INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PROYECTO 1

ALUMNO: MALAGON BAEZA ALAN ADRIAN
MARTINEZ CHAVEZ JORGE ALEXIS

GRUPO: 6CM3

U.A: SISTEMAS EN CHIP

PROFESOR: FERNANDO AGUILAR SÁNCHEZ

FECHA DE ENTREGA: 30 DE ABRIL DE 2023

OBJETIVO

Al término de la sesión, el alumno contará con la habilidad de desarrollar un robot capaz de moverse hacia adelante, atrás, izquierda y derecha, mediante un mando a distancia empleando sensores infrarrojos.

INTRODUCCIÓN TEÓRICA

SEÑAL ANALÓGICA

La señal analógica es aquella que presenta una variación continua con el tiempo, es decir, que a una variación suficientemente significativa del tiempo le corresponderá una variación igualmente significativa del valor de la señal (la señal es continua). Toda señal variable en el tiempo, por complicada que ésta sea, se representa en el ámbito de sus valores (espectro) de frecuencia. De este modo, cualquier señal es susceptible de ser representada descompuesta en su frecuencia fundamental y sus armónicos.[1]

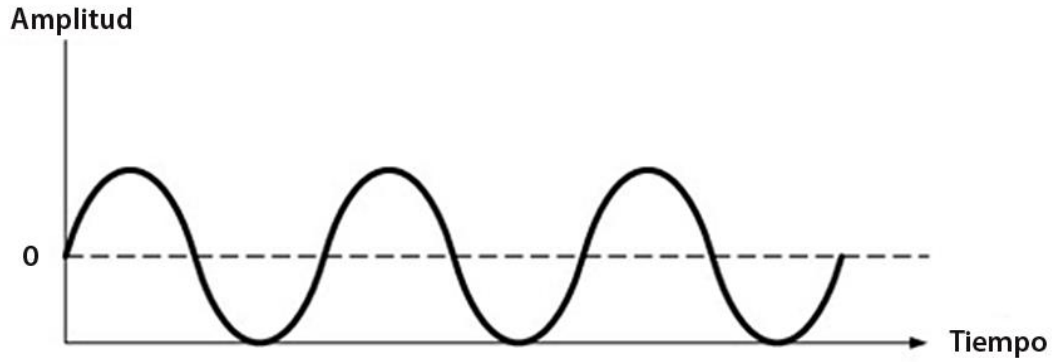


Figura 1. Representación de una señal analógica

El proceso matemático que permite esta descomposición se denomina análisis de Fourier. Un ejemplo de señal analógica es la generada por un usuario en el micrófono de su teléfono y que después de sucesivos procesos, es recibida por otro abonado en el altavoz del suyo. Es preciso indicar que la señal analógica, es un sistema de comunicaciones de las mismas características, mantiene dicho carácter y deberá ser reflejo de la generada por el usuario. Esta necesaria circunstancia obliga a la utilización de canales lineales, es decir canales de comunicación que no introduzcan deformación en la señal original.[1]

Las señales analógicas predominan en nuestro entorno (variaciones de temperatura, presión, velocidad, distancia, sonido etc.) y son transformadas en señales eléctricas, mediante el adecuado transductor, para su tratamiento electrónico. La utilización de señales analógicas en comunicaciones todavía se mantiene en la transmisión de radio y televisión tanto privada como comercial.[1]

Los parámetros que definen un canal de comunicaciones analógicas son el ancho de banda (diferencia entre la máxima y la mínima frecuencia a transmitir) y su potencia media y de cresta.[1]

CONTROL DE MOTOR DC CON PUENTE H

Este Control de motor DC con puente H se utiliza para controlar el sentido de giro de un motor de corriente continua (motor CC) y lleva este nombre, por la letra "H" que forman el arreglo de los transistores en el circuito. El circuito sirve para controlar motores de mediana potencia, debido a la limitación de potencia de los transistores.[2]

El sentido de giro del motor DC depende de los niveles de voltaje que existan en los puntos del circuito etiquetados como: "Avance" y "Retroceso". Solo uno de estos dos puntos puede estar a nivel alto para activar los transistores correspondientes. En la siguiente figura se muestra el circuito para armar el puente H.[2]

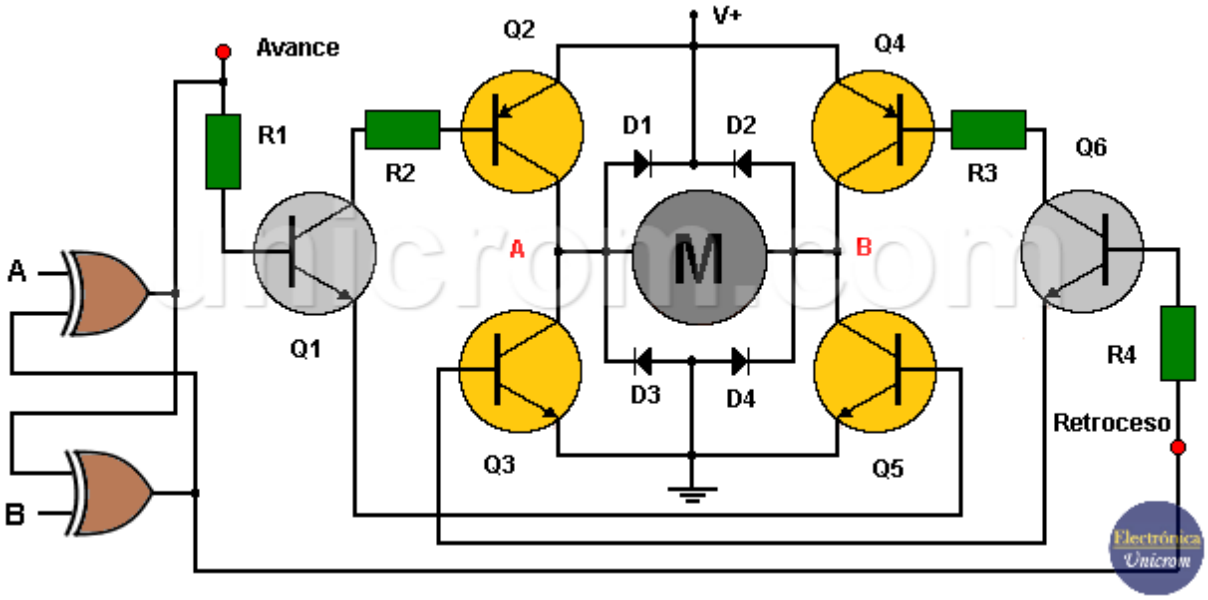


Figura 2. Circuito del puente H para motor DC

- Si el nivel de voltaje en la etiqueta "Avance" está en nivel alto se satura el transistor Q1 que a su vez hace entrar en saturación los transistores Q2 y Q5. Estos dos transistores permiten a circulación de corriente por el motor DC en un sentido.[2]
- Si el nivel de voltaje en la etiqueta "Retroceso" está en nivel alto se satura el transistor Q6 que a su vez hace entrar en saturación los transistores Q3 y Q4. Estos dos transistores permiten a circulación de corriente por el motor DC en el sentido contrario.[2]

Los diodos se colocan para la protección de los transistores, debido al cambio de polaridad en las bobinas del motor DC.[2]

MATERIALES Y EQUIPO EMPLEADO

- ✓ CodeVision AVR
- ✓ AVR Studio 4
- ✓ 2 Microcontroladora ATmega 8535
- ✓ 4 Push Button
- ✓ Módulo Sensor de Obstáculos TR5000
- ✓ Puente H L293D

DESARROLLO EXPERIMENTAL

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

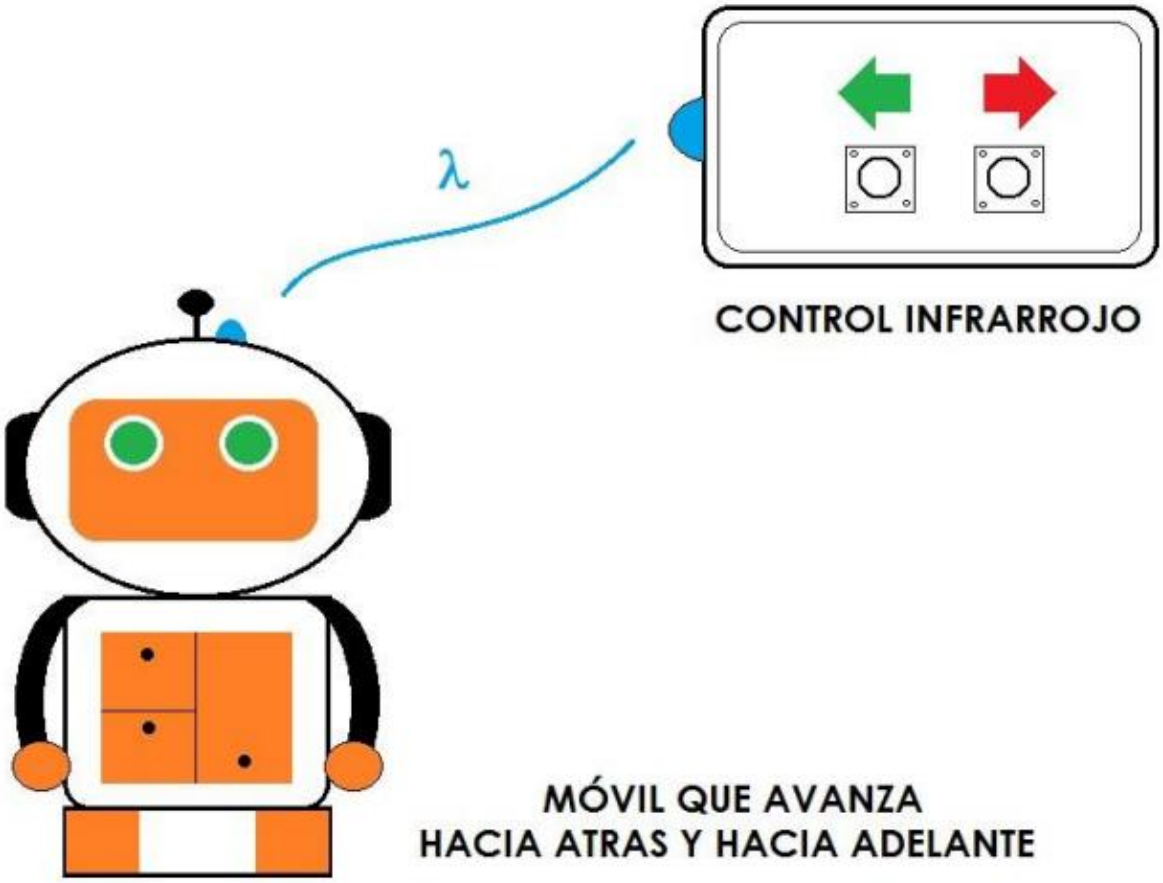


Figura 3. Esquema del proyecto

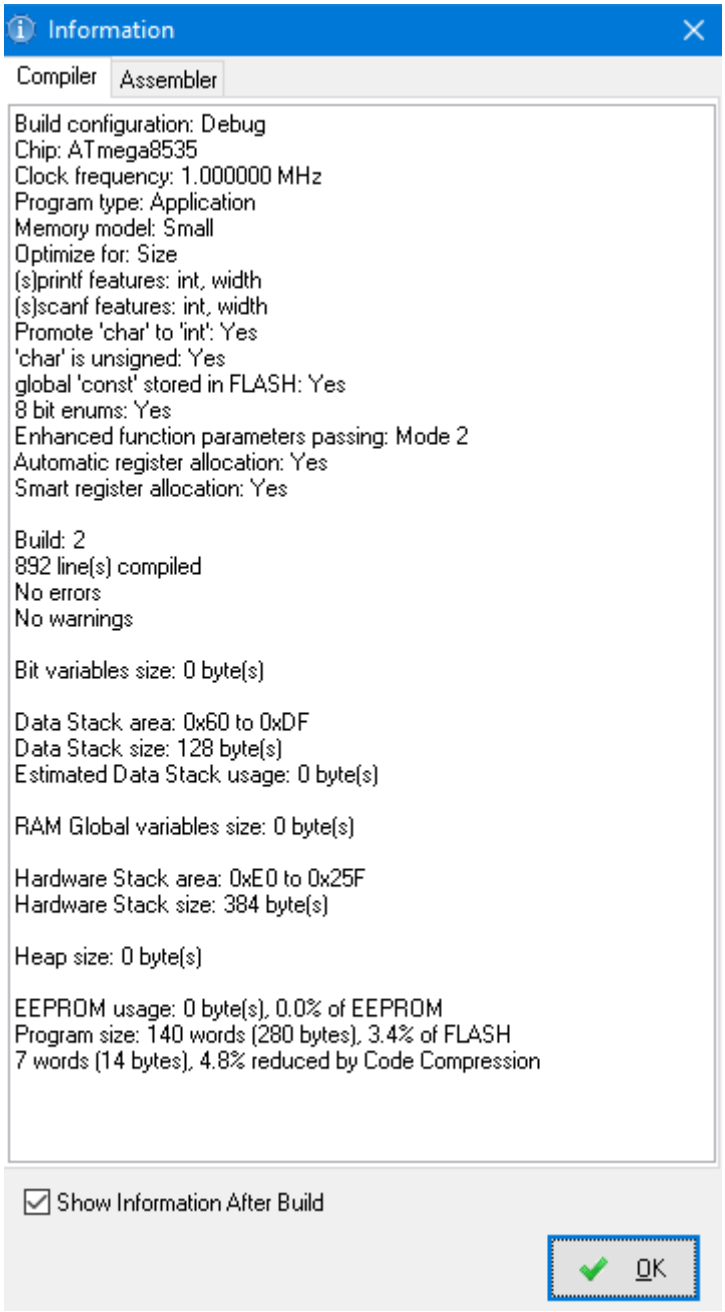


Figura 4. Compilación exitosa en CodeVision del código del EMISOR

Figura 5. Compilación exitosa en CodeVision del código del RECEPTOR

CÓDIGO GENERADO POR CODEVISION

- **Código para el EMISOR**

```
/*****************************************************
This program was created by the CodeWizardAVR V3.47
Automatic Program Generator
© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    :
Author  :
Company :
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

#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 para el RECEPTOR**

```
/*****************************************************
This program was created by the CodeWizardAVR V3.47
Automatic Program Generator
© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro
```

```
Project :
Version :
Date    :
Author  :
Company :
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

#include <mega8535.h>
#include <delay.h>
#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.0 = 0;
            PORTA.1 = 1;
            PORTA.2 = 0;
            PORTA.3 = 0;
        }
        else if(contador >= 40 && contador <= 60) {
            PORTA.0 = 1;
            PORTA.1 = 0;
            PORTA.2 = 0;
            PORTA.3 = 0;
        }
        else if(contador >= 65 && contador <= 85) {
            PORTA.0 = 0;
            PORTA.1 = 0;
            PORTA.2 = 1;
            PORTA.3 = 0;
        }
        else if(contador >= 90 && contador <= 110) {
            PORTA.0 = 0;
            PORTA.1 = 0;
            PORTA.2 = 0;
            PORTA.3 = 1;
        }
        else{
            PORTA = 0x00;
        }
      }
}
```

CIRCUITO ELECTRICO EN PROTEUS

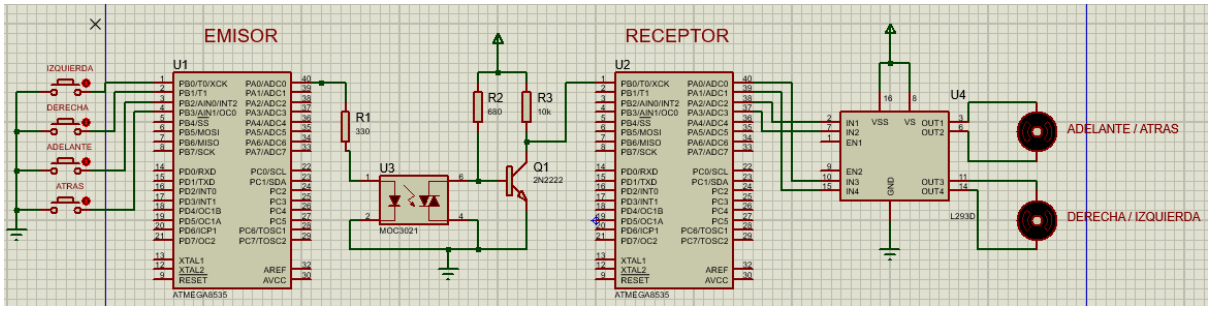
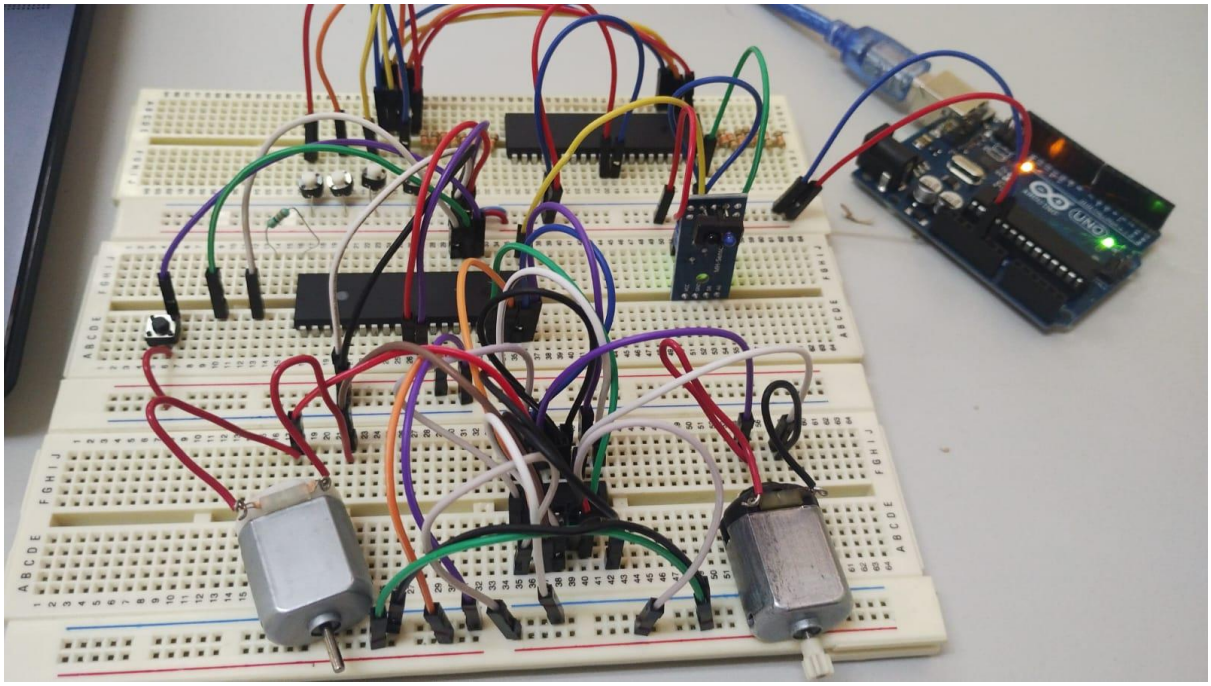


Figura 6. Circuito simulado en Proteus

CIRCUITO EN EL PROTO ARMADO



OBSERVACIONES Y CONCLUSIONES INDIVIDUALES

Malagón Baeza Alan Adrián

El desarrollo de este proyecto fue muy bueno para poner en práctica todos los conocimientos desarrollados en las prácticas previas. El uso del par infrarrojo se hizo de igual manera que en la práctica anterior con el empleo de un MOC3021.

La lógica para desarrollar la práctica no fue tan complicada, sin embargo, creo que el mayor reto que presenté fue el de interpretar la señal recibida en el microcontrolador que se usó para la parte del receptor de las señales. Emparejar las señales emitidas por el emisor e interpretarlas mediante condicionales y ciclos del lado del receptor hizo que esta práctica me llevara más tiempo de desarrollo que las anteriores.

Martínez Chávez Jorge Alexis

Creo que este proyecto es muy interesante y tiene varias aplicaciones que podrían servir para el desarrollo de proyectos más grandes. La transmisión de señales y ondas a través de un par infrarrojo nos puede ayudar para desarrollar proyectos con mandos a distancia que requieran la interpretación de distintos comandos para llevar a cabo distintas tareas.

El principal conocimiento que me llevo con el desarrollo de este proyecto es el manejo de distintas señales para la comunicación entre dos microcontroladores. Además de reforzar los conocimientos para el desarrollo y la construcción de un puente H para controlar el sentido de giro de un motor DC.

BIBLIOGRAFÍA

[1] EcuRed, “Señales analógicas y digitales - EcuRed,” *Ecured.cu*, 2019. https://www.ecured.cu/Se%C3%B1ales_anal%C3%B3gicas_y_digitales (accessed Mar. 21, 2022).
[2] Administrador, “Control de motor DC con puente H,” *Electrónica Unicrom*, Aug. 25, 2020. https://unicrom.com/control-de-motor-dc-con-puente-h/ (accessed Mar. 21, 2022).