

**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO.**

**INTRODUCCIÓN A LOS MICROCONTROLADORES.**

**REPORTE PRÁCTICA 9.**

**PROFESOR: AGUILAR SANCHEZ FERNANDO**

**Grupo: 3CM6**

**Alumno: Rojas Alvarado Luis Enrique**

**Boleta: 2014010995**

**Link de vídeo explicativo:** [**https://youtu.be/9bPdfw7DgEg**](https://youtu.be/9bPdfw7DgEg)

**OBJETIVO**

Los integrantes del equipo retomaran las habilidades adquiridas en las 8 prácticas anteriores para así poder realizar su primer proyecto.

**INTRODUCCIÓN**

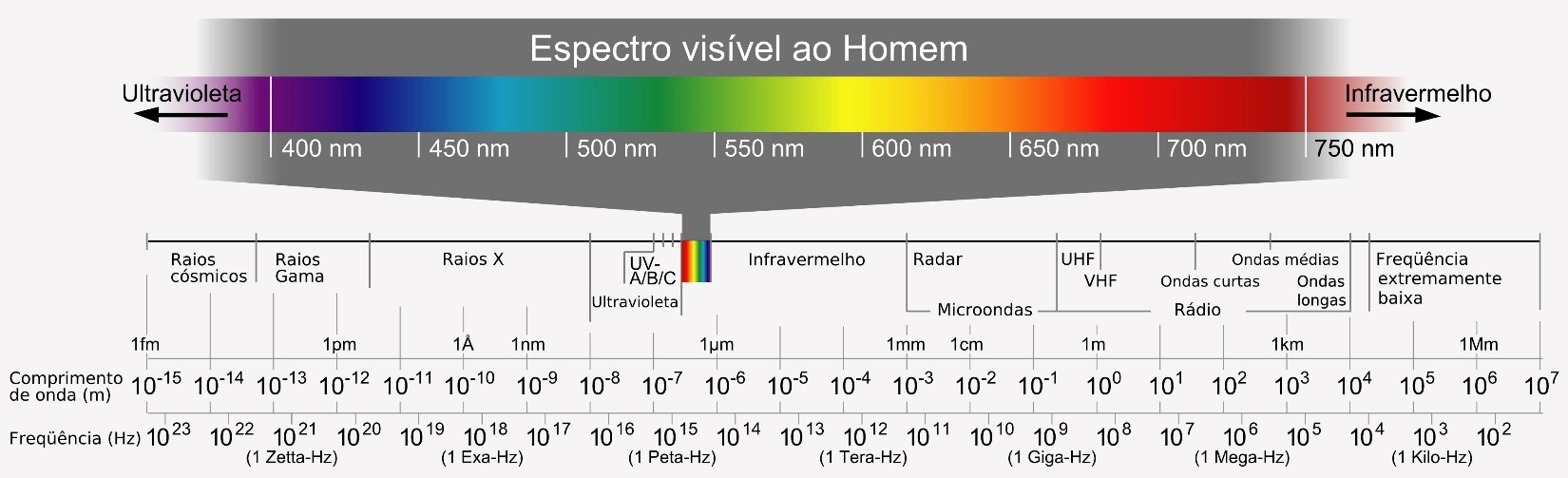
Los led’s infrarrojos son de 2 tipos:

Emisor: Los Led’s infrarrojos son usados en aplicaciones de control remoto, barreras fotoeléctricas, detección de objetos, entre otras aplicaciones; este led tiene un encapsulado de 5mm y funciona con una longitud de onda de 940nm.

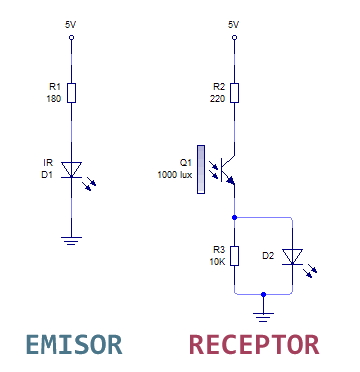
Receptor: El fotodiodo es un dispositivo que conduce una cantidad de corriente eléctrica proporcional a la cantidad de luz que lo ilumina o incide, este por lo general es usado en conjunto con un led emisor de infrarrojos en aplicaciones como detector de objetos, barreras fotoeléctricas o controles infrarrojos.

La palabra Led por si sola significa Diodo emisor de luz, mientras tanto IR significa rayos infrarrojos, la unión de estas palabras forma IRLED’s que significa LED`s Infrarrojos.

Cuando trabajamos con LED`s Infrarrojos debemos saber que no podremos ver a simple vista si estos están trabajando correctamente, esto se debe a que la luz o el espectro que emiten estos dispositivos es infrarrojo y no puede ser captada por los ojos humanos, en la siguiente escala se puede apreciar lo que se comenta en este párrafo:



En la siguiente imagen se puede apreciar perfectamente el diagrama de conexión para elaborar un circuito emisor y un receptor, utilizando LED`s Infrarrojos.

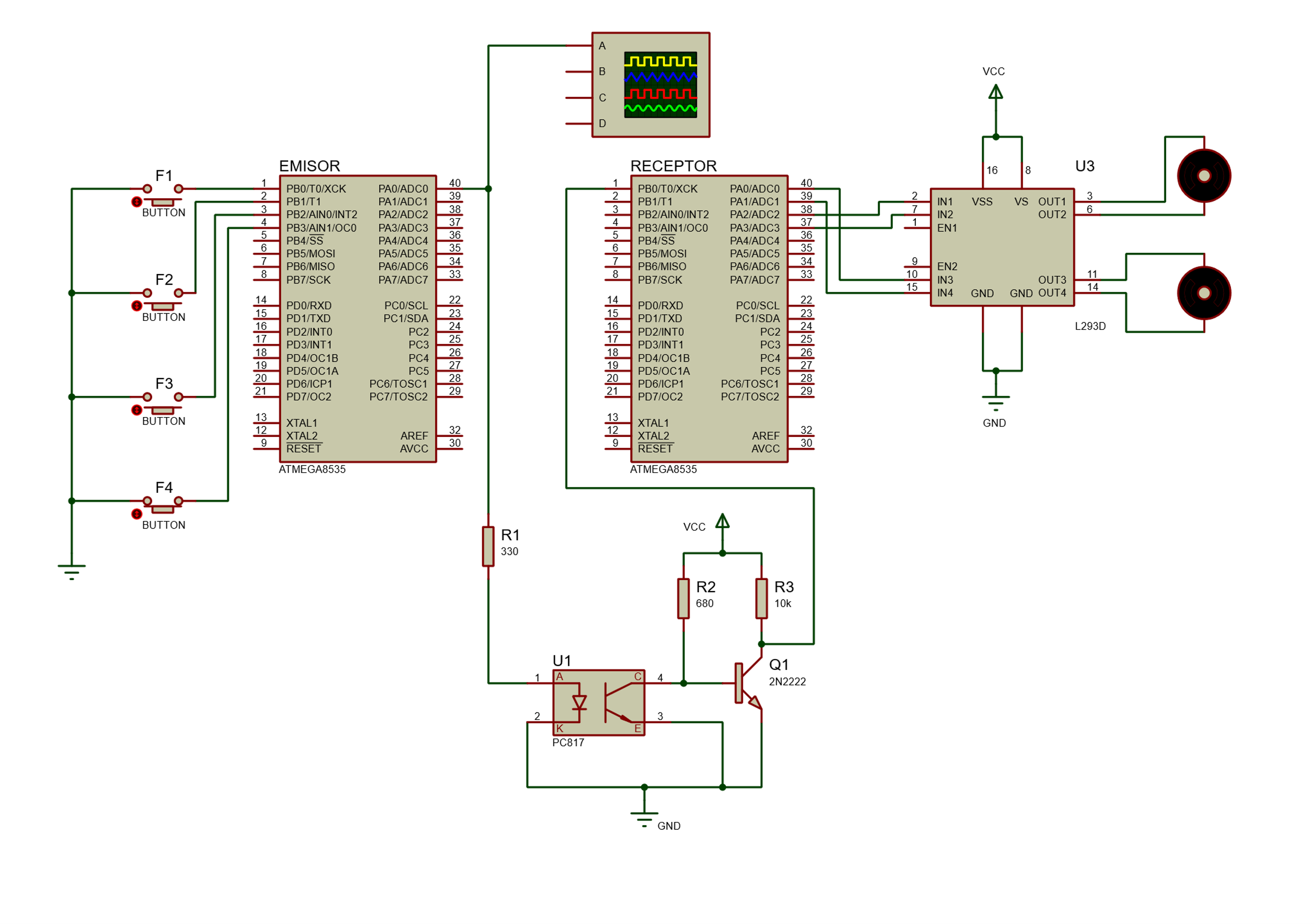


Una desventaja que poseen los LED`s Infrarrojos (IRLED’s) es que su señal puede ser interrumpida muy fácilmente, basta con colocar un objeto enfrente del emisor para que la señal no pueda llegar a su destino, debido a esto se dejo de utilizar estos sensores para algunas aplicaciones y fueron remplazados con tecnologías Bluetooth y Wifi.

**Desarrollo Experimental**

**1.-** Deberán lograr realizar un carro capaz de moverse hacia adelante, atrás, derecha e izquierda con un control mediante un par infrarrojo.

**Circuito en Proteus**

****

**Código Emisor**

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. This program was created by the
3. CodeWizardAVR V2.60 Evaluation
4. Automatic Program Generator
5. © Copyright 1998-2012 Pavel Haiduc, HP InfoTech s.r.l.
6. http://www.hpinfotech.com
8. Project :
9. Version :
10. Date    : 05/12/2020
11. Author  :
12. Company :
13. Comments:

16. Chip type               : ATmega8535
17. Program type            : Application
18. AVR Core Clock frequency: 1.000000 MHz
19. Memory model            : Small
20. External RAM size       : 0
21. Data Stack size         : 128
22. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
24. #include <mega8535.h>
25. #include <delay.h>
26. #define boton1 PINB.0
27. #define boton2 PINB.1
28. #define boton3 PINB.2
29. #define boton4 PINB.3
30. **void** main(**void**)
31. {
32. // Declare your local variables here
34. // Input/Output Ports initialization
35. // Port A initialization
36. // Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
37. DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) | (1<<DDA1) | (1<<DDA0);
38. // State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
39. PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);
41. // Port B initialization
42. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
43. DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0);
44. // State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
45. PORTB=(1<<PORTB7) | (1<<PORTB6) | (1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3) | (1<<PORTB2) | (1<<PORTB1) | (1<<PORTB0);
47. // Port C initialization
48. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
49. DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) | (0<<DDC0);
50. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
51. PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);
53. // Port D initialization
54. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
55. DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<DDD1) | (0<<DDD0);
56. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
57. PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);
59. // Timer/Counter 0 initialization
60. // Clock source: System Clock
61. // Clock value: Timer 0 Stopped
62. // Mode: Normal top=0xFF
63. // OC0 output: Disconnected
64. TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);
65. TCNT0=0x00;
66. OCR0=0x00;
68. // Timer/Counter 1 initialization
69. // Clock source: System Clock
70. // Clock value: Timer1 Stopped
71. // Mode: Normal top=0xFFFF
72. // OC1A output: Disconnected
73. // OC1B output: Disconnected
74. // Noise Canceler: Off
75. // Input Capture on Falling Edge
76. // Timer1 Overflow Interrupt: Off
77. // Input Capture Interrupt: Off
78. // Compare A Match Interrupt: Off
79. // Compare B Match Interrupt: Off
80. TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
81. TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (0<<CS10);
82. TCNT1H=0x00;
83. TCNT1L=0x00;
84. ICR1H=0x00;
85. ICR1L=0x00;
86. OCR1AH=0x00;
87. OCR1AL=0x00;
88. OCR1BH=0x00;
89. OCR1BL=0x00;
91. // Timer/Counter 2 initialization
92. // Clock source: System Clock
93. // Clock value: Timer2 Stopped
94. // Mode: Normal top=0xFF
95. // OC2 output: Disconnected
96. ASSR=0<<AS2;
97. TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) | (0<<CS20);
98. TCNT2=0x00;
99. OCR2=0x00;
101. // Timer(s)/Counter(s) Interrupt(s) initialization
102. TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);
104. // External Interrupt(s) initialization
105. // INT0: Off
106. // INT1: Off
107. // INT2: Off
108. MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
109. MCUCSR=(0<<ISC2);
111. // USART initialization
112. // USART disabled
113. UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);
115. // Analog Comparator initialization
116. // Analog Comparator: Off
117. ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);
118. SFIOR=(0<<ACME);
120. // ADC initialization
121. // ADC disabled
122. ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) | (0<<ADPS1) | (0<<ADPS0);
124. // SPI initialization
125. // SPI disabled
126. SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) | (0<<SPR0);
128. // TWI initialization
129. // TWI disabled
130. TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
132. **while** (1)
133. {
135. **if**(boton1==0){
136. PORTA.0=1;
137. delay\_us(2350);
138. PORTA.0=0;
139. delay\_us(2350);
141. }
142. **else** **if**(boton2==0){
144. PORTA.0=1;
145. delay\_us(620);
146. PORTA.0=0;
147. delay\_us(620);
149. }
150. **else** **if**(boton3==0){
152. PORTA.0=1;
153. delay\_us(350);
154. PORTA.0=0;
155. delay\_us(350);
157. }
158. **else** **if**(boton4==0){
159. PORTA.0=1;
160. delay\_us(135);
161. PORTA.0=0;
162. delay\_us(135);

165. }**else**{
166. PORTA.0=0;
167. }
169. }
170. }

**Código Receptor**

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. This program was created by the
3. CodeWizardAVR V2.60 Evaluation
4. Automatic Program Generator
5. © Copyright 1998-2012 Pavel Haiduc, HP InfoTech s.r.l.
6. http://www.hpinfotech.com
8. Project :
9. Version :
10. Date    : 05/12/2020
11. Author  :
12. Company :
13. Comments:

16. Chip type               : ATmega8535
17. Program type            : Application
18. AVR Core Clock frequency: 1.000000 MHz
19. Memory model            : Small
20. External RAM size       : 0
21. Data Stack size         : 128
22. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
24. #include <mega8535.h>
25. #include <delay.h>
26. #define boton PINB.0
27. bit botonp;
28. bit botona;
29. unsigned **long** **int** i=0;
30. **int** var=0;
32. **void** main(**void**)
33. {
34. // Declare your local variables here
36. // Input/Output Ports initialization
37. // Port A initialization
38. // Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
39. DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) | (1<<DDA1) | (1<<DDA0);
40. // State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
41. PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);
43. // Port B initialization
44. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
45. DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0);
46. // State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
47. PORTB=(1<<PORTB7) | (1<<PORTB6) | (1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3) | (1<<PORTB2) | (1<<PORTB1) | (1<<PORTB0);
49. // Port C initialization
50. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
51. DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) | (0<<DDC0);
52. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
53. PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);
55. // Port D initialization
56. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
57. DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<DDD1) | (0<<DDD0);
58. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
59. PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);
61. // Timer/Counter 0 initialization
62. // Clock source: System Clock
63. // Clock value: Timer 0 Stopped
64. // Mode: Normal top=0xFF
65. // OC0 output: Disconnected
66. TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);
67. TCNT0=0x00;
68. OCR0=0x00;
70. // Timer/Counter 1 initialization
71. // Clock source: System Clock
72. // Clock value: Timer1 Stopped
73. // Mode: Normal top=0xFFFF
74. // OC1A output: Disconnected
75. // OC1B output: Disconnected
76. // Noise Canceler: Off
77. // Input Capture on Falling Edge
78. // Timer1 Overflow Interrupt: Off
79. // Input Capture Interrupt: Off
80. // Compare A Match Interrupt: Off
81. // Compare B Match Interrupt: Off
82. TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
83. TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (0<<CS10);
84. TCNT1H=0x00;
85. TCNT1L=0x00;
86. ICR1H=0x00;
87. ICR1L=0x00;
88. OCR1AH=0x00;
89. OCR1AL=0x00;
90. OCR1BH=0x00;
91. OCR1BL=0x00;
93. // Timer/Counter 2 initialization
94. // Clock source: System Clock
95. // Clock value: Timer2 Stopped
96. // Mode: Normal top=0xFF
97. // OC2 output: Disconnected
98. ASSR=0<<AS2;
99. TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) | (0<<CS20);
100. TCNT2=0x00;
101. OCR2=0x00;
103. // Timer(s)/Counter(s) Interrupt(s) initialization
104. TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);
106. // External Interrupt(s) initialization
107. // INT0: Off
108. // INT1: Off
109. // INT2: Off
110. MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
111. MCUCSR=(0<<ISC2);
113. // USART initialization
114. // USART disabled
115. UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);
117. // Analog Comparator initialization
118. // Analog Comparator: Off
119. ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);
120. SFIOR=(0<<ACME);
122. // ADC initialization
123. // ADC disabled
124. ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) | (0<<ADPS1) | (0<<ADPS0);
126. // SPI initialization
127. // SPI disabled
128. SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) | (0<<SPR0);
130. // TWI initialization
131. // TWI disabled
132. TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
133. **while** (1)
134. {

137. **if** (boton==0){
138. botona=0;
139. }**else**{
140. botona=1;
141. }
143. **if** ((botonp==0)&&(botona==1)){ //hubo cambio de flanco de 0 a 1
144. var++;
145. } //Se incrementa la variable
146. botonp=botona;

149. i++;
151. **if**(i>260){
153. **if**(var>0 && var<=8){
154. PORTA=1;
155. }**else** **if**(var>8 && var<=24){
156. PORTA=1;
157. }**else** **if**(var>24 && var<=70){
158. PORTA=1;
159. }**else** **if**(var>70 && var<=200) {
160. PORTA=1;
161. }
163. i=0;
164. var=0;
165. PORTA=0;
166. }
168. i=0;
169. var=0;
170. PORTA=0;

173. }
174. }

**Conclusiones**

Lo que más se nos complicó en el proyecto fue acertar en el rango de los pulsos que generamos a través del emisor. El receptor fue relativamente sencillo, ya que solo se mandaban las indicaciones al puente h.

Aprendí la interacción del microprocesador con los periféricos asociados a esta práctica, reforzando así mis conocimientos de la arquitectura de este así como de los funcionamientos del periférico.

**Bibliografía**

[1] Circuitos electrónicos, “Sensor de infrarrojos (emisor y receptor). Recuperado el 06/12/2020

<http://www.circuitoselectronicos.org/2010/05/sensor-de-infrarrojos-emisor-y-receptor.html>