar LED 3
201 LDR R0, =0x20
202 STR R0, [R1]
203
204 LDR R2, =0 ; resetea R2
205 B Delay
206
207 Led3
208 LDR R1, =PB3 ; Apagar LED 1
209 LDR R0, =0x08
210 STR R0, [R1]
211
212 LDR R1, =PB4 ; Apagar LED 2
213 LDR R0, =0x10
214 STR R0, [R1]
215
216 LDR R1, =PB5 ; Enciende LED 3
217 LDR R0, =0x00
218 STR R0, [R1]
219
220 LDR R2, =0 ; resetea R2
221 B Delay
222
223 Ledapagados
224 LDR R1, =PB3 ; Apagar LED 1
225 LDR R0, =0x08
226 STR R0, [R1]
227
228 LDR R1, =PB4 ; Apagar LED 2
229 LDR R0, =0x10
230 STR R0, [R1]
231
232 LDR R1, =PB5 ; Enciende LED 3
233 LDR R0, =0x20

```

```

234 STR R0, [R1]
235
236 LDR R2, =0 ; resetea R2
237 B Delay
238
239 ledencendidos
240 LDR R1, =PB3 ; Apagar LED 1
241 LDR R0, =0x00
242 STR R0, [R1]
243
244 LDR R1, =PB4 ; Apagar LED 2
245 LDR R0, =0x00
246 STR R0, [R1]
247
248 LDR R1, =PB5 ; Enciende LED 3
249 LDR R0, =0x00
250 STR R0, [R1]
251
252 LDR R2, =0 ; resetea R2
253 B Delay
254
255 Lecturaa
256 LDR R1, =PB0 ; lee el pin B0
257 LDR R0, [R1]
258 CMP R0, #0x01 ; compara R0
259 ;BEQ Led3 ; Salto si es igual
260 ;BXEQ LR
261 BEQ Regresar
262
263 LDR R1, =PB1 ; lee el pin B1
264 LDR R0, [R1]
265 CMP R0, #0x02 ; compara R0
266 BEQ Principal ; salto si es igual
267
268 LDR R1, =PB2 ; lee el pin B2
269 LDR R0, [R1]
270 CMP R0, #0x04 ; compara R0
271 BEQ Principal ; salto si es igual
272
273 B Lecturaa
274
275 Lecturab
276 LDR R1, =PB0 ; lee el pin B0
277 LDR R0, [R1]
278 CMP R0, #0x01 ; compara R0
279 ;BEQ Led3 ; Salto si es igual
280 BEQ Principal
281
282 LDR R1, =PB1 ; lee el pin B1
283 LDR R0, [R1]
284 CMP R0, #0x02 ; compara R0
285 BEQ Regresar
286 ;BEQ Led3
287 ;BXEQ LR
288
289 LDR R1, =PB2 ; lee el pin B2
290 LDR R0, [R1] ; carga en R0
291 CMP R0, #0x04 ; compara R0
292 BEQ Principal ; salto si es igual
293
294 B Lecturab
295
296 Lecturac
297 LDR R1, =PB0 ; lee el pin B0
298 LDR R0, [R1]
299 CMP R0, #0x01 ; compara R0
300 ;BEQ Led3 ; Salto si es igual
301 ;BXEQ LR
302 BEQ Principal ; salto si es igual
303
304 LDR R1, =PB1 ; lee el pin B1
305 LDR R0, [R1]
306 CMP R0, #0x02 ; compara R0
307 BEQ Principal ; salto si es igual
308
309 LDR R1, =PB2 ; lee el pin B2
310 LDR R0, [R1] ; carga en R0
311 CMP R0, #0x04 ; compara R0
312 ;BEQ Principal ; salto si es igual
313 ;BXEQ LR ; Salto si es igual
314 BEQ Regresar
315
316 B Lecturac
317
318 Regresar
319 BX LR
320
321 ALIGN
322 END

```

Listing 1: Entorno de keil uvision

## VI. RESULTADOS

### A. Circuito en funcionamiento

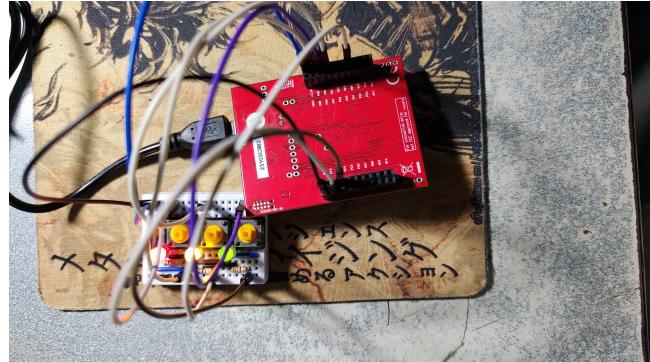


Figura 4: Circuito armado en protoboard

## VII. DISCUSIÓN DE RESULTADOS

### A. Control básico del Puerto B

- El código implementa correctamente la activación del puerto B y la configuración de los pines PB0-PB5 como entradas y salidas. Los pines PB3-PB5 están configurados como salidas para controlar tres LEDs, mientras que PB0-PB2 están configurados como entradas para recibir las señales de tres botones.

### B. Secuencia de LEDs

- El código realiza varias secuencias en las que los LEDs cambian de estado (encendido/apagado) en

diferentes patrones. La lógica principal está implementada en subrutinas llamadas Led1, Led2, Led3, Ledapagados, y ledencendidos, que controlan el estado de cada LED individualmente.

### C. Control de los botones

- Los botones conectados a PB0, PB1 y PB2 se leen continuamente. Dependiendo del botón que sea presionado, se ejecutan diferentes secuencias de LEDs. Esto permite una interacción dinámica entre el usuario y el microcontrolador, ya que las pulsaciones de los botones alteran el comportamiento del sistema.

### D. Estructura del delay

- Se utiliza un ciclo de delay controlado por un contador (R2) y un valor de referencia (R4) para ajustar el tiempo entre las acciones de encendido/apagado de los LEDs. Esto permite controlar la velocidad de parpadeo de los LEDs.

### E. Limitaciones y mejoras

- El código utiliza un mecanismo de polling para leer los botones, lo cual puede consumir recursos del microcontrolador innecesariamente. Sería más eficiente implementar interrupciones para detectar cuándo un botón ha sido presionado.
- Además, no se implementa un mecanismo de "debouncing" para los botones, lo que podría generar

lecturas incorrectas en caso de rebotes mecánicos.

## VIII. CONCLUSIONES

- Control funcional de LEDs y botones: El programa implementa correctamente el control de LEDs mediante la lectura de tres botones conectados al puerto B del microcontrolador. El sistema es capaz de cambiar dinámicamente entre varias secuencias de encendido/apagado de LEDs en función de la entrada del usuario.
- Interacción usuario-microcontrolador: La interacción con el usuario a través de botones es efectiva. La implementación de subrutinas independientes para cada secuencia de LEDs permite una fácil expansión o modificación del código para añadir nuevas secuencias.
- Posibles optimizaciones: El código podría beneficiarse de la implementación de interrupciones en lugar de un ciclo continuo de polling para la lectura de los botones. Esto reduciría el consumo de recursos del microcontrolador y haría el programa más eficiente.
- Aplicabilidad: Este tipo de código es adecuado para proyectos básicos de control de hardware en sistemas embebidos, como prototipos de interfaces físicas, entrenamiento de controladores y demostraciones de sistemas de entrada/salida digital.

## IX. REPOSITORIO

[https://github.com/Henry-Chacon/Practica\\_3\\_LAB\\_E5](https://github.com/Henry-Chacon/Practica_3_LAB_E5). [Fuente: Elaboración propia 2024].