INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PROYECTO 2: BEE-BOT

ALUMNO: MALAGON BAEZA ALAN ADRIAN
MARTINEZ CHAVEZ JORGE ALEXIS

GRUPO: 6CM3

U.A: SISTEMAS EN CHIP

PROFESOR: FERNANDO AGUILAR SÁNCHEZ

FECHA DE ENTREGA: 28 DE MAYO DE 2023

OBJETIVO

El objetivo de este proyecto es que diseñe un móvil programable el cual será capaz de grabarle una secuencia y después reproducirla.

INTRODUCCIÓN TEÓRICA

ROBOTS PROGRAMABLES

Los robots son máquinas que han sido programadas para implementar una serie de tareas independiente o semiautónomamente. La automatización implica el uso de software, máquinas u otra tecnología para implementar tareas. Generalmente, estas son manejadas manualmente por trabajadores humanos.[1]

De acuerdo con esto, el robot programable y la automatización tienen aplicaciones en varios campos. Algunas son el ejército, la agricultura, la fabricación, el comercio de divisas o la industria de la energía renovable. Las tareas para realizar en empresas de energía renovable han aumentado y se están convirtiendo en complejos desafíos de ingeniería. Por ello requieren la aplicación de robots y la automatización para ahorrar tiempo, aumentar la productividad y optimizar el rendimiento. [1]

Sin embargo, el término robot no está bien definido, al menos, no actualmente. Existe un gran debate en la comunidad científica, de ingeniería y de aficionados acerca de qué es exactamente un robot y qué no. [1]

Si tu visión de un robot es un dispositivo de apariencia algo humana que cumple órdenes de comando, entonces estás concibiendo esta tecnología como lo hace la mayoría de la gente. No es algo común ni práctico todavía, pero es un concepto que emerge de la literatura y del cine de ciencia ficción. [1]

FUNCIONAMIENTO DEL BEE-BOT

Beebot es un robot de la empresa inglesa TTS, que se desplaza 15 cm por paso programado y realiza giros de 90º. Además, permite programar paradas a modo de marcador. Toda su programación se realiza de manera manual presionando sobre unos sencillos botones donde marcar los movimientos a realizar. Permite programar hasta 40 pasos en una misma secuencia. Las actividades que podemos realizar con ellas van desde el trabajo de direccionalidad hasta el desarrollo de contenidos curriculares más complejos gracias a la aplicación de escenarios de aprendizaje.[2]

Para trabajar con ella, tras encenderla, deberemos tener en cuenta sólo tres aspectos:

- Cada vez que pulsemos un botón, los ojos de Bee-Bot parpadearán y oiremos un sonido que confirma la instrucción.[2]
- Bee-Bot siempre avanza o retrocede 15 cm y gira sobre sí misma 90º. La secuencia la realiza paso a paso, marcando cada acción con luz y sonido, animándonos a acompañarla y contar con ella. [2]
- Bee-Bot puede realizar hasta 40 movimientos. El botón de la X borra la memoria para empezar una nueva secuencia; en caso contrario repetirá la antigua secuencia y a continuación las nuevas instrucciones. [2]

MATERIALES Y EQUIPO EMPLEADO

- ✓ CodeVision AVR
- ✓ AVR Studio 4
- ✓ Microcontroladora ATmega 8535
- ✓ 7 Push Button
- ✓ Puente H L293D
- ✓ 2 motores DC

DESARROLLO EXPERIMENTAL

1. 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 1. Esquema del proyecto

2. El móvil tendrá que ser capaz de hacer el recorrido de las siguientes pistas.

Figura 2. Trayectorias que podrá recorrer el móvil

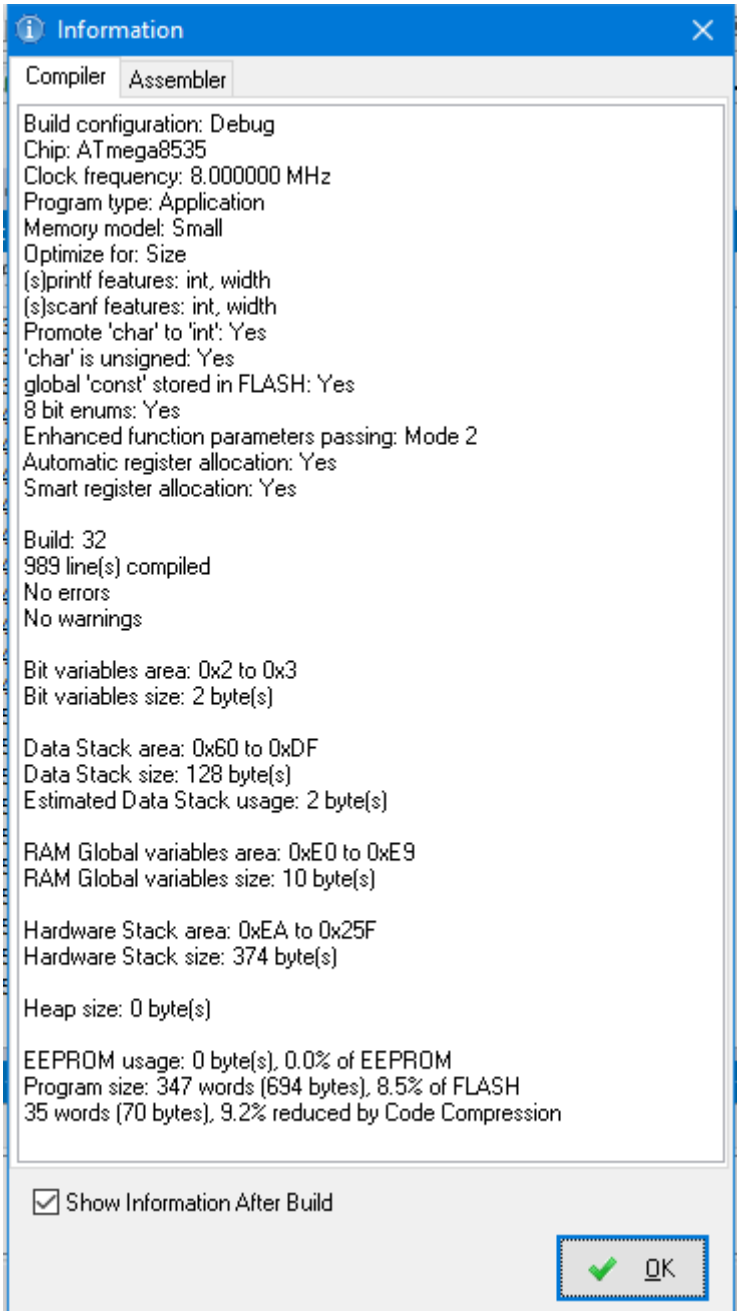


Figura 3. Compilación exitosa del código en CodeVisionAVR

CÓDIGO GENERADO POR CODEVISION

```
/*****************************************************
This program was created by the CodeWizardAVR V3.48b
Automatic Program Generator
© Copyright 1998-2022 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: 8.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

// I/O Registers definitions
#include <mega8535.h>
#include <delay.h>

//BOTONES DE ENTRADA
#define IZQ PINB.0
#define DER PINB.1
#define ARR PINB.2
#define ABA PINB.3
#define CLR PINB.4
#define PAU PINB.5
#define GO PINB.6

//CONTANTES PARA SALIDA
#define ARRIBA 0x5
#define ABAJO  0xA
#define IZQUIERDA 0x1
#define DERECHA 0x4

//VARIABLES
const unsigned int size = 10;
unsigned char read_index = 0, write_index = 0;
unsigned char inst[size];
bit a_izq, a_der, a_arr, a_aba, a_clr, a_pau, a_go, p_izq,
    p_der, p_arr, p_aba, p_clr, p_pau, p_go, mode = 0;
```

```
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){
    char current;
    char i;

    // BOTONES
    a_clr = CLR;
    a_pau = PAU;
    a_go = GO;

    if (p_clr == 1 && a_clr == 0) {
        for (i=0; i<size; i++) inst[i] = 0;
        write_index = 0;
        read_index = 0;
        mode = 0;
        delay_ms(20);
    }

    if (p_pau == 1 && a_pau == 0) {
        mode = 0;
```

```
        delay_ms(20);
    }

    if (p_go == 1 && a_go == 0) {
        mode = 1;
        delay_ms(20);
    }

    if (p_clr == 0 && a_clr == 1) delay_ms(20);
    if (p_pau == 0 && a_pau == 1) delay_ms(20);
    if (p_go == 0 && a_go == 1) delay_ms(20);

    //GRABADO Y LECTURA DE INSTRUCCIONES
    if (mode == 0 && write_index < size) {
        a_izq = IZQ;
        a_der = DER;
        a_arr = ARR;
        a_aba = ABA;

        if (p_izq == 1 && a_izq == 0) {
            inst[write_index] = 1;
            write_index++;
            delay_ms(20);
        }

        if (p_der == 1 && a_der == 0) {
            inst[write_index] = 2;
            write_index++;
            delay_ms(20);
        }

        if (p_arr == 1 && a_arr == 0) {
            inst[write_index] = 3;
            write_index++;
            delay_ms(20);
        }

        if (p_aba == 1 && a_aba == 0) {
            inst[write_index] = 4;
            write_index++;
            delay_ms(20);
        }

        if (p_izq == 0 && a_izq == 1) delay_ms(20);
        if (p_der == 0 && a_der == 1) delay_ms(20);
        if (p_arr == 0 && a_arr == 1) delay_ms(20);
        if (p_aba == 0 && a_aba == 1) delay_ms(20);
    }else if (mode == 1 && read_index < size) {
        current = inst[read_index];
```

```
        read_index++;

        switch(current){
            case 1:
                PORTA = DERECHA;
                delay_ms(50);
                PORTA = 0x00;
            break;

            case 2:
                PORTA = IZQUIERDA;
                delay_ms(50);
                PORTA = 0x00;
            break;

            case 3:
                PORTA = ARRIBA;
                delay_ms(50);
                PORTA = 0x00;
            break;

            case 4:
                PORTA = ABAJO;
                delay_ms(50);
                PORTA = 0x00;
            break;
        }
    }
    p_izq = a_izq;
    p_der = a_der;
    p_arr = a_arr;
    p_aba = a_aba;
    p_clr = a_clr;
    p_pau = a_pau;
    p_go = a_go;
}
}
```

CIRCUITO ELECTRICO EN PROTEUS

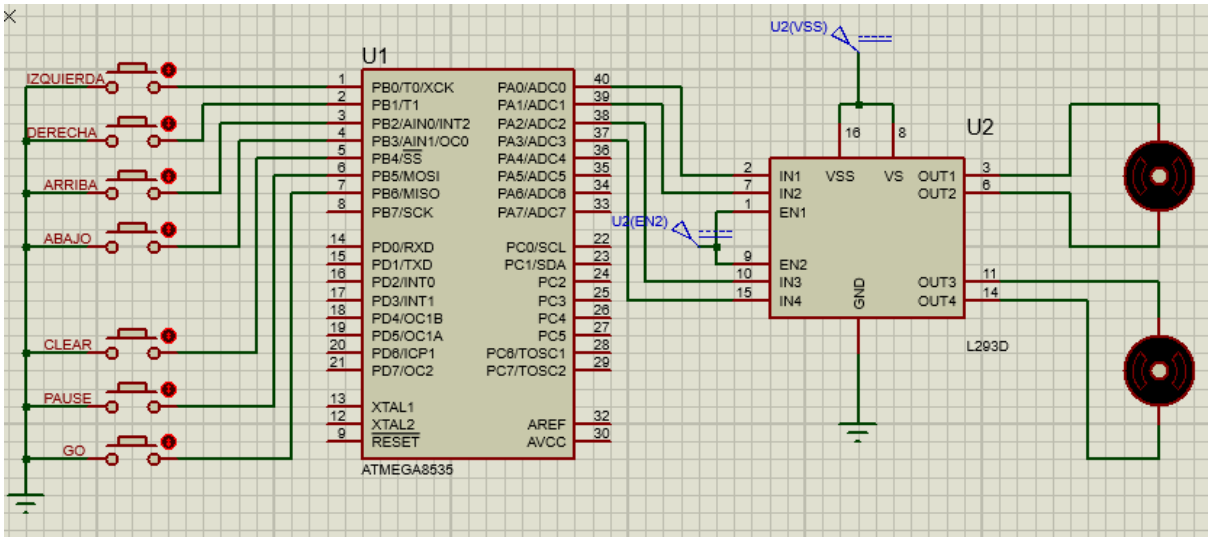
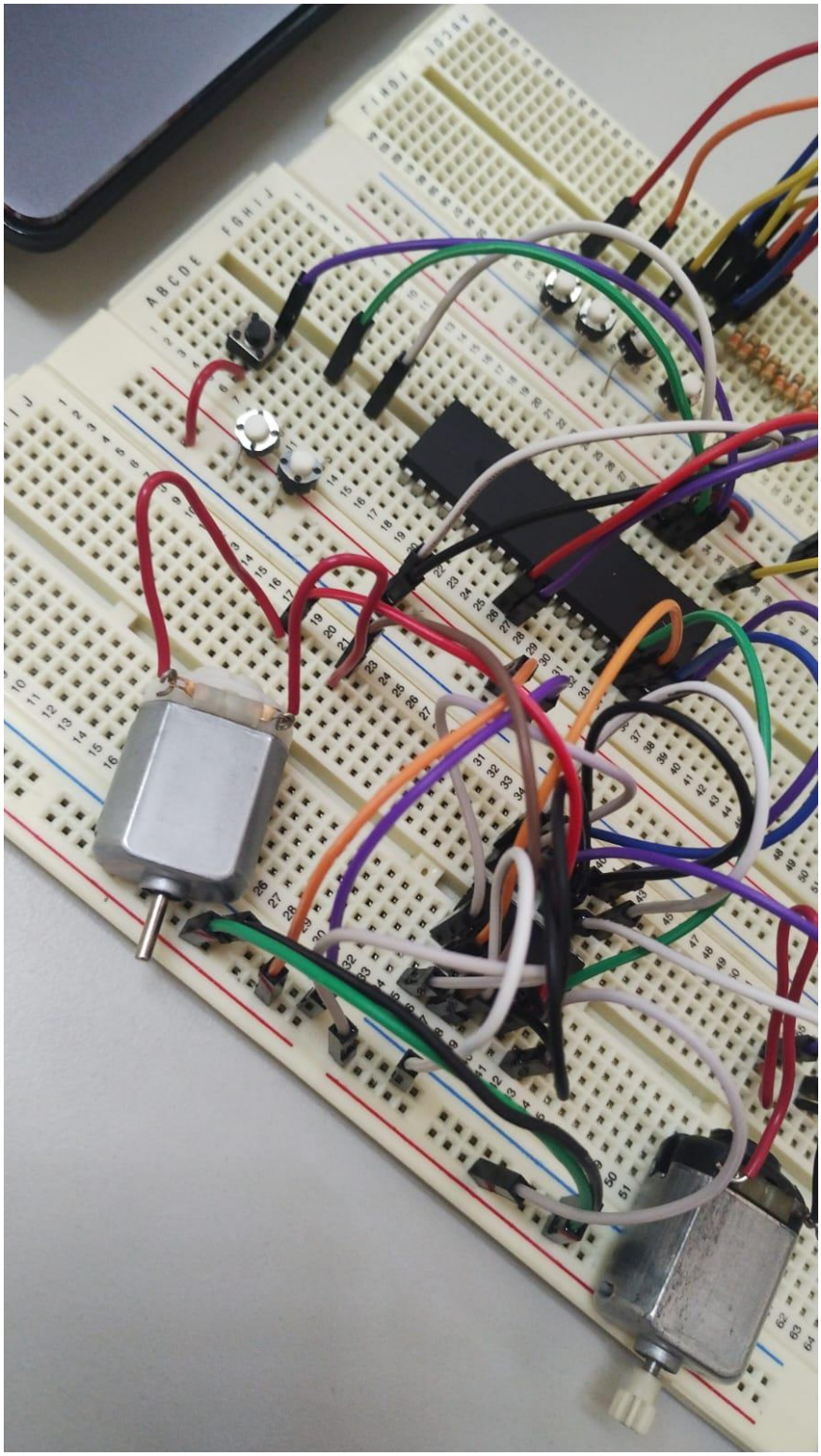


Figura 4. Circuito simulado en Proteus

CIRCUITO EN EL PROTO ARMADO



OBSERVACIONES Y CONCLUSIONES INDIVIDUALES

Malagón Baeza Alan Adrián

Este proyecto es muy interesante para desarrollar, puesto que se trata de un robot que se diseñó para ser un robot educativo con el fin de que los niños se interesen en la robótica y la programación.

La lógica de la programación del proyecto realmente no es tan compleja, sin embargo, me costó bastante tiempo el poder desarrollarla para que el proyecto funcionara al 100%. Para poder desarrollar esta práctica se tuvo que hacer uso de los conocimientos previamente desarrollados en las prácticas desarrolladas anteriormente.

Martínez Chávez Jorge Alexis

El mayor reto que presenté al desarrollar este proyecto fue el de hacer que leyera uno por uno los movimientos que tiene que hacer el robot, puesto que al principio sólo leía el primer movimiento y se quedaba ciclado hasta presionar el botón “CLEAR” y meter una nueva rutina. Para poder solucionar el problema opté por implementar una sentencia switch case para leer el movimiento actual y después de ejecutarlo detener todos los motores, para que en una nueva ejecución del ciclo cambie la posición del arreglo dónde se almacenan los movimientos y seguir con la ejecución adecuadamente.

Con el desarrollo de este proyecto pude aprender sobre el uso de arreglos en un programa para poder programar movimientos o subrutinas específicas mediante el uso de botones.

BIBLIOGRAFÍA

[1] Esneca Business School, “El robot programable: origen, concepto y sectores,” *Esneca*, Jun. 10, 2019. https://www.esneca.com/blog/robot-programable/ (accessed May 01, 2022).
[2] Gobierno de Canarias, “Robots educativos para edades tempranas,” *Centro del Profesorado Gran Canaria Sur*, Jun. 10, 2017. https://www3.gobiernodecanarias.org/medusa/edublog/cprofesgrancanariasur/robots-educativos-para-edades-tempranas/ (accessed May 01, 2022).