A dark green vertical bar is positioned on the left side of the page. A green arrow-shaped banner points to the right from this bar, containing the date '21-9-2021'. In the bottom-left corner, there are several thin, curved lines in dark green and light gray, resembling stylized grass or reeds.

21-9-2021

Proyecto 2: Bee-bot

Martínez Coronel Brayan Yosafat

3CM17 Introducción a los microcontroladores
FERNANDO AGUILAR SÁNCHEZ

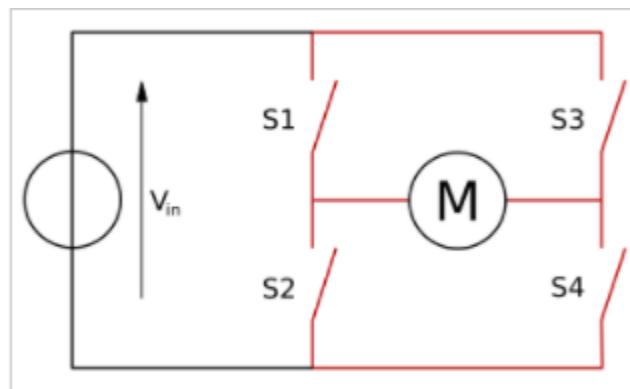
Objetivo

El objetivo de este proyecto es diseñar un móvil programable el cual será capaz de grabar 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 programar, ejemplo 10, 20, 30, etc.). Para iniciar la secuencia deberá oprimir el botón GO y para detenerse el botón PAUSE o terminar la secuencia previamente grabada.

Introducción teórica

Puente H

Un Puente en H es un circuito electrónico que generalmente se usa para permitir a un motor eléctrico DC girar en ambos sentidos, avance y retroceso. Son ampliamente usados en robótica y como convertidores de potencia. Los puentes H están disponibles como circuitos integrados, pero también pueden construirse a partir de componentes discretos.

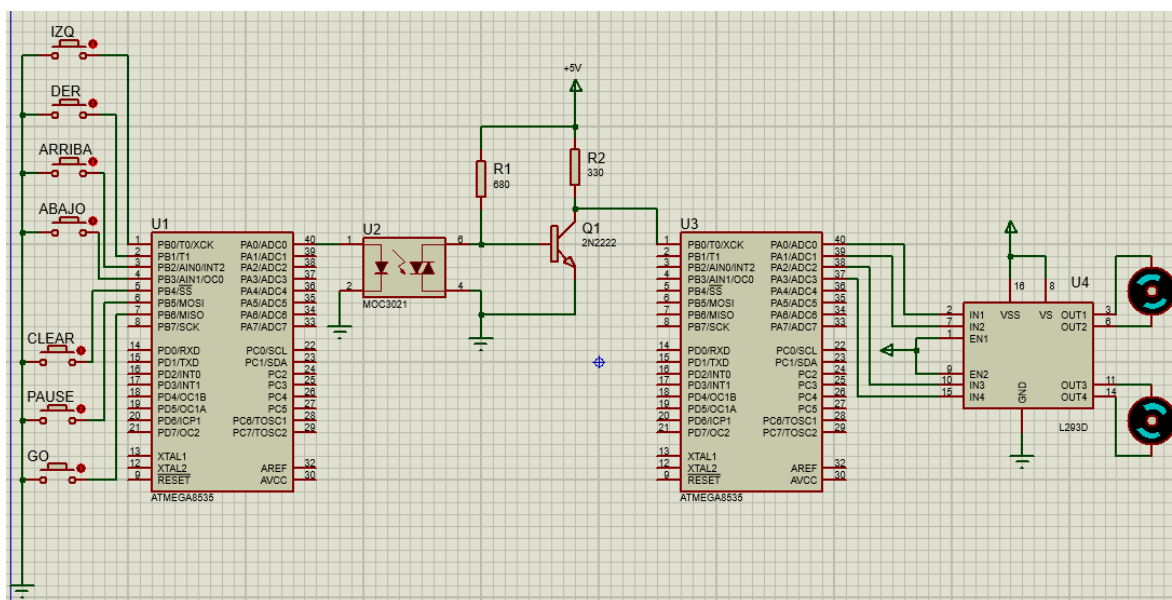


El término "puente H" proviene de la típica representación gráfica del circuito. Un puente H se construye con 4 interruptores (mecánicos o mediante transistores). Cuando los interruptores S1 y S4 están cerrados (y S2 y S3 abiertos) se aplica una tensión positiva en el motor, haciéndolo girar en un sentido. Abriendo los interruptores S1 y S4 (y cerrando S2 y S3), el voltaje se invierte, permitiendo el giro en sentido inverso del motor.

Con la nomenclatura que estamos usando, los interruptores S1 y S2 nunca podrán estar cerrados al mismo tiempo, porque esto cortocircuitaría la fuente de tensión. Lo mismo sucede con S3 y S4.

Desarrollo experimental

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



Estructura del programa

Código A

/*****

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 : 13/11/2021

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 ARR PINB.2
```

```
#define ABA PINB.3
```

```
#define CLR PINB.4
```

```
#define PAU PINB.5
```

```
#define GO PINB.6
```

```
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;
```

```
// 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
DDB=(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);

// Instrucciones: grabar y leer
if (mode == 0 && write_index < size) { // Puede grabar
    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: // Derecha  
        PORTA = 0xff;  
        delay_ms(50);  
        PORTA = 0x00;  
        delay_ms(450);  
        break;
```

```
    case 2: // Izquierda  
        PORTA = 0xff;  
        delay_ms(25);  
        PORTA = 0x00;  
        delay_ms(475);  
        break;
```

```
    case 3: // Arriba  
        PORTA = 0xff;  
        delay_ms(75);  
        PORTA = 0x00;  
        delay_ms(425);  
        break;
```

```
    case 4: // Abajo  
        PORTA = 0xff;  
        delay_ms(100);  
        PORTA = 0x00;  
        delay_ms(400);  
        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;
}
}

```

Código B

/*****

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 : 13/11/2021

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;
```

```
}  
}  
}
```

Observaciones y Conclusiones

Afortunadamente no me tomó tanto tiempo como la otra, aunque el botón de pausa hace lo que quiere, seguramente si el envío fuera diferente entonces no habría problema con eso, fuera de eso, creo que fue divertido hacer este proyecto, Espero no tardar tanto con la entrega del resto de prácticas.

Bibliografía

Puente H: [https://es.wikipedia.org/wiki/Puente_H_\(electr%C3%B3nica\)](https://es.wikipedia.org/wiki/Puente_H_(electr%C3%B3nica))