

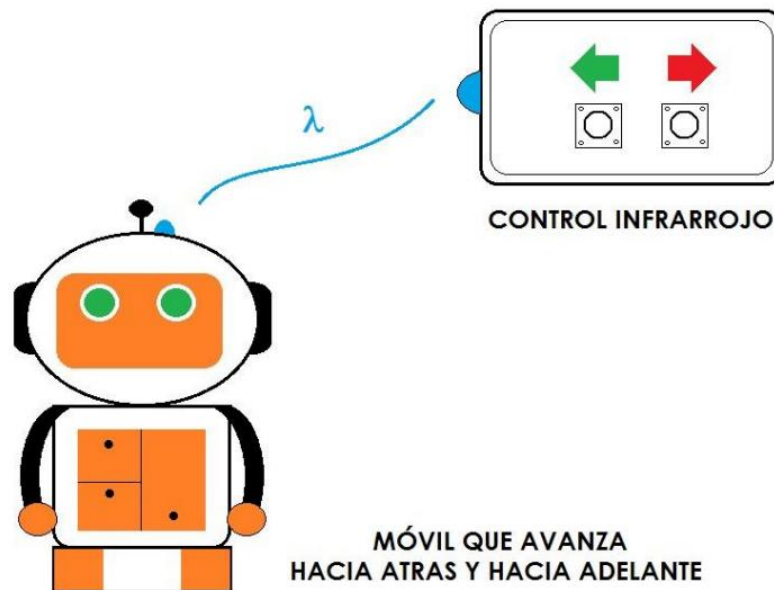


**INSTITUTO POLITÉCNICO NACIONAL**  
**ESCUELA SUPERIOR DE CÓMPUTO**

**SISTEMAS EN CHIP**

**PRACTICA No. 09**

**"PROYECTO 1"**



**ALUMNO:** RENTERIA ARRIAGA JOSUE.

**GRUPO:** 6SCM1

**PROFESOR:** FERNANDO AGILAR SÁNCHEZ.

18/NOVIEMBRE/2022

## **I. OBJETIVO GENERAL:**

Diseñe un móvil controlado vía infrarrojo, solo controle movimiento hacia atrás y hacia adelante como lo indican las flechas.

## **II. INTRODUCCIÓN TEÓRICA**

En este proyecto podremos saber utilizar correctamente y aplicar la implementación del código en el lenguaje de programación C de los conceptos de implementar flancos de subida y bajada, implementación de puentes H para controlar motores e implementación de infrarrojos que se pueda observar en el movimiento de unos motores y ver cómo se comportan al cambiar los valores de entrada con botones.

En este proyecto se estudiará la implementación de un Programa y Circuito para programar dos ATmega, el primero ATmega recibirá 4 botones de entrada (izquierda, derecha, arriba y abajo) y dependiendo el botón que se oprima tendremos una salida diferente, la salida estará dada con el emisor del par infrarrojo (dependiendo la opción se enviara un rango de señales para saber que opción se desea), el segundo ATmega recibirá la señal que nos envió el primero y se procesara para sacar en 4 puertos la salida, que se mostrara en el movimiento de los motores. La práctica nos mostrara en los motores como cambia el valor dependiendo del valor de nuestras entradas dadas por los motores. En este caso del proyecto se ocuparán algunos conceptos obtenidos anteriormente en el curso , mas que nada los conceptos de todas las anteriores practicas ya realizadas.

Ocuparemos algunos elementos como lo dice el formato de la practica ya que ocuparemos el simulador Code Vision para poder implementar el código para después programar nuestros ATmega8535, en mi caso se ocupará el programa Khazama AVR con su programador para poder programar nuestro ATmega con el archivo .hex que se generó anteriormente. Para la parte del armado del circuito se utilizarán botones, un par infrarrojo, un puente H y 2 motores.

Algunos conceptos que se necesitan para poder entender algunos conceptos de la practica son los siguientes:

Un control remoto infrarrojo (IR) **envía señales de luz infrarroja**. Una luz infrarroja no puede verse a simple vista, pero puede ser visible con el uso de una cámara digital, la cámara del teléfono celular o videocámara. Los controles remotos de IR llevan el símbolo (IR).

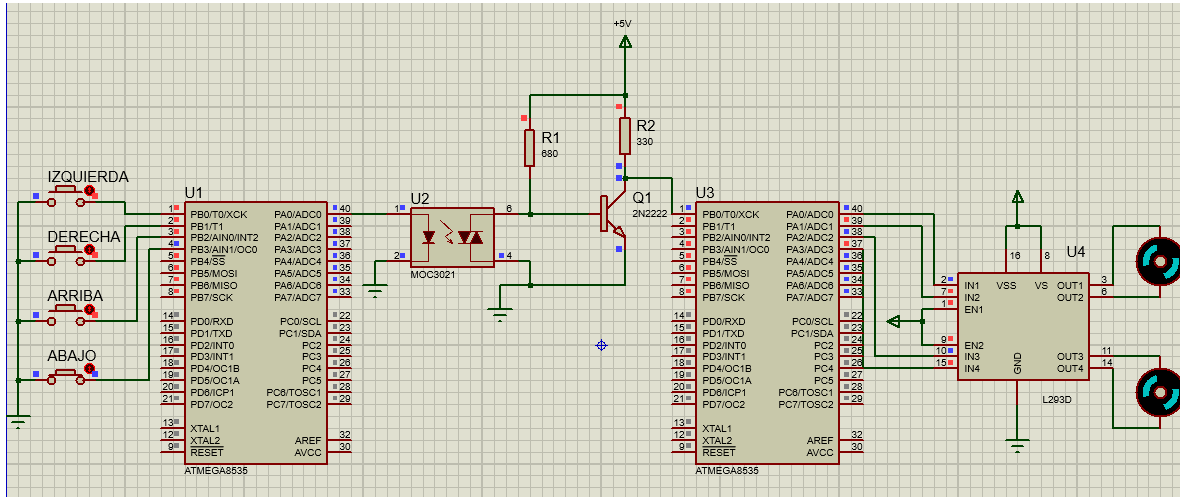


Esta práctica nos ayudara a entender cómo se pueden implementar los circuitos con dos microcontroladores y todo el proceso que conlleva realizar la programación del microcontrolador.

Con esta introducción se podrá comprender y realizar el desarrollo de la práctica que se desarrolla más adelante.

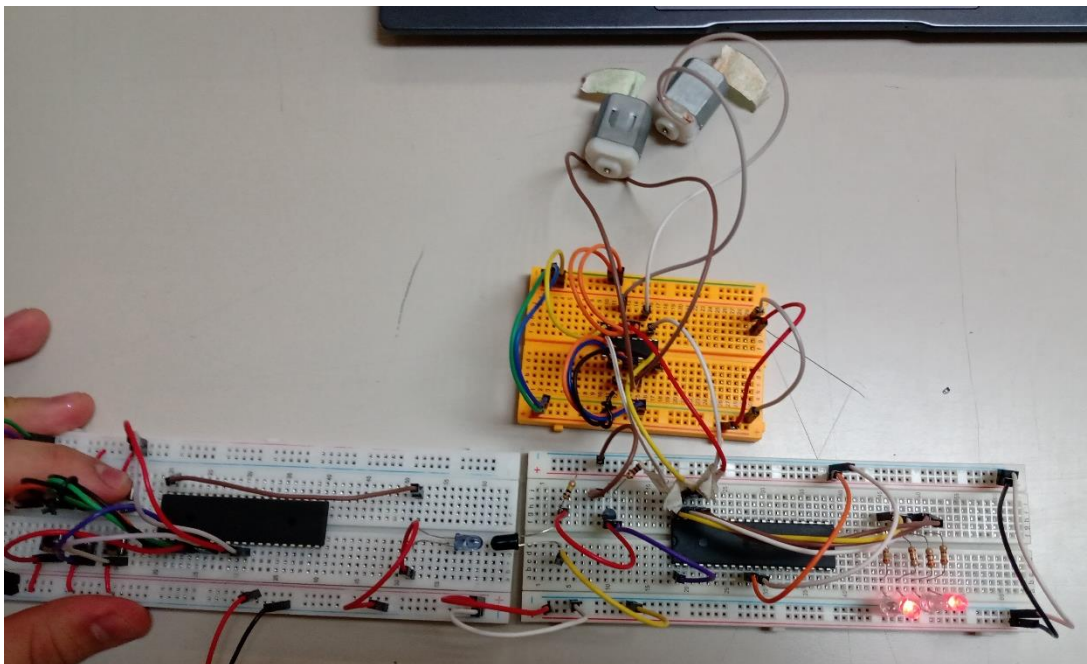
### III. DESARROLLO EXPERIMENTAL

1. Diseñe un móvil controlado vía infrarrojo, solo controle movimiento hacia atrás y hacia adelante como lo indican las flechas.



*Fig.1\_Circuito Armado y Simulado en Proteus.*

### CIRCUITO EN FISICO ARMADO.



*Fig.2\_Circuito Armado en Físico (Proyecto).*

## IV. CÓDIGO

### CÓDIGO DEL PRIMER ATMEGA8535.

```
#include <mega8535.h>
#include <delay.h>
#define IZQ PINB.0
#define DER PINB.1
#define ARRIBA PINB.2
#define ABAJO PINB.3

// Declare your global variables here

void main(void)
{
    // Declare your local variables here

    // Input/Output Ports initialization
    // Port A initialization
    // Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
    DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) | (1<<DDA1) |
    (1<<DDA0);
    // State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
    PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) |
    (0<<PORTA1) | (0<<PORTA0);

    // Port B initialization
    // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
    DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) |
    (0<<DDB0);
    // State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
    PORTB=(1<<PORTB7) | (1<<PORTB6) | (1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3) | (1<<PORTB2) |
    (1<<PORTB1) | (1<<PORTB0);

    // Port C initialization
    // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
    DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) |
    (0<<DDC0);
    // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
    PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) |
    (0<<PORTC1) | (0<<PORTC0);

    // Port D initialization
    // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
    DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<DDD1) |
    (0<<DDD0);
    // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
    PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) |
    (0<<PORTD1) | (0<<PORTD0);

    // Timer/Counter 0 initialization
    // Clock source: System Clock
    // Clock value: Timer 0 Stopped
```

```

// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) |
(0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);

// USART initialization
// USART disabled

```

```

UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8)
| (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) |
(0<<ACIS0);
SFIOR=(0<<ACME);

// ADC initialization
// ADC disabled
ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) | (0<<ADPS1)
| (0<<ADPS0);

// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) |
(0<<SPR0);

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

while (1)
{
    if(IZQ == 0) {
        PORTA = 0xff;
        delay_ms(25);
        PORTA = 0x00;
        delay_ms(475);
    } else if(DER == 0) {
        PORTA = 0xff;
        delay_ms(50);
        PORTA = 0x00;
        delay_ms(450);
    } else if(ARRIBA == 0) {
        PORTA = 0xff;
        delay_ms(75);
        PORTA = 0x00;
        delay_ms(425);
    } else if(ABAJO == 0) {
        PORTA = 0xff;
        delay_ms(100);
        PORTA = 0x00;
        delay_ms(400);
    }
}
}

```

## CÓDIGO DEL SEGUNDO ATMEGA8535.

```
#include <mega8535.h>
#include <delay.h>
#define ARRIBA 0x5
#define ABAJO 0xA
#define IZQ 0x2
#define DER 0x1
#define ENTRADA PINB.0
int contador = 0, i = 0;

// Declare your global variables here

void main(void)
{
    // Declare your local variables here

    // Input/Output Ports initialization
    // Port A initialization
    // Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
    DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) | (1<<DDA1) |
    (1<<DDA0);
    // State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
    PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) |
    (0<<PORTA1) | (0<<PORTA0);

    // Port B initialization
    // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
    DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) |
    (0<<DDB0);
    // State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
    PORTB=(1<<PORTB7) | (1<<PORTB6) | (1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3) | (1<<PORTB2) |
    (1<<PORTB1) | (1<<PORTB0);

    // Port C initialization
    // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
    DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) |
    (0<<DDC0);
    // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
    PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) |
    (0<<PORTC1) | (0<<PORTC0);

    // Port D initialization
    // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
    DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<DDD1) |
    (0<<DDD0);
    // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
    PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) |
    (0<<PORTD1) | (0<<PORTD0);

    // Timer/Counter 0 initialization
    // Clock source: System Clock
    // Clock value: Timer 0 Stopped
    // Mode: Normal top=0xFF
```



```

// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) |
(0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);

// USART initialization
// USART disabled

```

```

UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8)
| (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) |
(0<<ACIS0);
SFIOR=(0<<ACME);

// ADC initialization
// ADC disabled
ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) | (0<<ADPS1)
| (0<<ADPS0);

// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) |
(0<<SPR0);

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

while (1)
{
    contador = 0;

    for (i = 0; i < 500; i++) {
        if (ENTRADA == 1) contador++;
        delay_ms(1);
    }

    if(contador >= 15 && contador <= 35) {
        PORTA = IZQ;
    } else if(contador >= 40 && contador <= 60) {
        PORTA = DER;
    } else if(contador >= 65 && contador <= 85) {
        PORTA = ARRIBA;
    } else if(contador >= 90 && contador <= 110) {
        PORTA = ABAJO;
    } else {
        PORTA = 0x00;
    }
}
}

```

## **V. OBSERVACIONES Y CONCLUSIONES**

Gracias a la realización de la práctica podemos observar cómo se utiliza e implementa un circuito con dos microcontroladores (ATMega8535) y como gracias a su programador pudimos logra programar el integrado con nuestro código en lenguaje C. En esta práctica al momento de armar los circuitos y simularlos pudimos observar cómo se implementan los dispositivos con un control infrarrojo (como lo son los carritos de control remoto), que realmente fue un poco difícil saber si al inicio funcionaba o no el circuito (hasta que se implementaron unos leds para saberlo).

Tras la investigación y explicación de la práctica previa me quedo un poco más claro como poder implementar estos circuitos y programar el ATMega8535. También al observar como funcionan los moteres y el puente H, aunque en mi caso funcionaban bien solos, pero, al conectarlo con los ATMega les faltaba potencia e iban lentos.

Para concluir esta práctica podemos decir que fue una experiencia interesante ya que no había realizado un circuito con un ATMega que envíe las señales y otro que las reciba e interprete y ver si en realidad se estaba ejecutando correctamente. Gracias a las anteriores practicas fue más fácil realizar el proyecto.

## **VI. BIBLIOGRAFÍA**

- ✓ CodeVisionAVR. (2018). CodeVisionAVR. 20/Septiembre/2022, de CodeVisionAVR Sitio web: <http://www.hpinfo.tech.ro/cvavr-download.html>
- ✓ Sony. (2018). Probar si su control remoto envía una señal infrarroja (IR). 18/Noviembre/2022, de Sony, Sitio web: <https://www.sony-latin.com/es/electronics/support/articles/00025283>

