Chaleco de Seguridad

para Bicicletas

Asesora
Martinez Licona Alma Edith

Alumno
Rodríguez Sánchez Eduardo

Chaleco de Seguridad para Bicicletas
Universidad Autónoma Metropolitana
Proyecto Terminal II

Rodríguez Sánchez Eduardo

Resumen

BORRADOR
Documentación del proyecto terminal en computación en donde se fabrica un chaleco con iluminación para usuarios de bicicleta.

Índice

1. Objetivo

Crear un chaleco para usuarios de bicicletas de tracción humana, pedaleo asistido o con motor que permita alertar a conductores de vehículos motorizados, no motorizados y peatones sobre el movimiento que realiza el usuario en tiempo real.

2. Introducción

Una forma de contribuir a la seguridad vial empleando tecnología consiste en la fabricación y diseño de un producto; un chaleco controlado mediante Arduino, de arquitectura simple, que permita a cualquier persona construirlo, modificarlo y mejorarlo. La ventaja del dispositivo permite colocarlo en bolsas, mochilas u otras prendas, siempre y cuando se mantenga la orientación del dispositivo. El mercado ofrece una basta variedad de lamparas LED pero pocas forman parte de un sistema inteligente, y ninguna es programable. La intensidad luminosa es la base de este proyecto, la seguridad se vincula con la capacidad del sistema de visibilizar al usuario, los componentes modulares ofrecerán portabilidad para que el usuario lleve a donde sea y pueda cargarlo via USB.

2.1. Planteamiento del Problema

La movilidad en la Ciudad de México es un tema que ha tomado relevancia en los últimos años, las ciclovías se extienden por las alcaldías Benito Juárez, Cuauhtemoc, Miguel Hidalgo, Iztapalapa, etc. El número de ciclistas va en aumento así como los carriles confinados por lo que es necesario establecer reglas, mecanismos y políticas publicas para la protección del ciclista.

En las normas para la circulación de vehículos del Reglamento de Tránsito de la Ciudad de México publicado el 17 de agosto de 2015, se indica una regla general a cualquier tipo de vehiculo: *es indispensable que los conductores deban indicar la dirección de su giro o cambio de carril, mediante luces direccionales.*

Los conductores de vehículos no motorizados que no cumplan con las obligaciones estipuladas en los capítulos II y III, son amonestados verbalmente por los agentes y orientados a conducirse de conformidad con lo establecido por las disposiciones aplicables. Si bien las sanciones no son graves, un comportamiento irracional puede provocar un accidente entre un vehiculo o peaton y un ciclista, y dejar secuelas en el ciclista.

2.2. Planteamiento de la Visión

La seguridad de la persona es importante al rodar por la ciudad en condiciones de noche o baja visibilidad. La iluminación de la bicicleta es una técnica importante para hacerse notar al circular por una vialidad. Sin embargo, el tamaño y forma de la bicicleta puede reducir la visibilidad de una lampara tradicional que se ajusta debajo del asiento, por lo que resulta importante que las luminarias se encuentren en el cuerpo de la persona y no en la estructura de la bicicleta. Un sistema de luces sencillo y apropiado permitirá al ciclista en movimiento alertar a otros usuarios de la vía su desplazamiento y en su caso el cambio de carril. En otras palabras, un chaleco con luces direccionales que se activan al girar el torso hacia una u otra direción.

2.3. Elección de componentes

2.3.1. Plataforma

El mundo de los dispositivos listos para usarse y de fácil aprendizaje estan tomando fuerza en el desarrollo de proyectos. Actualmente los proyectos DIY (Do It Yourself) por sus siglas en inglés, son muy populares debido a la gran cantidad de información disponible en internet, materiales y plataformas que promueven el desarrollo Open Source.

Las tarjetas con microcontroladores estan en boga debido a su bajo costo y facilidad de programar. Tarjetas como Raspberry Pi o Intel Edison cuentan con un microprocesador, son más sofisticadas pero su costo es mayor. En algunos casos conviene utilizar un microprocesador, y en otros un microcontrolador se adapta mejor. En caso del chaleco se usará Arduino Uno R3 ya que no se requiere gran capacidad de procesamiento, ni un sistemo operativo montado en la tarjeta que procese los datos en tiempo real, basta con coordinar los sensores para interpretar los datos que envian.

Arduino Uno Existen dos versiones o marcas de Arduino cuyos sitio web son **arduino.org** y **arduino.cc**. Hay bibliotecas y ejemplos de ambas compañias, afortunadamente es indistinto utilizar uno u otro contenido, solo es importante verificar que esta hecho para el hardware o modelo de interés, en este caso Arduino Uno.
En la figura 1 se muestra la tarjeta Arduino Uno R3, el cerebro de la tarjeta es un microcontrolador **ATmega328P**. Su precio oscila entre 100 y 500 MXN. Un clon de Arduino funciona exactamente de la misma manera pues el microcontrolador no cambia.

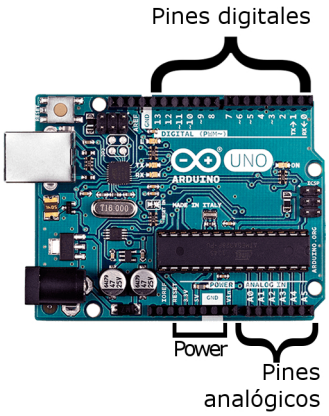


Figura 1: *Arduino Uno R3*

Arduino Micro En la figura 2 se muestra otro modelo de Arduino, al ser una tarjeta pequeña permitire montarse facilmente en la ropa. Este modelo en particular ya tiene implementada la interfaz USB por lo que solo es necesario conectarlo a la computadora para poder programarlo al igual que el modelo Uno. El voltaje de operación es de 5V, tiene pines analógicos, digitales, y canales para PWM. El único inconveniente de este modelo es el precio que oscila entre 680 y 1000 pesos MXN. Arduino Micro es ideal para el proyecto pero el costo total hace al chaleco caro por lo que se optará por el modelo económico.

Figura 2: *Arduino Micro utiliza un microcontrolador ATmega32U4.*

2.3.2. Limitaciones de Arduino Uno

ARDUINO MICROCONTROLLER	
Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog I/O Pins	6
EEPROM	1 KB
DC Current per I/O Pins	40 mA on I/O Pins; 50 mA on 3,3 V Pin

Figura 3: *Especificaciones de Arduino Uno R3*

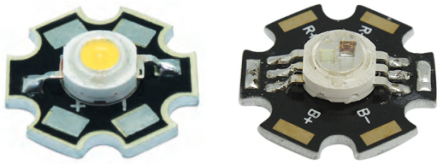
Arduino Uno tiene catorce pines digitales que pueden ser configurados como entradas o salidas, de los cuales seis pueden ser usados como salidas PWM. Utilizando la modulación de pulso se controlará la intensidad luminosa de los leds limitando la corriente que pasa a través de ellos. Posteriormente se agrega una etapa de amplificación para incrementar aun más la luminosidad.
El voltaje de operación es de 5V y consume aproximadamente 46.5 mA sin incluir la corriente que suministran los puertos. En las especificaciones del producto se advierten los siguientes puntos:

- Cada puerto I/O suministra una corriente de $20mA$ a 5V en condiciones estables
- La corriente máxima DC por puerto I/O es de $40mA$
- La corriente que suministra VCC es de hasta $200mA$
- La suma de todas las corrientes de los puertos no debe exceder los $150mA$.

Es necesario aplicar modulación de pulso en los LEDs para que los puertos no excedan la corriente máxima como se mostrará mas adelante. Además, al conectar distintos sensores es importante conocer la corriente de operación de cada módulo para no exceder los limites de Arduino y quemar la tarjeta.

2.3.3. Luces

Las luces traseras comerciales para bicicleta utilizan LEDs ultrabrillantes, la ventaja es que consumen muy poca corriente pero en condiciones de poca luz no son muy intensos y además requieren de resistencias para limitar la corriente que fluye en ellos. En cambio, un LED de potencia como el de la figura 4a, se controla por medio de un driver que suministra una corriente constante, por lo que no es necesario el uso de resistencias.
Por ejemplo, la corriente nominal de un LED blanco puro de 3W es de $750mA$, mientras que la corriente nominal para cada color en el LED RGB es de $350mA$. Dos cosas son importantes, el voltaje de operación y la corriente nominal.



(a) **(b)**

Colores disponibles	Voltaje de Operación [V]		Consumo [W]	Flujo Luminoso [lm]	
	Min	Max		Min	Max
R	2.0	2.6	1.0	30	40
G	3.0	3.8	1.0	50	60
B	3.0	3.8	1.0	10	25

(c)

Figura 4: *(a) LED blanco puro, (b) led RGB,(c) características de cada color.*

Los puertos de Arduino no son capaces de suministrar corrientes tan grandes aunque el voltaje y corriente de cada puerto es suficiente para poner en funcionamiento las luces a un nivel bajo. Así, a un voltaje de $5V$ y una corriente de $20mA$ la corriente es tan pequeña que el LED apenas y se calienta.

A medida que la corriente suministrada aumenta, el calor en el componente también lo hace, un disipador de calor ayuda a transferir la temperatura y evitar daños en las luminarias. Una elección adecuada de valores por debajo de los nominales permitirá mantener un nivel suficiente de luz sin generar mucho calor en el sistema.

2.3.4. Sensores y otros componentes

El proyecto necesita de dispositivos que permitan detectar movimiento como el acelerómetro, el giroscopio o el GPS. Los primeros dos son más económicos y fáciles de programar. El peso y tamaño de los componentes son caracteristicas esenciales en la elección de sensores.

Acelerómetro ADXL335 Es capaz de medir la aceleración de la gravedad de la tierra. En la figura 5 se muestra el chip montado en una placa, las resistencias de *pull-up* ya estan incluidas. Este sensor tiene una salida analógica por cada eje, lo que se mide en las terminales de salida es un voltaje que varía de acuerdo a la posición en que se encuentre el chip.
Las ventajas que tiene este modulo son:

1. Tamaño reducido
2. Ligero
3. $V_{cc} = 3{,}3V$ y consume I = $350\mu A$

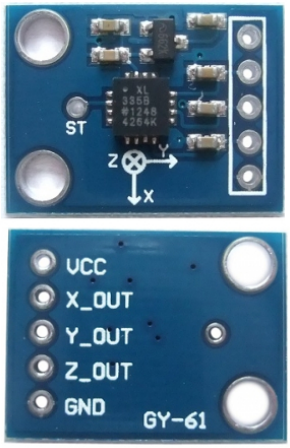


Figura 5: *Acelerómetro ADXL335.*

Las pruebas revelan que las variaciones de voltaje en las terminales X_OUT, Y_OUT, Z_OUT son constantes incluso cuando el sensor esta fijo, lo que implica ruido. El modelo es muy sencillo y ayuda a entender el funcionamiento del acelerómetro pero se necesita un dispositivo más sofisticado que incluya un giroscopio pues el acelerómetro solo detecta movimiento cuando la aceleración de la gravedad se ejerce sobre alguno de los ejes coordinados, es decir, la cara del sensor que apunte hacia arriba tendrá el mayor valor de voltaje, ya sea X, Y o Z.

Utilizando esta principio, un eje se toma como referencia para saber que el usuario se encuentra en una posición fija y vertical al andar en bicicleta como se muestra en la figura 6. Si ocurre un accidente y el usuario cae al suelo entonces la aceleración de la gravedad se ejerce sobre un eje distinto al de referencia y así se detectan movimientos bruscos que cambian radicalmente la posición del usuario.

Los ejes en una bicicleta Al rodar una bicicleta las llantas siempre se encuentran alineadas si se viaja en linea recta. La bicicleta siempre va sobre el plano xy y el casco apunta hacia el eje z, pensando en un sistema de referencia como el de la figura 6.
Si se considera al eje z como eje de referencia, el usuario siempre se encuentra en una posición segura si la aceleración de la gravedad recae sobre este eje. El sistema implementa la señal SOS si la aceleración se ejerce sobre un eje distinto al de referencia, lo cual implicaría una caída de la bicicleta o cualquier otro percance.

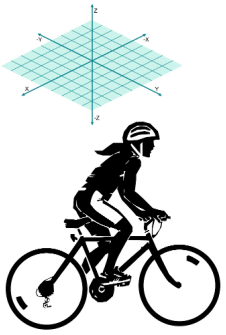


Figura 6: *Ejes coordenados X, Y, Z.*

Acelerómetro y giroscopio de tres ejes MPU-6050

Algunas características del dispositivo son:

1. Tamaño pequeño y ligero
2. $V_{cc} = 3{,}3V$, consume $I = 350\mu A$
3. Incluye regulador de voltaje
4. Se comunica con I2C
5. Se puede sincronizar con otros módulos.

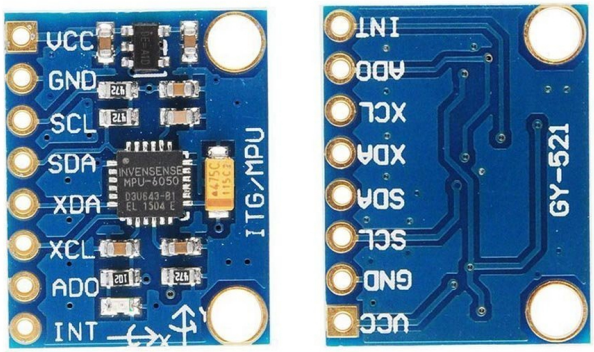


Figura 7: *Acelerómetro con giroscopio GY-521.*

Giroscopio y detección del á ngulo Utilizando la capacidad del sensor para medir movimientos rotacionales es posible activar un trigger que active las luces direccionales. El sentido del giro determina si la vuelta es hacia la izquierda o la derecha. En la figura 8 se muestra que cuando un ciclista da vuelta, gira alrededor del eje vertical con respecto al piso, o como lo hemos llamado eje de referencia.

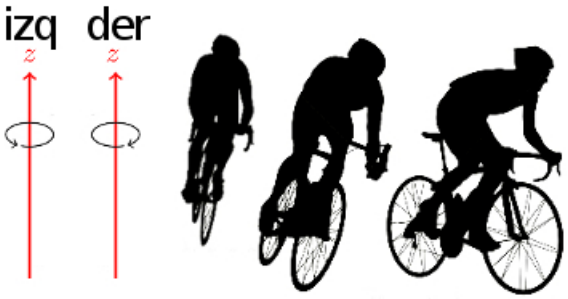


Figura 8: *Giro alrededor de un eje al dar vuelta.*

Con este módulo es posible medir dos cosas, la aceleración de la gravedad y la rotación, lo que se traduce a una luz de emergencia SOS activada por el acelerómetro y un interruptor de luces direccionales controlado por el giroscopio.

Hasta ahora el sistema se compone esencialmente de una tarjeta Arduino, un sensor

que incluye acelerometro y giroscopio y luces LED. La integración de todos los elementos se hace en una protoboard para pruebas muy básicas, en nuestro caso se soldarán los elementos a una tarjeta llamada protoshield la cual permitirá montar todo en una sola tarjeta que se conecta a Arduino.

Protoshield Existen métodos para ensamblar los componentes como tarjetas de circuito impreso, sin embargo un protoshield se puede montar en el arduino para tener una sola pieza que posteriormente se fija en el chaleco. En la figura 9 se muestra el escudo para el modelo Uno.

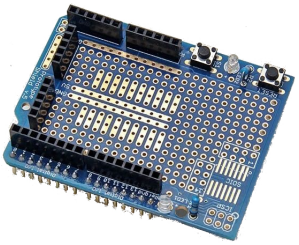


Figura 9: *Protoshield v.5.*

3. Desarrollo

En la figura 10 se muestra el flujo general de la aplicación. A groso modo, una vez energizado el sistema lo primero que hace Arduino es configurar los puertos como puertos de salida, despues se ejecuta una secuencia de luces para verificar que todos los puertos funcionan correctamente, si algun LED no enciende en esta etapa significa que el puerto falló o hay una desconexión en alguna de las terminales.

Cuando termina la secuencia de verificación se inicializa la comunicación I2C con el giroscopio, es decir, se le manda un mensaje para despertarlo y tenerlo a la espera de envio de datos. Posteriormente se ejecuta el patrón de luces para el modo standby, esto es, la secuencia que siempre se ejecuta mientras el sistema esta activo. A partir de este punto el sistema solo lee los sensores, para detectar si hubo algún giro lo que implica un cambio en el valor del giroscopio o un cambio significativo en el acelerómetro.

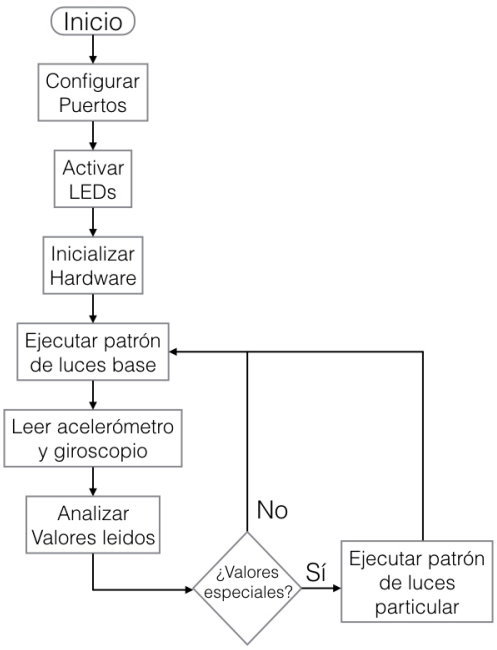


Figura 10: *Diagrama de flujo del programa.*

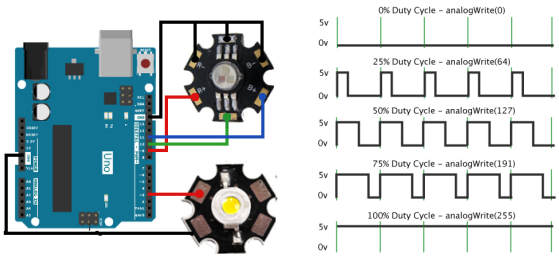
El sistema lee el acelerometro para saber si hubo un cambio en la posición del usuario, y lee el giroscopio para saber si el usuario ha activado la luz direccional. La única forma de detener el sistema es desconectando la alimentación. El siguiente paso consiste en desarrollar el primer bloque, es decir, configurar y conectar correctamente los LEDs, posteriormente se configurarán y programarán los sensores.

3.1. Prueba de LEDs

La terminal negativa del LED se conecta a tierra y la terminal positiva se conecta a uno de los puertos digitales marcados con el simbolo ∼, lo cual significa que se puede utilizar PWM. En la figura 11a se muestra la forma en que se conecta un LED blanco de 3W y un led RGB de 3W. Cada PIN en uso se configura como salida, interesa mandar un uno digital (5 Volts) para encender el LED, y un cero para apagarlo. Los pines se configuran como sigue:

```
const int RED = 9, GREEN = 10, BLUE = 11, LEFT=6, RIGHT=5, SUB=3;
void setup() {
  pinMode(SUB,OUTPUT);
  pinMode(LEFT, OUTPUT);
  pinMode(RIGHT, OUTPUT);
  pinMode(RED,OUTPUT);
  pinMode(GREEN,OUTPUT);
  pinMode(BLUE,OUTPUT);}
```

Se definen como constantes los colores RGB y los LEDs blancos que representan vuelta hacia la derecha y hacia la izquierda ya que estos no cambiaran una vez soldados los componentes.



(a) **(b)**

Figura 11: *(a) Diagrama de conexiones, (b) Variación del pulso de salida al utilizar la función* ***analogWrite()****.*

3.1.1. Limitando la Corriente con PWM

Recordando las especificaciones, cada puerto de Arduino puede suministrar una corriente nominal de $20mA$. Como el led se conecta directamente el uso de la función **digitalWrite()** provoca que se consuma una corriente de hasta $40mA$, lo que puede dañar el puerto con el uso prolongado.

```
pinMode(pin, OUTPUT);
digitalWrite(pin, HIGH);
```

Para evitar que los LEDs demanden una gran cantidad de corriente de los puertos se utiliza la función **analogWrite()** la cuál nos permite modificar el ancho del pulso en la señal de salida como se muestra en la figura 11b. Con esto se tiene un rango de valores

que van de cero hasta 255. El valor máximo que devuelve **analogWrite()** equivale a un uno digital o 5 Volts. A diferencia de **digitalWrite()**, la función **analogWrite()** permite regular la intensidad del LED cambiando el valor del segundo parametro como sigue:

```
analogWrite(RED,80); //Enciende led a un valor de 80 en la escala 0-255
```

Después de varias pruebas se construye una tabla que relaciona la corriente de salida del puerto con el valor que toma el parámetro de la función **analogWrite()**, en otras palabras, para limitar la corriente a un valor aproximado se elige cualesquiera de los valores en el primer renglón de la tabla de la figura 12. Por ejemplo, para suministrar 1mA al LED azul , se necesita **analogWrite(BLUE,5)**.

I(mA)	R	G	B	W
1	4	4	5	4
2	7	8	9	8
3	10	12	14	12
5	18	20	26	21
10	36	40	45	56
15	55	61	59	65
26	87	92	101	111

Figura 12: *Tabla de mapeo que relaciona un valor de corriente con su respectivo valor para cada uno de los LEDs.*

La estabilidad del circuito se probó por periodos de hasta 5 horas, los LEDs no se calientan y la corriente tampoco aumenta significativamente pues los LEDs nunca estan encendidos permanentemente si no que parpadean de acuerdo a la secuencia.

El patron para verificar que todos los puertos se configuraron y funcionan correctamente es el siguiente, **secuenciaRGB()** se manda llamar desde **setup()** para que solo se ejecute una vez antes de que el programa entre al bucle infinito **loop()**.

```
void secuenciaRGB(){
  analogWrite(RED,80); //Rojo
  delay(170);
  analogWrite(BLUE,59); //Rosa = rojo y azul
  delay(170);
  analogWrite(RED,0); //Apaga rojo y queda azul
  analogWrite(GREEN,61); //Turquesa = azul y verde
  delay(170);
  analogWrite(BLUE,0); // Apaga azul queda verde
  delay(170);
  analogWrite(RED, 55); //Amarillo
  delay(170);
  analogWrite(RED,0);
  analogWrite(GREEN,0);
  analogWrite(BLUE, 59); //Azul
  delay(170);
  analogWrite(BLUE,0);
  delay(150);
  analogWrite(LEFT,10);//leds blancos
  analogWrite(RIGHT,10);
  delay(8);
  analogWrite(LEFT,0);
  analogWrite(RIGHT,0);
  delay(250); }
```

Una vez que el programa entra al bucle **loop()** y finaliza el setup, un patrón de luces debe estar corriendo todo el tiempo para iluminar el chaleco, simulando las sirenas que tienen algunos vehículos se crea la secuencia **standby()**.

```
void standby(){
    int i=80;
    while(i>=0){
      analogWrite(RED,i);
      delay(28);
      i-=5;
    }
    analogWrite(LEFT,56);
    analogWrite(RIGHT,56);
    analogWrite(BLUE,59);
    analogWrite(SUB,56);
   delay(100);
  analogWrite(LEFT,0);
   analogWrite(RIGHT,0);
       analogWrite(BLUE,0);
   analogWrite(SUB,0);
   delay(10);
}
```

El usuario final tiene la ventaja de diseñar sus propias secuencias y agregar las pausas necesarias, sin embargo estos patrones no deben ser muy extensos pues al finalizar la rutina de LEDs el microcontrolador debe actualizar los valores del acelerómetro y giroscopio para determinar la posición del usuario. A mayor duración de la secuencia de luces, más es el tiempo que tarda el programa en sensar si hay un giro en el torso del usuario. Como consecuencia, si el usuario gira mientras la secuencia de luces se ejecuta el programa no lo va a detectar. Es recomendable que las rutinas de luces no excedan los $500ms$ e importante mencionar que un LED encendido de manera prolongada a alta intensidad consume mayor corriente y tiende a calentarse con el transcurso del tiempo.

3.1.2. Incrementando la Intensidad Luminosa

Amplificador transistorizado Esta etapa es opcional y solamente brinda mayor corriente en los LEDs, lo que se traduce en mayor intensidad luminosa. Se utilizará un transistor BJT **2N2222A** como el de la figura 13 y una resistencia de $2{,}2k\Omega$ por cada LED. El transistor es un dispositivo de tres terminales: **Base, Colector y Emisor** capaz de amplificar corriente y voltaje. Un transistor tiene tres modos de operación: **Saturación**, la corriente crece rapido y el voltaje es pequeño. **Corte**, las corrientes del transistor son pequeñas por lo que consieramos que se comporta como un circuito abierto. **Región activa**, la corriente de colector $\mathbf{I_C}$ no varía mucho, es estable y hace que el transistor se comporte de forma lineal.

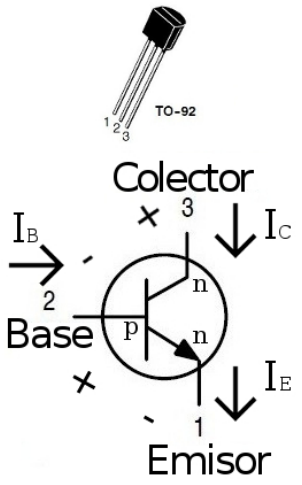


Figura 13: *Transistor 2N2222A con encapsulado plastico TO-92, en el simbolo se muestra el flujo de las corrientes.*

A pesar de que no hay componente de corriente alterna (AC) se utilizará una configuración llamada de emisor común, su nombre se debe a que la terminal emisor esta conectada a tierra y es común a la señal de entrada y la señal de salida, en otras palabras, la entrada se conecta en la base y la salida se mide desde el colector, el emisor va a tierra.

Polarización de un transistor Si solo observamos la unión *Base-Emisor* de la figura 13 podemos pensar en un diodo rectificador y aplicar las mismas suposiciones para analizar en corriente directa, es decir:

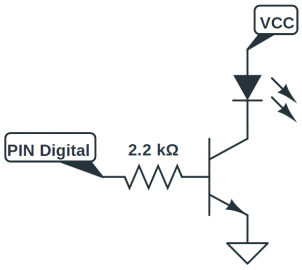
- Para romper la barrera entre placas se necesita al menos una diferencia de potencial de $0{,}7\,V$, es decir el voltaje de la placa positiva es mayor al voltaje de la placa negativa por $0{,}7\,V$. Aquí la base es la placa positiva (transistor NPN).
- La unión base emisor esta polarizada en directa si el voltaje de la base es mayor al voltaje del emisor en al menos $0{,}7\,V$ y polarizado en inversa si el voltaje de base es menor al voltaje del emisor
- Si la unión *Base-Emisor* esta polarizada en inversa entonces el transistor se comporta como un circuito abierto entre colector y emisor.

Para conseguir que haya corriente en el colector debemos inyectar los portadores del emisor en la región de base. Esto se realiza polarizando directamente la unión *Base-Emisor* (Veatch 1981, p.42). Si en cualquier instante se elimina la polarización directa entonces no habrá corriente en el colector. Las relaciones de las corrientes que fluyen a través del transistor estan dadas por las siguientes ecuaciones:

$$\begin{aligned} I_E &= I_B + I_C \\ \beta I_B &= I_C \end{aligned} \tag{1}$$

Las corrientes que se indican en la figura 13 cumplen con (1), esto es, la corriente que fluye por el emisor es igual a la corriente de colector más la corriente de base, y la corriente de colector es un múltiplo escalar de la corriente de base.

En este caso Arduino suministra una corriente de base I_B, el transistor amplifica dicha corriente y la suministra al LED en el colector para aumentar la intensidad luminosa.
El diagrama de conexiones se muestra en la figura 14a. Cada que el programa ejecuta la instrucción **analogWrite(PIN,valor)**, en el puerto correspondiente se emite un voltaje y una corriente. El rango es suficiente para polarizar en directa al transistor y generar una corriente mucho mayor en el colector que aumentará la luminosidad del LED dramáticamente.
Las pruebas generan una nueva tabla que relaciona el valor de la función **analogWrite()** y la corriente generada, ahora, en la terminal del colector como se muestra en la figura 14b.



(a)

analogWrite()	I (mA)
0	0
16	12.2
32	23.7
48	34.8
64	46.6
80	57.9
96	69.2
112	80.9
128	92.6
144	104.1
160	115.3
176	127
192	138.9
208	150.3
224	162.1
240	171.6

(b)

Figura 14: *(a)Diagrama de conexiones, (b)Mapeo correspondiente una vez que se conectan los transistores.*

Cuando al LED se le proporcionan corrientes mayores a los 80mA por periodos de hasta 250ms el calentamiento es evidente y se corre el riesgo de desgastar los componentes.

Si el usuario decide utilizar una etapa de amplificación es indispensable calcular los tiempos que los LEDs se mantienen encendidos. La función **analogWrite()** puede tomar el valor máximo de 255 en intervalos pequeños de tiempo pero a medida que el tiempo de encendido se prolonga, el valor de intensidad debe disminuir, de los contrario los LEDs se calientan y se puede dañar el transistor. La siguiente función es la versión final para el patrón de luces con etapa de amplificación:

```
void powerStandby(){
    int i;
    for(i=80; i>=0; i-=5){
      analogWrite(RED,i);
      delay(16);
    } //Parpadeo Color Rojo, duracion: 256ms
    delay(10);
    for(i=70; i>=0; i-=5){
      analogWrite(BLUE,i);
      delay(15);
    } //Parpadeo Color Azul, duracion: 210ms
    delay(10);
    analogWrite(SUB,intensidad);
    analogWrite(LEFT,intensidad);
    analogWrite(RIGHT,intensidad);
  delay(intensidad); //Duracion: 45 o 190ms
  analogWrite(LEFT,0);
   analogWrite(RIGHT,0);
       analogWrite(SUB,0);
   delay(50);
}
```

3.2. Leyendo el Sensor

El acelerómetro con giroscopio robustecerá al sistema, pues le permitirá tomar decisiones en base a las lecturas de giro y el eje vertical de referencia sobre el cual recae la aceleración de la gravedad como se mostró previamente en la figura 8.

La biblioteca **Wire.h** permite la comunicación con dispositivos I2C. La transmisión de datos se divide en dos partes: la línea SDA es la linea donde se envian los datos, la linea SCL es el reloj para poder sincronizar las peticiones del dispositivo maestro con los esclavos. En el protocolo I2C el dispositivo que solicita datos necesita enviar una petición a la dirección del sensor, especificarle si va a realizar lectura o escritura y la cantidad de registros que se solicitan.

Al iniciar el sistema, Arduino activa el sensor y lo mantiene a la espera. Esto se realiza dentro de **setup()** pues solo es necesario hacerlo una vez.

```
void setup() {
  Serial.begin(9600);//Inicia comunicacion serial
  Wire.begin();
  Wire.beginTransmission(MPU_ADDR); // Comienza transmision con (GY-521)
  Wire.write(0x6B); // PWR_MGMT_1 register, (Power management)
  Wire.write(0); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
}
```

Una vez que el programa entra al bucle infinito se ejecuta la secuencia de luces y posteriormente se miden dos parámetros: la aceleración en el eje de referencia y cambios en el giroscopio respecto a este mismo eje. Por lo tanto se diseñan dos funciones, una para el acelerómetro y otra para el giroscopio.

3.2.1. Solicitando Datos

Lo que tienen en común la lectura del acelerometro y el giroscopio es la manera en que se leen los datos. En ambos casos se necesita especificar la dirección del registro inicial que se quiere leer, posteriormente se indica el numero total de registros a leer.

```
void solicitarDatos(int RegistroInicial,int CantidadRegistros){
 Wire.beginTransmission(MPU_ADDR); //Iniciar transmision
 Wire.write(RegistroInicial); // Empezar con el registro
 Wire.endTransmission(false);
 Wire.requestFrom(MPU_ADDR, CantidadRegistros, true); //Solicitar datos
}
```

Por ejemplo, la aceleración en X se compone de una parte alta que corresponde a la dirección de memoria **0x3B** y una baja que corresponde a la dirección **0x3C**, en su conjunto forman el entero de 16 bits que representa el valor de la aceleración en X. Tanto acelerómetro como giroscopio tienen 3 ejes, en total son 6 ejes que es igual a seis registros, donde cada registro tiene un byte alto y uno bajo, por eso la cantidad de registro siempre es un multiplo de dos.

3.2.2. Lectura Acelerómetro

Al leer la aceleración sobre el eje de referencia, el valor es alto. El eje de referencia siempre es el eje que apunta hacia el espacio, es decir, perpendicular a la superficie de la tierra. Si un accidente llegase a ocurrir y por ejemplo, el eje de referencia es Y, la nueva posición va a tener a X como eje perpendicular lo que provoca que el chaleco emita una alerta de emergencia. Al momento en que el eje de referencia vuelve a su posición original la alerta de emergencia se detiene. La función **actualizarAccel()** es recursiva e implementa de manera sencilla el encendido y apagado de la secuencia del patron de emergencia SOS.

```
void actualizarAccel(){
 solicitarDatos(0x3D, 2);
 accelerometer_y = Wire.read()<<8 | Wire.read();
 Serial.print("aY = ");
 Serial.println(convert_int16_to_str(accelerometer_y));
 while(accelerometer_y < 9000){
   sosRGB();
   actualizarAccel();
}
```

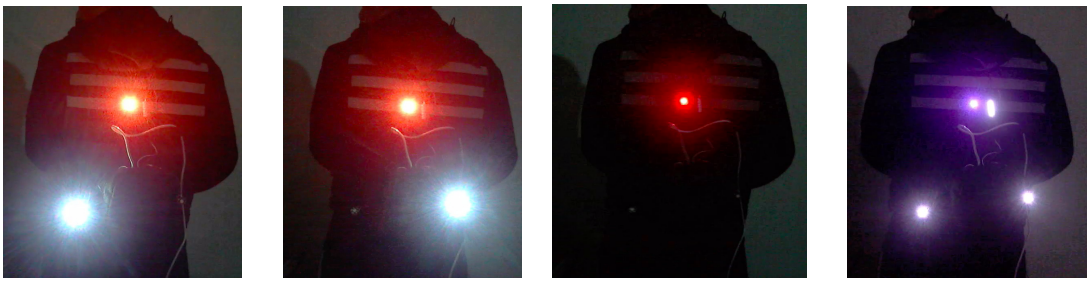
3.2.3. Lectura Giroscopio

Finalmente la función para leer el giroscopio permite encender las luces direccionales. El valor del giro sobre el eje de referencia activa las luces para indicar que se realizará un movimiento hacia la izquierda o hacia la derecha. **GyroUpdate** es el valor del eje de referencia, el giro sobre el eje z ayuda a detectar movimientos bruscos, mientras este valor no exceda cierto limite las direccionales se pueden encender de lo contrario es probable que el usuario haya tenido un movimiento brusco debido al camino y por lo tanto las luces direccionales no se activan aunque haya un giro sobre el eje de referencia. La condición se cumple solamente cuando el usuario hace un giro con el torso de manera suave que no afecta significativamente a los demás ejes.

```
void actualizarGyro(){
     solicitarDatos(0x45,4);
     GyroUpdate = Wire.read()<<8 | Wire.read();
     gyro_z = Wire.read()<<8 | Wire.read();
     if(gyro_z > -1900 || gyro_z < 1900 ){
       Serial.print("GyroUpdate = "); Serial.println(GyroUpdate);
       Serial.print("Gyro_z = "); Serial.println(gyro_z);
       if(GyroUpdate>= 9900 ){
          parpadeoAmarillo(LEFT);
       }
       else if(GyroUpdate<= -9900){
          parpadeoAmarillo(RIGHT);
       }
     }
}
```

Esta doble verificación en los ejes ayuda a detectar falsos positivos. A través de las pruebas se nota que pasar por un bache o un sitio muy irregular hay un giro significativo en todos los ejes lo cual ayuda a detectar que se trata de un movimiento brusco y no una activación de direccional. Dada la orientación del dispositivo los valores negativos de giro corresponden la vuelta a la derecha y los valores positivos la vuelta a la izquierda.

En la figura 15 se muestran cuatro casos, 15a y 15b son las direccionales que indican la vuelta, es decir corresponden a la función **parpadeoAmarillo()**. Aquí el led RGB toma un color naranja y solo se activa el led blanco correspondiente. Mientras el sistema se actualice y el valor no este fuera de rango se alterna entre las fotos 15c y 15d.



(a) **(b)** **(c)** **(d)**

Figura 15: *(a) Vuelta a la izquierda, (b) Vuelta a la derecha, (c) Sistema energizado, (d) Captura de patrón de luces base.*

4. Conclusión

Los resultados arrojan un sistema final que es visible a una distancia de 20 metros con etapa de potencia el cual tiene sus pros y sus contra. Primero es importante destacar que el ensamblaje final sí requiere habilidad para soldar los transistores y acomodar los cables, así como el sensor, lo cual resta puntos a la facilididad de construirlo. Por otra parte el balance en el suministro de energía permite una eficiencia tal que con una bateria de 4000mAh se realizaron viajes de hasta 3 horas continuas. Una de las ventajas sobre las luces tradicionales es que al utilizar LEDs de potencia a una corriente relativamente baja se puede lograr una intensidad luminosa suficiente sin sacrificar muchos recursos energeticos.

Referencias

[1] Malvino Albert Paul, Bates David J (2007), *Principios de electrónica*, España: McGrawHill.

[2] Veatch Henry C (1981), *Fundamentos y aplicaciones de los circuitos de transistor*, España: Marcombo.

5. Anexo I

5.1. Cotización del proyecto

- Arduino Uno R3: $162,00
- MPU6050, acelerómetro con giroscopio: $90 pesos
- Power bank, $92 pesos
- 4 LEDs de 3W blanco puro, $80 pesos
- 1 LED de 3W RGB, $30 pesos
- Protoshield, $90 pesos
- 4 transistores BJT, $10 pesos
- 4 resistencias de 2.2k a medio watt, $1 pesos
- 4 metros de cable de distintos colores, $50 pesos
- Chaleco, $90 pesos

El **costo total** del proyecto es de $695 pesos a precio de mayoreo. Si el volumen de piezas aumenta el costo total del chaleco se reduce hasta en un 30 %. La personalización del chaleco tiene un impacto directo en el costo, pero la base del sistema que provee la iluminación tiene un costo aproximado de $300 e incluye la tarjeta Arduino, los cuatro LEDs Blancos y un juego de resistencias y transistor para cada LED. Por ejemplo, el uso de LEDs RGB es opcional y se considera un requerimiento no funcional, por lo tanto se puede omitir. El sistema se puede montar en una chamarra deportiva y esto también puede ahorrar el costo del chaleco. Se pueden reutilizar baterias tipo Power Bank y disminuir aun más el costo. Incluso si es necesario y el usuario así lo desea puede utilizar una versión distinta de Arduino, siempre y cuando contemple un minimo dos puertos con PWM.