Alejandro Ossio Díaz, A01209122.

Gerardo Daniel Naranjo Gallegos, A01209499.

Roberto Saavedra Figueroa, A01209689.

Profesor: Agustín Domínguez Oviedo.

Curso de microcontroladores.

ITESM, campus Querétaro.

10 de marzo de 2016.

**Práctica 1: Luces Kitt.**

# Explicación del circuito y código.

**Circuito:**

El circuito consiste en un microprocesador (chip) ATMega328P, resistencias de 270Ω, *LED’s*, un potenciómetro y un *switch*.

El microprocesador ATMega328P es programado por medio de la computadora, con la ayuda del software Atmel Studio y con el código que se puede encontrar posteriormente en el documento. Este chip, al ser un microcontrolador, se encargará de llevar a cabo todo el procesamiento necesario para cumplir con el código; el objetivo es prender *LED’s* en distintas secuencias.

El potenciómetro funciona regulando el tiempo. El potenciómetro rotatorio funciona como una resistencia variable o, mejor dicho, que puede ser regulada por el usuario a través de una perilla. El potenciómetro manda un valor entre 0 y 1023 (en función del movimiento de la perilla), dato que será interpretado por el microprocesador y que determina el tiempo que permanece encendido un led. Dicho de otra manera, el potenciómetro regula la velocidad de cambio de un led a otro en la secuencia.

El *switch* es utilizado para seleccionar entre dos secuencias distintas, simulando a Kitt el auto increíble. Se utiliza el 0 para seleccionar una secuencia y el 1 para la otra secuencia.

Utilizamos 8 *LED’s* que prenden en dos secuencias distintas, dependiendo de la posición del *switch* puede ser una u otra, que simulan la secuencia de Kitt (el auto increíble). Cabe mencionar que cada chip tiene su respectiva resistencia.

La primera secuencia consiste en encender el primer led, esperar un tiempo con ese led encendido (según el potenciómetro) y finalmente apagar el led y encender el siguiente led; así sucesivamente hasta el último y viceversa, del último al primero.

La segunda secuencia consiste en encender el primer y último led simultáneamente, esperar un tiempo (potenciómetro) y posteriormente apagar ambos *LED’s* para después encender el segundo y el penúltimo (séptimo); así sucesivamente hasta encender el cuarto y el quinto al mismo tiempo y se repite el proceso a la inversa, es decir, se enciende el tercero y el sexto led, después el segundo y el séptimo… hasta completar un ciclo y volver a iniciar indefinidamente.

**Código:**

El código está hecho en C. El archivo generado al compilar el código en CodeVision se programa en el microcontrolador ATMega328P con el software Atmel Studio. Se incluyen tres librerías, “MEGA328P.h”, “delay.h” y “stdio.h”.

Existe una parte que se encarga de interpretar los valores recibidos por el potenciómetro y transformarlos en una variable de tiempo, la cual determina cuánto tiempo permanecerá encendido un led antes de apagarse y encender el siguiente (o bien, en el código, pasar a la siguiente instrucción).

Posteriormente, se encuentra otra parte del código. Inicia con una condición “if” cuya función es seleccionar si “PINC.0” es igual a 1 o a 0; “PINC.0” hace referencia a la lectura del pin C0 del ATMega328P que recibe la señal del *switch*. Por lo tanto, si recibe un 1, hará una secuencia y si recibe un 0 hará otra. Posteriormente, se utiliza la función “PORTD=0x01;” lo que significa que se manda una señal al puerto D (PORTD), la señal es en hexadecimal (0x) y es 01. Lo que significa que, en binario, recibe 00000001; con lo cual, el puerto D solo tendrá un uno en su bit menos significativo. El uno en el puerto D encenderá el led en esa posición. Después, encontramos la función “delayKitt();” que delimita el tiempo que permanecerá encendido el led, antes de apagarlo y pasar a la siguiente función; el tiempo está dado por la conversión analógico-digital (proveniente del potenciómetro), medido en microsegundos. La secuencia del puerto D y el delay se repiten hasta completar las distintas secuencias y en forma recíproca.

# Listado del programa en C.

#include <MEGA328P.h>

#include <delay.h>

#include <stdio.h>

// Voltage Reference: AVCC pin

#define ADC\_VREF\_TYPE ((0<<REFS1) | (1<<REFS0) | (0<<ADLAR))

// Read the AD conversion result

unsigned int read\_adc(unsigned char adc\_input)

{

ADMUX=adc\_input | ADC\_VREF\_TYPE;

// Delay needed for the stabilization of the ADC input voltage

delay\_us(10);

// Start the AD conversion

ADCSRA|=(1<<ADSC);

// Wait for the AD conversion to complete

while ((ADCSRA & (1<<ADIF))==0);

ADCSRA|=(1<<ADIF);

return ADCW;

}

void delayKitt()

{

delay\_ms(read\_adc(4));

}

void main()

{

// ADC initialization

// ADC Voltage Reference: AVCC pin

// ADC Auto Trigger Source: ADC Stopped

// Digital input buffers on ADC0: On, ADC1: On, ADC2: On, ADC3: On

// ADC4: Off, ADC5: On

DIDR0=(0<<ADC5D) | (1<<ADC4D) | (0<<ADC3D) | (0<<ADC2D) | (0<<ADC1D) | (0<<ADC0D);

ADMUX=ADC\_VREF\_TYPE;

ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);

ADCSRB=(0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

DDRD=0xFF;

while(1)

{

if(PINC.0==1){

PORTD=0x01;

delayKitt();

PORTD=0x02;

delayKitt();

PORTD=0x04;

delayKitt();

PORTD=0x08;

delayKitt();

PORTD=0x10;

delayKitt();

PORTD=0x20;

delayKitt();

PORTD=0x40;

delayKitt();

PORTD=0x80;

delayKitt();

PORTD=0x40;

delayKitt();

PORTD=0x20;

delayKitt();

PORTD=0x10;

delayKitt();

PORTD=0x08;

delayKitt();

PORTD=0x04;

delayKitt();

PORTD=0x02;

delayKitt();

}

if(PINC.0==0){

PORTD=0x81;

delayKitt();

PORTD=0x42;

delayKitt();

PORTD=0x24;

delayKitt();

PORTD=0x18;

delayKitt();

PORTD=0x24;

delayKitt();

PORTD=0x42;

delayKitt();

}

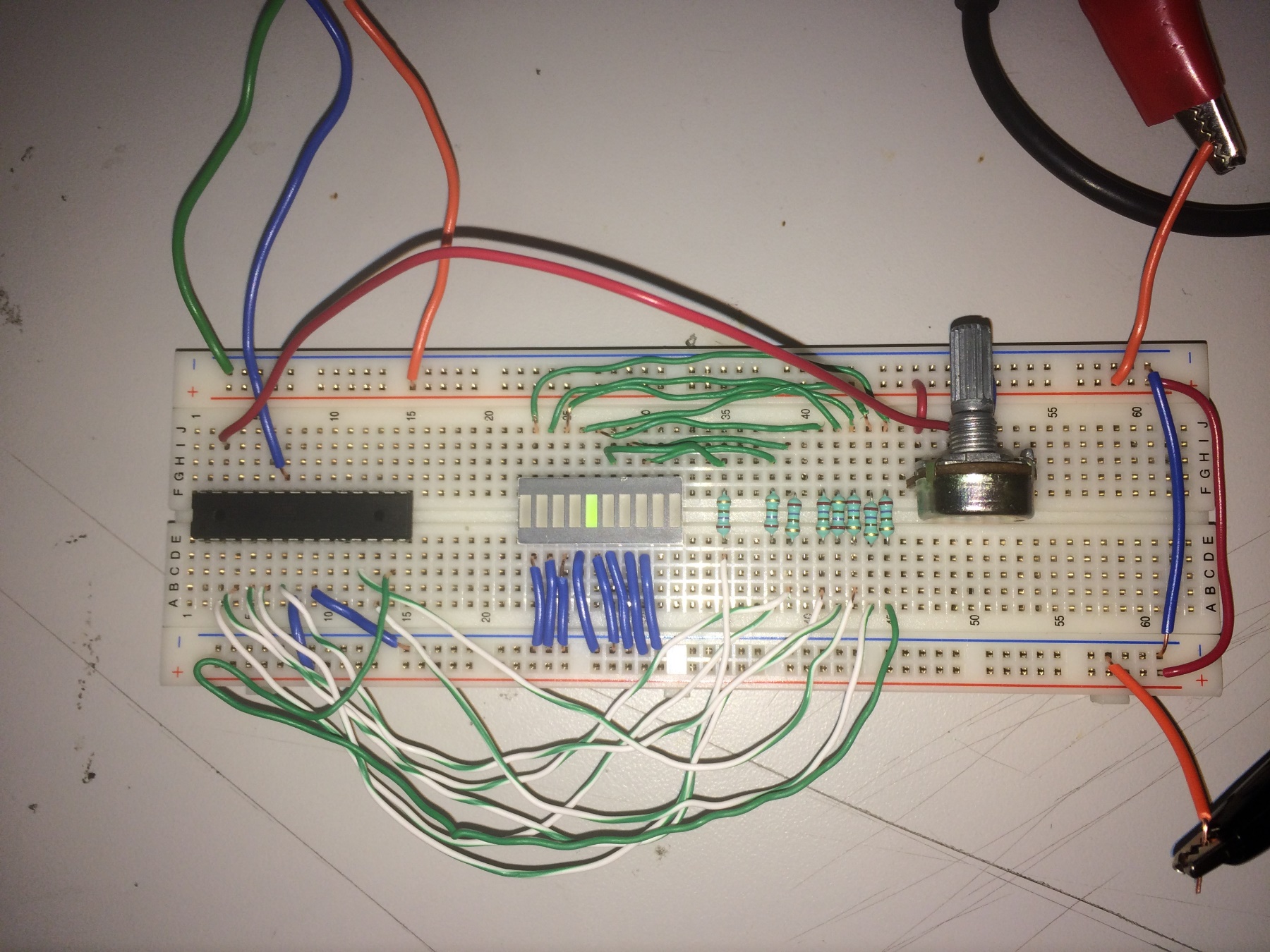
}

}

# Diagrama eléctrico completo de la práctica.



# Fotografía del circuito funcionando.



# Conclusiones.

**Gerardo Naranjo:**

Esta práctica me ayudó a practicar lo que hemos visto en clase, referente a la programación en C, CodeVision y el microcontrolador ATMega328P. Considero que la herramienta de Proteus es de gran ayuda para verificar el funcionamiento del código en el circuito, antes de programar el chip.

**Roberto Figueroa:**

Durante el desarrollo de la practica tuvimos algunos problemas, como que el chip no nos funcionaba debido a que estaba quemado, por lo que tuvimos que optar por otro, el cual había un sinfín de errores, que no se debía a nuestro circuito, si no a los instrumentos que estábamos utilizando, una vez que descubrimos el error, pudimos resolver fácilmente el circuito y lo aplicamos sin ningún problema, fue una buena experiencia para irnos familiarizándonos con el microcontrolador.

**Alejandro Ossio:**

Con esta primera práctica pudimos poner en práctica el microcontrolador y aprendimos cómo utilizarlo y programarlo, además, se pudo observar que el microcontrolador además de trabajar con 1's y 0's puede recibir valores analógicos, como fue el caso de controlar el delay de los *LED's* con el potenciómetro. De este modo pude darme la idea de lo que puede hacer un microcontrolador.