Instituto Politécnico Nacional

Escuela Superior de Cómputo

Departamento de Ciencias e Ingeniería de la Computación

Sistemas en Chip

Proyecto 02

"Bee-Bot"

Profesor: Fernando Aguilar Sánchez

Grupo: 6CM1

Equipo 3

Alumnos:

Ocampo Téllez Rodolfo

Patlani Mauricio Adriana

Ruvalcaba Flores Martha Catalina

Sandoval Hernández Eduardo

Fecha de entrega: 23/12/2022

Introducción Teórica

El microcontrolador ATMEGA8535 fabricante ATMEL. En la figura 1 se muestra el microcontrolador ATMEGA8535 la cual maneja datos de 8 bits es decir su bus de datos de 8 bits. Aunque este contiene tres registros los cuales son el *x*, *y* y *z*, los cuales manejan datos de 16 bits.

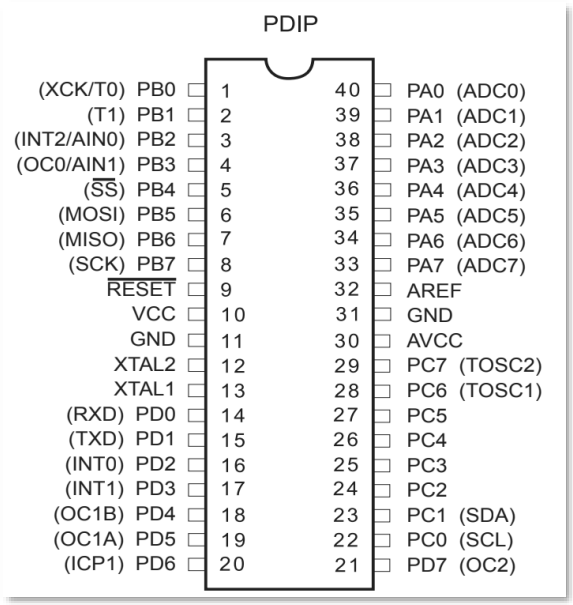

Figura 1. Configuración de pines ATmega8535

La comunicación interna del microcontrolador se categoriza en 4 puertos, en la figura 1 se puede analizar la etiquetación de los puertos PA0 al PA7, PB0 al PB7, PC0 al PC7 y PD0 al PD7. Cabe recordar que maneja datos de 8 bits.

Posee un oscilador interno de 1MHz, sin embargo, como es un oscilador RC, es susceptible a variar la frecuencia con respecto a las variaciones de temperatura. Por otro lado, puede conectarse un oscilador de 0 a 8 MHz o de 0 a 16 MHz.

Puente H

El puente H juega un papel muy importante en el área de la robótica y la automatización. Invertir el giro de un motor es posible gracias a dispositivos semiconductores y contactores eléctricos. El puente H es un mecanismo electrónico que se encarga de invertir el giro de un motor usando un elemento básico en electrónica como lo es el transistor.

Este versátil dispositivo tiene la habilidad de comportarse como un interruptor electrónico y como un amplificador. Cómo se hace necesario su uso para invertir el giro de un motor, solo para este caso se utilizará como un interruptor.

Su nombre proviene de su estructura de 4 interruptores en forma de letra H y un motor en el centro tal como se muestra en la figura 2.

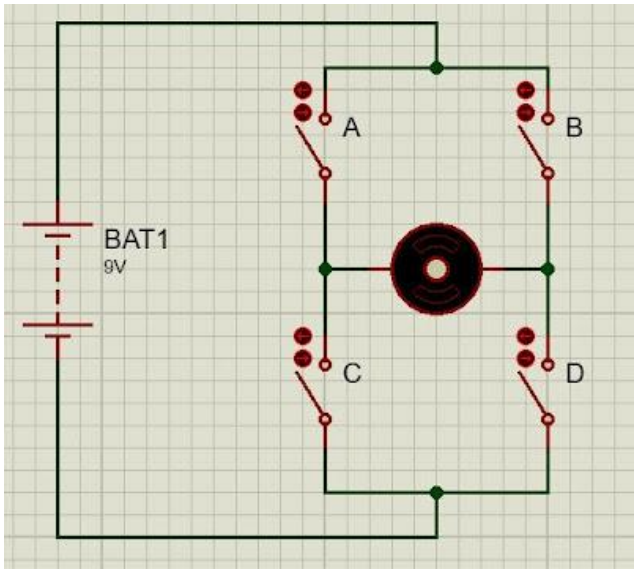

Figura 2. Esquema del Puente H.

Cuando se requiere que el motor que está ubicado en el centro de dicho circuito se mueva hacia la derecha es decir que tenga un giro horario normal, son los interruptores A y D los que se necesitan activar. Si asumimos que el motor tiene polaridad, de + a – y activamos los interruptores A y D la corriente circundante será de menos a más por la batería por lo cual circulará la corriente por el interruptor A, pasando por la polaridad positiva del motor, luego trasladándose hacia la polaridad negativa y por ende por el interruptor D habrá circulación de corriente por lo cual ese será el circuito o la malla circundante de corriente. al haber diferencia de potencial entre las terminales positiva y negativa habrá fuerza contra electromotriz por el motor de corriente directa DC.

Por el contrario, cuando se desea que el motor gire en sentido antihorario se deben activar los interruptores B y C. Desactivando los interruptores A y D y activando los interruptores B y C se puede observar que la corriente no circulará por el polo positivo del motor sino por el polo negativo del mencionado, por lo cual la corriente entrará por el interruptor B y saldrá por el interruptor C y de esta manera habrá fuerza electromotriz en sentido anti-horario.

A	B	C	D	MOTOR
OFF	OFF	OFF	OFF	APAGADO
ON	OFF	OFF	ON	GIRO HORARIO
OFF	ON	ON	OFF	GIRO ANTIHORARIO
ON	ON	ON	ON	FRENADO

Tabla 1. Estados de funcionamiento del puente H.

Bee-bot

Es un robot educativo diseñado para desarrollar las capacidades elementales de la programación, pensamiento computacional, concentración, ubicación espacial y estrategia. Consiste en una “abeja” robot que debe programarse para conseguir que efectúe unos movimientos determinados sobre una cuadrícula.

Los Bee-Bot aceptan instrucciones o comandos del tipo adelante, atrás, girar a la izquierda y girar a la derecha, que se programan mediante unas intuitivas teclas de dirección. Una vez programada la secuenciación elegida, simplemente hay que pulsar en la tecla GO para que el Bee-Bot ejecute el programa.

Desarrollo Experimental

1.- El objetivo de este proyecto es diseñe un móvil programable el cual será capaz de grabarle una secuencia y después reproducirla. Con el botón CLEAR se borrará la última secuencia y estará listo para ingresar la nueva con los botones de FLECHAS (ustedes determinen hasta cuantas secuencias podrán programas, ejemplo 10, 20, 30, etc.). Para iniciar la secuencia deberá oprimir el botones GO y para detenerse el botón PAUSE o terminar la secuencia previamente grabada.

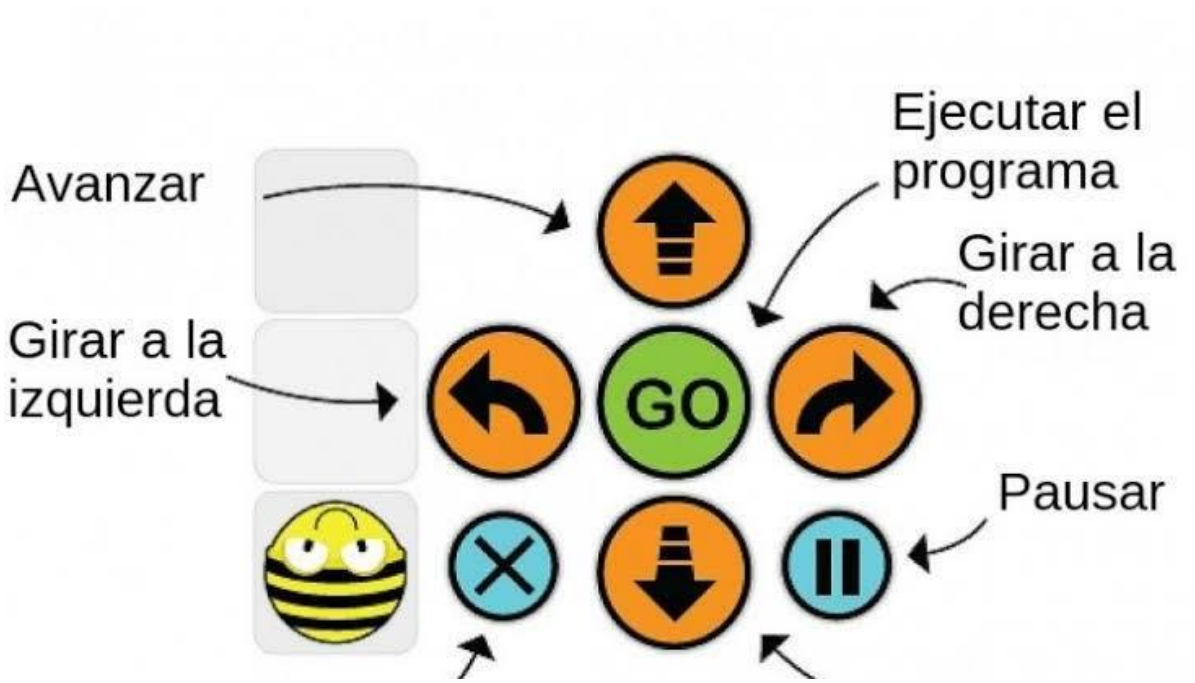

Figura 3. Esquema del proyecto.

Figura 4. Trayectorias que podrá recorrer el móvil.

Estructura del programa

Código de la configuración de los periféricos utilizados y código del programa principal en C, proporcionados por el IDE de CodeVision, correspondientes al **primer Atmega**.

 # #include <mega8535.h>

#include <delay.h>

#define Delante PINB.0

#define Reversa PINB.1

#define Izquierda PINB.2

```
#define Derecha   PINB.3

#define Pause     PINB.4

#define Delete    PINB.5

#define Go        PINB.6

#define Salida    PORTA.0

void main(void)

{

// Input/Output Ports initialization

// Port A initialization

DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) |
(1<<DDA1) | (1<<DDA0);

PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization

DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) |
(0<<DDB1) | (0<<DDB0);

PORTB=(1<<PORTB7) | (1<<PORTB6) | (1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3) |
(1<<PORTB2) | (1<<PORTB1) | (1<<PORTB0);

// Port C initialization

DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) |
(0<<DDC1) | (0<<DDC0);

PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization

DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);

PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) |
(0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

…

…

…
```

```
while (1)
    {
    if(Delante == 0){
        Salida = 1;
        delay_ms(25);
        Salida = 0;
        delay_ms(475);
    } else if(Reversa == 0){
        Salida = 1;
        delay_ms(50);
        Salida = 0;
        delay_ms(450);
    } else if(Izquierda == 0){
        Salida = 1;
        delay_ms(75);
        Salida = 0;
        delay_ms(425);
    } else if(Derecha == 0){
        Salida = 1;
        delay_ms(100);
        Salida = 0;
        delay_ms(400);
    } else if(Pause == 0){
        Salida = 1;
        delay_ms(125);
        Salida = 0;
        delay_ms(375);
    } else if(Delete == 0){
        Salida = 1;
```

```
        delay_ms(150);
        Salida = 0;
        delay_ms(350);
    } else if(Go == 0){
        Salida = 1;
        delay_ms(175);
        Salida = 0;
        delay_ms(325);
    }
    }
}
```

Código de la configuración de los periféricos utilizados y código del programa principal en C, proporcionados por el IDE de CodeVision, correspondientes al **segundo Atmega**.

```
// I/O Registers definitions
#include <mega8535.h>
#include <delay.h>
#define Delante 0x05
#define Reversa 0x0A
#define Izquierda 0x01
#define Derecha 0x04
#define Entrada PINB.0
int contador, i = 0, aux;
unsigned char secuencia[10];
int num_secuencias = 0;
void main(void)
{
// Input/Output Ports initialization
// Port A initialization
```

```
DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) |
(1<<DDA1) | (1<<DDA0);

PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization

DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) |
(0<<DDB1) | (0<<DDB0);

PORTB=(1<<PORTB7) | (1<<PORTB6) | (1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3) |
(1<<PORTB2) | (1<<PORTB1) | (1<<PORTB0);

// Port C initialization

DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) |
(0<<DDC1) | (0<<DDC0);

PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization

DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);

PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) |
(0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

…

…

…

while (1)
    {
    contador = 0;
    for(i = 0; i < 500; i++){
        if(Entrada == 1)
            contador++;
        delay_ms(1);
    }
    if(contador >= 15 && contador <= 35){
```

```
        if(num_secuencias <= 10){
            secuencia[num_secuencias] = Delante;
            num_secuencias++;
        }
    }else if(contador >= 40 && contador <= 60){
        if(num_secuencias <= 10){
            secuencia[num_secuencias] = Reversa;
            num_secuencias++;
        }
    }else if(contador >= 65 && contador <= 85){
        if(num_secuencias <= 10){
            secuencia[num_secuencias] = Izquierda;
            num_secuencias++;
        }
    }else if(contador >= 90 && contador <= 110){
        if(num_secuencias <= 10){
            secuencia[num_secuencias] = Derecha;
            num_secuencias++;
        }
    }else if(contador >= 115 && contador <= 135){
        PORTA = 0x00;
    }else if(contador >= 140 && contador <= 160){
        PORTA = 0x00;
        for(aux = 0; aux < num_secuencias; aux++)
            secuencia[aux] = 0x00;
        num_secuencias = 0;
    }else if(contador >= 165 && contador <= 185){
        if(num_secuencias > 0){
            PORTA = secuencia[num_secuencias-1];
```

```
            delay_ms(500);
            secuencia[num_secuencias-1] = 0x00;
            num_secuencias--;
        }
        PORTA = 0x00;
    }
    else {
        PORTA = 0x00;
    }
    }
}
```

Simulaciones

La simulación se realizó en el software Proteus Design Suite, previamente cargado el programa en formato ".hex" obtenido del software CodeVision AVR, el cual es un software para el microcontrolador Microchip AVR y sus contrapartes XMEGA.

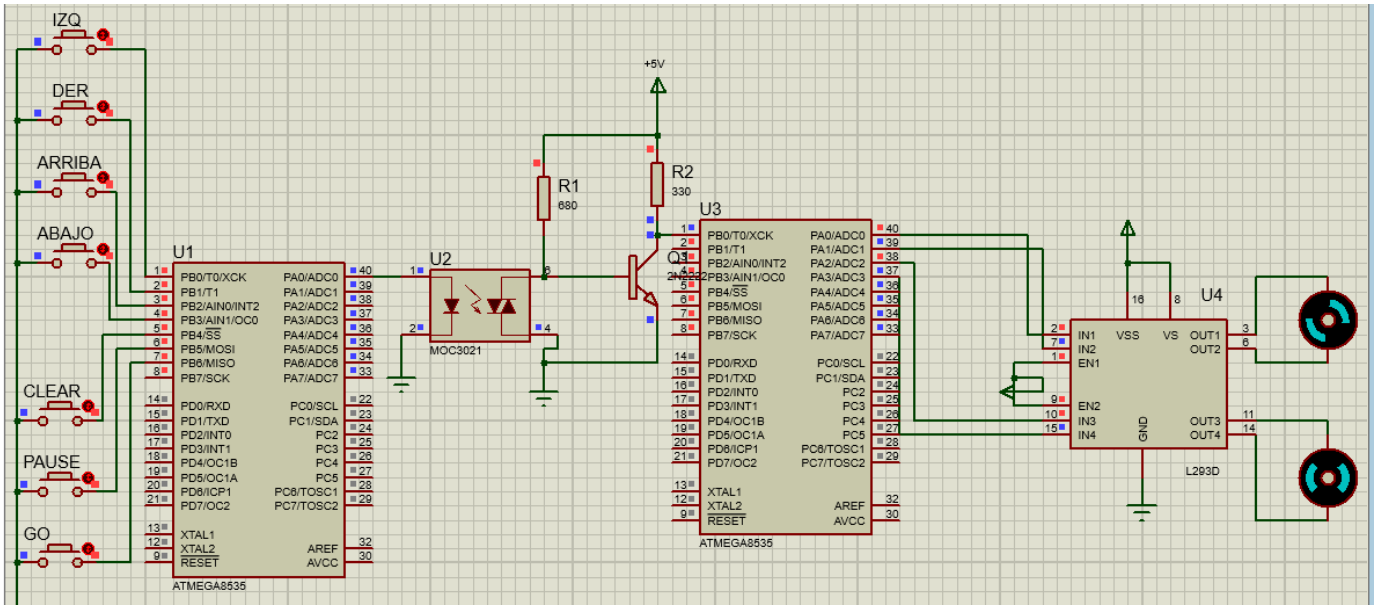

Figura 5. Simulación del proyecto 2.

Fotografía del circuito armado

A continuación, se muestra la figura 6 la evidencia sobre la realización y prueba del proyecto 2.

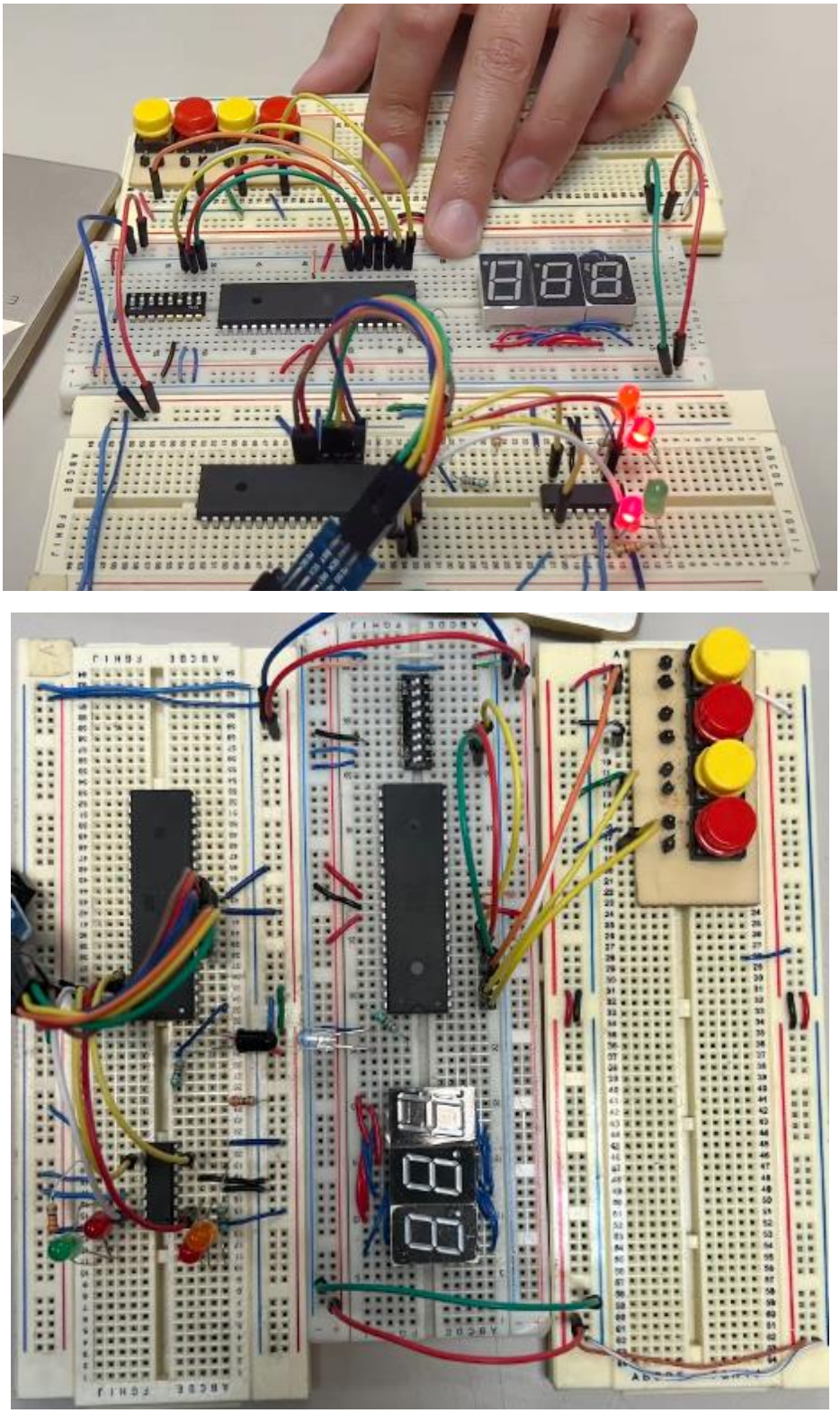

Figura 6. Circuito armado y funcionamiento del proyecto 2.

Observaciones y conclusiones Individuales

Ocampo Téllez Rodolfo

Este proyecto se pudo realizar utilizando como base el proyecto 1, en el cual ya se había diseñado un robot que pudiera seguir instrucciones: atrás, adelante, izquierda o derecha. Ahora, para emular un bee-bot, se tuvo que agregar los botones de play, pause and go, así como correspondiente funcionalidad de los Atmega para que las instrucciones se pudieran quedar guardadas mediante arrays utilizando la memoria del microcontrolador, y estos se reprodujeran hasta que el usuario presionara 'play', para los nuevos botones también se tuvieron que usar diferentes frecuencias de modo que se pudiera detectar cual era la funcionalidad que había seleccionado el usuario.

Patlani Mauricio Adriana

Este segundo proyecto parece ser una versión mejorada del proyecto 1, ya que se uso la base del primero y se implementaron los cocimientos adquiridos en las prácticas anteriores como la 11 al hacer uso de la memoria EEPROM el cual se guardaron una secuencia de instrucciones (un máximo de 10) y se implementaron 3 botones más (play, pause, go). Este proyecto guarda una serie de instrucciones en la memoria EEPROM y mediante los botones se inicia, se pausa o reanudan las instrucciones, por lo que en el código se usó una forma de implementar colas la cual es que la primera instrucción en entrar es la primera en salir (FIFO).

Ruvalcaba Flores Martha Catalina

El proyecto 2 requería que almacenara un conjunto de instrucciones en secuencia, tales instrucciones son las mismas que el proyecto 1, por lo que se usó como base para realizar este proyecto. Para solucionar el requisito de almacenar las instrucciones, se utilizaron los conocimientos de la memoria EEPROM, y fue más sencilla su realización, puesto que se utilizó la implementación de una cola, ya que es un FIFO, el primero en entrar es el primero en salir, así fue como las instrucciones se iban ejecutando. En este caso, ya no se utilizaron los motores y en lugar de estos se utilizaron cuatro leds para observar el comportamiento del robot, este robot pudo almacenar hasta 10 instrucciones y era ejecutado por sus respectivos botones de ejecutar, pausa y reiniciar.

Sandoval Hernández Eduardo

Continuando con el proyecto 1 y se le agregó la capacidad de almacenar una sucesión de instrucciones para que en este caso el "robot" se pudiera mover en las direcciones y secuencia indicada por la especificación de la práctica, lo cual fue sencillo de implementar teniendo el circuito previo y añadiendo los botones necesarios, así como las instrucciones necesarias al primer microcontrolador. Puede que aún se puede mejorar en algunos aspectos como el hecho de que el que almacena las instrucciones sea el primer microcontrolador y no el segundo, etc.

Bibliografía

• Atmel Corporation. (2006). 8-bit Microcontroller with 8K Bytes In-System Programmable Flash. [Online]. Disponible en: https://drive.google.com/file/d/1acpQaDlsyLHr3w3ReSgrlXRbshZ5Z-lb/view

• Sensoricx. (s.f). Puente H. [Online]. Disponible en: https://sensoricx.com/circuitos-para-armar/puente-h-funcionamiento-explicacion-detallada/

• Euskera. (s.f). Bee-Bot: robot infantil programable. [Online]. Disponible en: https://codigo21.educacion.navarra.es/autoaprendizaje/bee-bot-robot-infantil-programable/