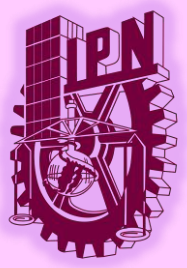

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Introducción a los microcontroladores

3CM16

Proyecto 01

Equipo 7

Integrantes:

- **Bocanegra Heziquio Yestlanezi**
- **Dominguez Durán Alan Axel**
- **Hernandez Mendez Oliver Manuel**
- **Martinez Cruz José Antonio**

Profesor: Aguilar Sanchez Fernando

Índice

Introducción

Transmisor

es un dispositivo electrónico que se utiliza para enviar señales o información de un punto a otro. Su función principal es convertir la información en una forma adecuada para su transmisión a través de un medio de comunicación [1].

Conexión con ATMega8535

Conexión de componentes: Para construir un transmisor utilizando el ATMega8535, necesitarás conectar varios componentes adicionales, como un oscilador para generar la frecuencia de transmisión, un amplificador de potencia para aumentar la señal, un modulador para modular la información [2].

Programación del microcontrolador: El ATMega8535 se programa utilizando un lenguaje de programación de bajo nivel como C o ensamblador. Deberás escribir un código que controle el funcionamiento del transmisor, incluyendo la generación de la señal de transmisión, la modulación de la información y la interacción con otros componentes [2].

Protocolo de comunicación: Determina el protocolo de comunicación que utilizarás para transmitir la información.

Receptor

son dispositivos diseñados para captar y procesar señales electromagnéticas con el fin de convertirlas en información útil [3].

Puente H

es un circuito electrónico utilizado para controlar la dirección y la velocidad de un motor DC (corriente continua). Se llama puente H debido a su configuración en forma de una letra "H". Este circuito es ampliamente utilizado en aplicaciones como robótica, control de motores, vehículos eléctricos y sistemas de tracción [4].

consta de cuatro interruptores controlados electrónicamente, generalmente transistores o relevadores, dispuestos de manera que formen dos caminos de corriente entre la fuente de alimentación y el motor. Estos caminos permiten que la corriente fluya en ambas direcciones a través del motor, lo que permite controlar su giro en ambas direcciones [4].

El funcionamiento del puente H se basa en la conmutación adecuada de los interruptores. Dependiendo de la combinación de estados de los interruptores, se pueden obtener cuatro posibles configuraciones: avance, retroceso, frenado y apagado. En la configuración de avance, se permitirá que la corriente fluya en una dirección a través del motor, lo que lo hará girar en una dirección específica. En la configuración de retroceso, se invierte la dirección de la corriente y, por lo tanto, el sentido de giro del motor [4].

El puente H también ofrece la capacidad de frenado, donde los interruptores se abren y se cierran rápidamente, lo que genera una corriente inversa en el motor y ayuda a detenerlo rápidamente. En la configuración de apagado, todos los interruptores están abiertos, lo que detiene el flujo de corriente hacia el motor [4].

Es importante destacar que el puente H debe ser controlado adecuadamente para evitar la activación simultánea de los interruptores, lo que podría generar un cortocircuito y dañar el circuito y el motor. Para ello, se utilizan técnicas de control como la modulación por ancho de pulso (PWM) para regular la velocidad y la dirección del motor de manera segura y eficiente [4].

Desarrollo

Utilizando los conocimientos adquiridos mediante la asignatura de introducción a los microcontroladores desarrollaremos el armado de un circuito utilizando dos microcontroladores ATMega5835 que desarrollaran la función de un receptor y un transmisor, además de esto le colocaremos un puente H, a continuación, en la imagen 1 se puede apreciar el circuito armado en protoboard funcionado.

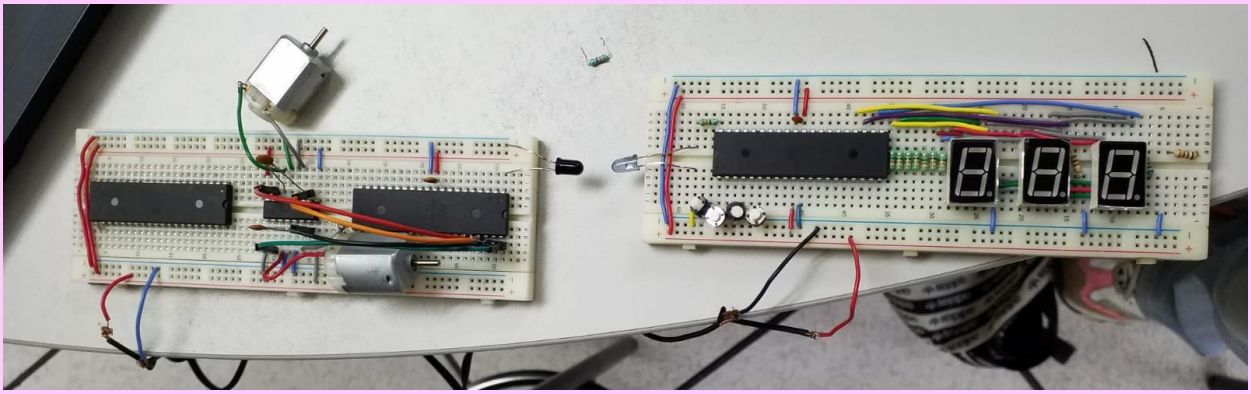

Imagen 1 circuito armado en protoboard

Transmisor

Como podemos observar en la imagen 1 tenemos uno de los microcontroladores ATMega8535 conectado a 4 botones los cuales nos permiten hacer los movimientos adelante, atrás, derecha e izquierda.

Un poco mas de cerca podemos observar la conexión en la imagen 2.

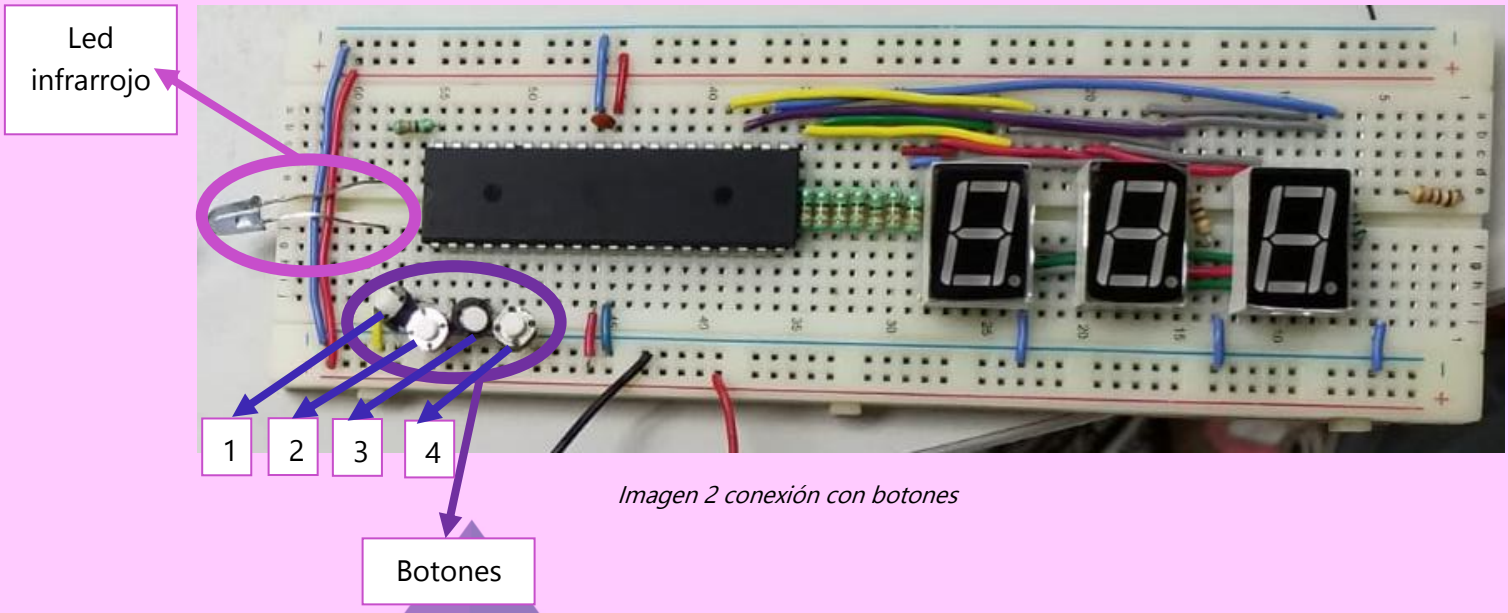

Imagen 2 conexión con botones

Así como también podemos observar del lado izquierdo el led infrarrojo el cual generara una serie de pulsos conforme se opriman los botones enumerados el 1 a 4.

Si oprimimos el botón 1 se generará una frecuencia "cuadrada" 1 aplicando un delay después generamos una salida 0 en la terminal y nuevamente colocamos un delay, esto nos formará una señal cuadrada, así por cada botón que oprimimos se generará una frecuencia diferente, la señal se puede modificar cambiando el valor del delay y así podemos obtener una señal diferente de cada frecuencia. Las señales deben de ser distintas para poder diferenciar el botón oprimido.

Botones

Botón izquierda

```
      if(IZQ == 0) {

            PORTA = 0xff;
            delay_ms(25);
            PORTA = 0x00;
            delay_ms(475);
```

Botón derecha

```
else if(DER == 0) {
             PORTA = 0xff;
             delay_ms(50);
             PORTA = 0x00;
             delay_ms(450);
```

Botón arriba

```
else if(ARRIBA == 0) {
             PORTA = 0xff;
             delay_ms(75);
             PORTA = 0x00;
             delay_ms(425);
```

Botón abajo

```
else if(ABAJO == 0) {
             PORTA = 0xff;
             delay_ms(100);
             PORTA = 0x00;
             delay_ms(400);
```

en caso de no oprimir ningún botón, no se genera ninguna frecuencia y no pasa al led.

Receptor

La señal viaja a través del aire, por lo cual con un infrarrojo transmitimos la información. El receptor y transmisor deben de ser aislados, por lo cual los coloramos en dos protoboard como se muestra en la imagen 1.

Diseño

Nosotros utilizamos un while con un contador iniciado en 0 y después su ciclo for.

```
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;
        }
      }
```

El programa se encarga de cotar pulsos, el cual nos dará los pulsos que entran y la cuenta de la frecuencia que ingresa.

Los valores que obtenemos dependerán del valor del retardo que agregamos que en nuestro caso particular nuestro delay es de 1 ms para así lograr cuentas diferentes.

Puente H

A través de un puente H controlaremos los dos motores solicitados en el proyecto 1.

El puente H controla los dos motores que cambiara el sentido del movimiento para simular la parte del movimiento de los cuatro botones.

Cuando en el receptor se detecta la frecuencia 1, en la salida digital del puente H se genera la combinación correspondiente para generar el movimiento.

Si hay dos motores para que se muevan hacia delante deben de tener el mismo movimiento, en el mismo sentido.

Para mover los motores, se generará en las entradas del puente H.

Nosotros primero probamos que el puente H tuviera funcionamiento para después conectar lo demás, si conectamos los valores 11 o 00 el motor no se mueve, pero si son distintas el motor se mueve y si son distintas no lo hará, ocupamos el mismo funcionamiento para el segundo motor.

Simulación proteus

A continuación, en la imagen 3 podemos observar el armado en proteus del circuito para generar la simulación, con base en esta simulación funcionado obtuvimos el armado del circuito en el protoboard.

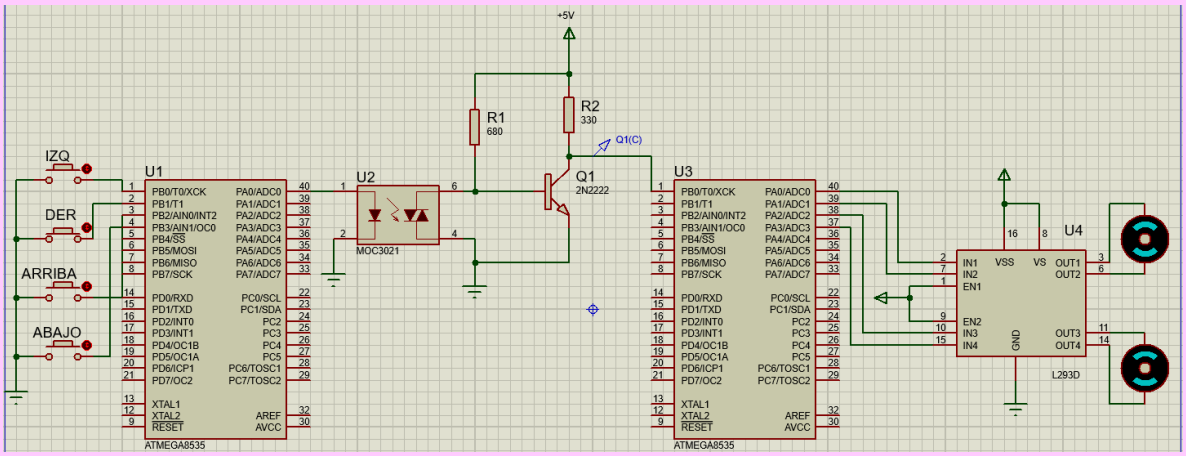

Imagen 3 simulación del circuito en proteus

Codificación

Transmisor

```
/*****************************************************
This program was created by the CodeWizardAVR V3.46a
Automatic Program Generator
© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 18/05/2023
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);
```

```
        }
      }
}
```

Receptor

```
/*****************************************************
This program was created by the CodeWizardAVR V3.46a
Automatic Program Generator
© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 18/05/2023
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 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;
        }
      }
}
```

Conclusiones

Bocanegra Heziquio Yestlanezi

Puedo concluir del primer proyecto, que fue difícil ya que lo armamos tres veces ya que no podíamos percatarnos del error que tenía, ya que no sabíamos si era le transmisor, el receptor, el puente H o los programas o el armado o los leds que podían no servir. Para la ultima vez que armamos decidimos verificar primero el puente H para estar seguros de que ese no era el problema, pero si era el principal problema, lo resolvimos y solucionando eso los motores comenzaron a tener movimiento.

Dominguez Durán Alan Axel

En este primer proyecto se implementaron módulos utilizados en prácticas previas, como fueron el bloqueador de rebotes de los push buttons y el módulo infrarrojo, en este caso se logró comunicar 2 micros a través del módulo infrarrojo. En primera instancia fue complicado comunicarlos ya que la frecuencia de los micros inflluye en la frecuencia de comunicación del emisor y receptor; pasado la parte de la intercomunicación de los micros, fue relativamente sencillo comunicar la salida del micro receptor al puente H para poder potenciar los motores. Sin duda una práctica desafiante pero educativa, ya que pudimos acoplar varios de los conocimientos anteriores a pesar de que había varios puntos donde podía fallar nuestro proyecto.

Hernandez Mendez Oliver Manuel

Considero que fue una proyeco interesante ya que permitió al equipo darse cuenta de la importancia de los transmisores y receptores infrarrojos, esto gracias a que brindan una forma inalámbrica de comunicación entre el microcontrolador y el robot, mientras que los puentes H permiten controlar los motores del robot para lograr el movimiento deseado. Estos componentes son elementos clave en el diseño y desarrollo de sistemas robóticos controlados por microcontroladores.

Martinez Cruz José Antonio

Como este proyecto logramos establecer una comunicación entre dos microprocesadores de manera satisfactoria, a pesar de que nos tomó tiempo lograr este objetivo. Como primera parte, el correcto funcionamiento de los componentes fue algo que nos retrasó en su implementación del circuito ya que nuestros diodos transmisor y receptor no trabajaban a pesar de su correcta instalación, sin embargo, solo basto con cambiar a otro diodo. Y en cuestión de programación fue un reto lograr la correcta sincronización de las ondas que permiten la conexión entre receptor y emisor. A pesar de estos inconvenientes, el objetivo fue alcanzado y se logró mostrar el funcionamiento de los motores.

Referencias

[1] Wireless Communications: Principles and Practice" por Theodore S. Rappaport.

[2] "The AVR Microcontroller and Embedded Systems: Using Assembly and C" por Muhammad Ali Mazidi, Sarmad Naimi y Sepehr Naimi.

[3] Principles of Electronic Communication Systems" de Louis E. Frenzel Jr 1938.