UNIÓN DE ASOCIACIONES DE INGENIEROS TÉCNICOS INDUSTRIALES Y GRADUADOS EN INGENIERÍA DE LA RAMA INDUSTRIAL DE ESPAÑA (UAITIE)

“CONVOCATORIA 2019”

IV PREMIO NACIONAL DE INICIACIÓN A LA INVESTIGACIÓN TECNOLÓGICA

“Carreteras Inteligentes”

AUTOR/ES:
Rubén Canora, Miguel Campomanes, Alejandro Aylagas

BLOQUE TEMÁTICO:
Ahorro energético

NIVEL EDUCATIVO:
3º ESO

COORDINADOR:
Manuel Ruano

Marzo de 2019

UNIÓN DE ASOCIACIONES DE INGENIEROS TÉCNICOS INDUSTRIALES Y GRADUADOS EN INGENIERÍA DE LA RAMA INDUSTRIAL DE ESPAÑA

Resumen

El proyecto “Carreteras Inteligentes” ha sido concebido como propuesta eficiente a fin de solventar el excesivo consumo de alumbrado eléctrico en autovías que no tienen mucho tráfico. De este modo, el objetivo de esta memoria es plantear un significativo ahorro energético en cuanto al servicio de alumbrado, sin que ello conlleve un riesgo de la seguridad vial.

El sistema está supeditado a la detección del movimiento de los vehículos que circulan por la carretera. Al ser detectada la presencia de un automóvil, la farola más cercana al mismo, envía una señal a todas las farolas del circuito cerrado. En función de la proximidad del vehículo, las restantes farolas se encenderán o no, y así iluminarán la autovía, anticipándose al paso del automóvil. De este modo, el conductor no percibirá un descenso notable de la intensidad lumínica a su paso, dado que el circuito reacciona a su movimiento. Una vez que el vehículo ha sobrepasado la farola ésta se apagará hasta que se detecte el paso de otro vehículo.

Palabras Clave

Ahorro, eficiencia energética, carreteras, electricidad, alumbrado e innovación.

UNIÓN DE ASOCIACIONES DE INGENIEROS TÉCNICOS INDUSTRIALES Y GRADUADOS EN INGENIERÍA DE LA RAMA INDUSTRIAL DE ESPAÑA

Índice

UNIÓN DE ASOCIACIONES DE INGENIEROS TÉCNICOS INDUSTRIALES Y GRADUADOS EN INGENIERÍA DE LA RAMA INDUSTRIAL DE ESPAÑA

Nuestro Proyecto

1. Desarrollo

1.1 Introducción

Durante los dos últimos siglos, la industria tecnológica ha ido evolucionando vertiginosamente con el fin de mejorar la calidad de vida de las personas. Sin embargo, estos hitos de I+D requieren de muchos recursos; una materia prima, que estamos explotando constantemente, cuya substracción genera tales niveles de contaminación, que estamos mermando el planeta donde vivimos. Según un estudio del Programa de las Naciones Unidas para el Medio Ambiente (BBC News, Mundo, 2011), a mediados del presente siglo, consumiremos tres veces más los recursos que usamos ahora. Esto es un gran problema. Si a la tasa de población mundial que no para de crecer, le sumamos la mala gestión de los recursos naturales disponibles, no tendremos futuro. Por eso, hay que buscar maneras de satisfacer las necesidades humanas, optimizando eficientemente el uso de los recursos. En el proyecto “Carreteras Inteligentes”, se ha centrado en la búsqueda de una solución eficiente al consumo de la energía eléctrica en un contexto real y trasladable a la rutina diaria.

1.2 Objetivos

- Desarrollar un sistema viable de alumbrado inteligente de carreteras, que reduzca significativamente el consumo de energía eléctrica y no reduzca la seguridad de los conductores.

- Realización de un proyecto funcional en el que se puedan usar los conocimientos asimilados y materiales utilizados en clase durante el curso de Tercero de Secundaria

1.3 Metodología

Los pasos seguidos para realizar este proyecto son: búsqueda de información, implementación del sistema completo en placas de prototipado (programa y circuito), prueba del proyecto en seis farolas (verificar la secuencia adecuada de su funcionamiento), soldadura y diseño de la carcasa 3D.

1.3.1. Búsqueda de información

Este prototipo dispone de los siguientes componentes electrónicos: sensores de ultrasonidos (para detectar el movimiento), placas Arduinos (microcontrolador), LED's (para iluminar la carretera) y sensores de luz (LDR).

Primero, trabajamos con los sensores de ultrasonidos para detectar el movimiento. Se partió de un código sencillo (presentado a continuación).

```
#define trigPin 13
#define echoPin 12
#define led 11

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(led, OUTPUT);
}

void loop() {
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration/2) / 29.1;

  if (distance < 10) {
    digitalWrite(led,HIGH);
}

else {
digitalWrite(led,LOW);
}

Serial.print(distance);
Serial.println(" cm");
delay(500);
}
```

Figura 1: Código del ultrasonidos

Para iluminar de forma eficiente con LED's de electrónica analógica básica, era necesario emplear varios dispositivos por farola. Se buscó la manera de encender varios simultáneamente en un mismo pin digital. Finalmente, se optó por usar un transistor 2N3904 que amplificara la señal. Hemos usado este circuito:

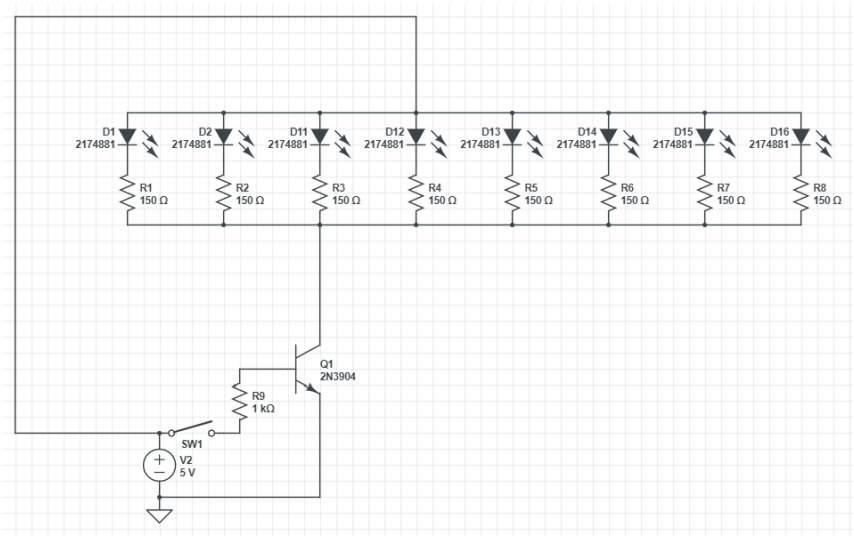

Figura 2: Circuito eléctrico de los LED's

Asimismo, se quería regular la intensidad lumínica de los LED's. Se utilizó un pin digital de salida del "*Modulación por Ancho de Pulso*"(PWM):

```
intdigPin = 10;
void setup() {

}
void loop() {
analogWrite(digPin,127); // Señal PWM a 50% en el PIN 10
}
```

La elección de una resistencia LDR permite que el circuito condicione su funcionamiento según la luz incidente recibida. Así, el proyecto reaccionará al nivel de luz del ambiente (noche, día, nublado). El código básico empleado es este:

```
const int LEDPin = 13;
const int LDRPin = 2;

void setup()
{
pinMode(LEDPin, OUTPUT);
pinMode(LDRPin, INPUT);
}

void loop()
{
   int value = digitalRead(LDRPin);
   if (value == HIGH)
   {
      digitalWrite(LEDPin, HIGH);
      delay(50);
      digitalWrite(LEDPin, LOW);
      delay(50);
   }
}
```

Figura 3: Código LDR

Se cambió el comando digitalRead por analogRead, pudiendo jugar con valores numéricos en vez de HIGH o LOW

La mayor dificultad fue la comunicación entre las distintas placas Arduinos entre sí. Inicialmente, esta conexión se planteó vía jerarquía maestro-esclavo, denominada I2C. Este sistema funciona con un maestro (que es el que controla a sí mismo y a los demás) y los esclavos (que no pueden empezar una comunicación entre ellos, sino que es éste el que tiene que "darles permiso"). No obstante, para una implementación real, no parece favorable una farola líder que eje de gobernación piramidal. Se optó porque todas las farolas pudieran compartir libremente información entre ellas sin restricciones. Con este fin, se ha empleado un sistema novedoso de comunicación entre placas Arduino, llamado PJON. PJON es un sistema de comunicación entre dispositivos en el que estos están conectados a un bus y envían datos a través de él.

Hay diversas maneras de vincular dispositivos entre sí, pero se ha empleado "SoftwareBitBang", donde todas las placas Arduino son maestros y están conectados a un mismo pin digital y al GND. Los paquetes de datos son enviados a todos los dispositivos conectados al bus. Cuando una farola

detecta movimiento envía un mensaje a todas las demás farolas indicando su número de orden. Cuando otra farola recibe el mensaje anterior evalúa su distancia a la farola que detectó el movimiento y entonces enciende su lámpara si está suficientemente próxima.

Al principio del código, se debe incluir la librería: #include<PJON.h>
Para inicializar el bus, se llamará a la función bus.begin ();
Para recibir mensajes, se utilizará bus.receive(50000);
Y al final del void loop, es necesario escribir bus.update ();
Para enviar mensajes a todos los dispositivos, se utilizará PJON_BROADCAST:

```
bus.send_packet(PJON_BROADCAST, "Message for all connected devices.", 34);
```

Figura 4: Código de envío de mensajes

En el código se usará bus.send. Con el código bus.send_packet se envía una sola vez, y con bus.send se envía las veces necesarias hasta que le lleguen a las demás farolas. Dado que se quiere enviar el mensaje a todas las farolas, se utilizará el código PJON_BROADCAST. A continuación, se emitirá el mensaje entre comillas, y el tercer comando del paréntesis, indica el número de caracteres que tiene el mensaje.

Para recibir el mensaje, se empleará la función void receiver_function (). Se pretende que cada farola determine la posición de la farola que envía el mensaje y decidir si encenderse o no. Para ello, se compararán señales a partir de la función receiver_function():

```
void receiver_function(   //Función para recibir mensajes
  uint8_t *payload,
  uint16_t length,
  const PJON_Packet_Info &packet_info)  {

  if((char)payload[0] == 'M') {   //Si el mensaje que recibo es M, miro a ver quién lo envía
    Serial.println("Mensaje de movimiento");
    if (packet_info.header & PJON_TX_INFO_BIT) {
      int dispositivoRemitente = (int) packet_info.sender_id;   //Determino quién lo envía
      Serial.print("Dispositivo remitente: ");
      Serial.println(dispositivoRemitente);
      if (dispositivoRemitente < FAROLA) {                 //Si el que lo envía está detrás...
        Serial.println("¡Esa farola está antes de mí!");
        int distanciaEntreFarolas = FAROLA - dispositivoRemitente;    //Miro a ver si me tengo que encender
        if (distanciaEntreFarolas <= NUMERO_FAROLAS_ENCENDER) {
          Serial.println("¡Esa farola está muy cerca!");
          if (dia == false) {
            encenderLeds();
          }
        }
      } else if (dispositivoRemitente > FAROLA) {     //Si la farola está detrás mía
        if (FAROLA <= NUMERO_FAROLAS_ENCENDER) {      //Determino si estoy al principio
          int distanciaEntreFarolas = FAROLA + (NUMERO_FAROLAS - dispositivoRemitente);   //Miro si el que envía es el último
          if (distanciaEntreFarolas <= NUMERO_FAROLAS_ENCENDER) {   //Decido si encenderme o no
            Serial.println("Remote device is close to me!");
            if (dia == false) {
              encenderLeds();
            }
          }
        }
      }
    }
  }

}
```

Figura 5: Código para recibir mensajes

1.3.2. Prueba del sistema completo:

– **Código:** En el código, se escriben tanto varias variables, que se modifican según el entorno (¿es de día o de noche?, ¿se ha detectado movimiento o no?, etc.), y variables, que establecen valores fijos (la potencia lumínica que se identifica como luz tenue o luz intensa, número de farolas, etc.)

Este es el código:

```
    #include<PJON.h>       //Esta es la librería utilizada para la comunicación
    #include<SimpleTimer.h>  //Esta es la librería utilizada para el temporizador

    #define FAROLA 1             //Este es el número de dispositivo que corresponde a la
farola
    #define NUMERO_FAROLAS 2    //Este es el número de farolas que hay en el circuito
    #define NUMERO_FAROLAS_ENCENDER 1   //Este es el número de farolas que se deben
                                        //encender cuando una detecta movimiento
    #define TIEMPO_ENCENDIDO_FAROLAS 1000   //El tiempo que duran encendidas

    #define trigPin 4   //Pin trig
    #define echoPin 2   //Pin echo

    intvalorLDR = 0;//Valor que va a variar en función de la luz
    booldia = true; //Variable que establece si es de día o no
    intpinLDR = A7; //Pin de la LDR

    intledBajo = 25;//Luz tenue
    intledAlto = 255; //Luz alta
    intledApagado = 0;//Luz apagada

    boolmovimientoDetectado;//Variable que cambia en función de si se detecta movimiento

    SimpleTimerlampTimer;
    int Temporizador = -1;  //Esta variable la usamos para que si estuviera encendido el
                            //temporizador, cuando se activara la función de encender
luces otra
                            //vez, se reseteará el temporizador

    PJON<SoftwareBitBang> bus(FAROLA); //Pin del bus


    voidencenderLeds() {   //Función de encender leds y activar el temporizador
    Serial.println("Encender luces a tope");
    analogWrite(3, ledAlto);
    if (Temporizador >= 0) {
    lampTimer.deleteTimer(Temporizador);
      }
      Temporizador = lampTimer.setTimeout(TIEMPO_ENCENDIDO_FAROLAS, lampTimeout);
    }

    voidapagarLeds() {   //función de apagar la luz
    Serial.println("Apagar la luz");
    analogWrite(3, ledApagado);
    }

    voidluzTenue() {   //Función de atenuar la luz
    Serial.println("Atenuar luz");
    analogWrite(3, ledBajo);
    }

    voidlampTimeout() {    //Atenuar la luz después del temporizador
    luzTenue();
    }

    void setup() {

    Serial.begin(9600);

    pinMode(3, OUTPUT);
    digitalWrite(3, LOW);  //Apagar el Led

    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);

    bus.set_receiver(receiver_function);
    bus.strategy.set_pin(5); //Pin del bus
```

```
    bus.begin();

    }

    voidreceiver_function( //Función para recibir mensajes
    uint8_t *payload,
      uint16_t length,
    constPJON_Packet_Info&packet_info)  {

    if((char)payload[0] == 'M') {   //Si el mensaje que recibo es M, miro a ver quién lo
envía
    Serial.println("Mensaje de movimiento");
    if (packet_info.header& PJON_TX_INFO_BIT) {
    intdispositivoRemitente = (int) packet_info.sender_id;   //Determino quién lo envía
    Serial.print("Dispositivo remitente: ");
    Serial.println(dispositivoRemitente);
    if (dispositivoRemitente< FAROLA) {                      //Si el que lo envía está
delante...
    Serial.println("¡Esa farola está antes de mí!");
           //Miro a ver si me tengo que encender
    intdistanciaEntreFarolas = FAROLA - dispositivoRemitente;
    if (distanciaEntreFarolas<= NUMERO_FAROLAS_ENCENDER) {
    Serial.println("¡Esa farola está muy cerca!");
    if (dia == false) {
    encenderLeds();
              }
            }
          } else if (dispositivoRemitente> FAROLA) {
    //Como el circuito es circular, las farolas posteriores me pueden afectar
    if (FAROLA <= NUMERO_FAROLAS_ENCENDER) {
              //Determino si soy de los primeros
    intdistanciaEntreFarolas = FAROLA + (NUMERO_FAROLAS - dispositivoRemitente);
              //Miro si el que envía es uno de los últimos que me pueden afectar
    if (distanciaEntreFarolas<= NUMERO_FAROLAS_ENCENDER) {
               //Decido si encenderme o no
    Serial.println("¡Esta farola está muy cerca!");
    if (dia == false) {
    encenderLeds();
                }
              }
            }
          }
        }
      }

    }


    void loop() {

    lampTimer.run();
    bus.update();
    bus.receive(50000);

    booleraDia = dia;

    valorLDR = analogRead(pinLDR); //Compruebo si es de día o de noche
    dia = (valorLDR> 500);

    if (dia) {

        // Apagamos la luz
    apagarLeds();

      } else {

        // Seguimos la lógica del funcionamiento nocturno
```

```
    boolseDetectoMovimiento = movimientoDetectado;

        // Evaluamos la distancia
    long duracion, distancia;
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duracion = pulseIn(echoPin, HIGH);
        distancia = (duracion/2) / 29.1;

    if (distancia < 50) {     //Si el movimiento está en un rango menor que 50...
    Serial.print("distancia = ");
    Serial.println(distancia, DEC);
    movimientoDetectado = true; //Digo que hay movimiento y me enciendo
    } else{   //Si el movimiento no está en un rango menor que 50...
    movimientoDetectado = false; //Digo que no hay movimiento y no me enciendo
        }

    if (movimientoDetectado) {
    encenderLeds();
        } else {
    if (eraDia) {
    luzTenue();
          }
        }

    // Evaluamos el envío del mensaje
    if ((!seDetectoMovimiento) && (movimientoDetectado))  {
    //Si he pasado de no detectar a detectar...
    Serial.println("Enviando mensaje...");
    bus.send(PJON_BROADCAST, "M", 1);     //Envío el mensaje
        }

      }

    }
```

– **Circuito:** el circuito consta de dos partes: la base (Arduino Nano + sensor ultrasonidos + el conector del bus) y la parte superior de la farola (LDR + LED's). El esquema del circuito es este:

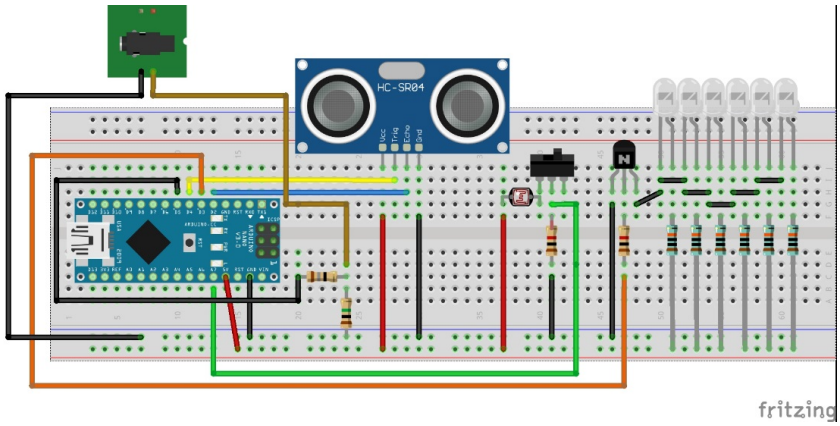

Figura 6: Circuito final

Del circuito salen el cable del bus (pin digital 5), y el cable del GND (para compartir tierras). Los pines del sensor de ultrasonidos son: ECHO=2 y TRIG=4. La resistencia LDR está conectada al A7, y, además, está conectado a un interruptor (como en el dibujo) para que en un sentido dé un valor analógico de 0 (para hacer pruebas con luz, pero la placa Arduino lo interprete como periodo nocturno) y en el otro sentido, dé el valor que reciba la resistencia LDR (en el prototipo funcional). Los LED's son conectados al transistor 2N3904 para amplificar la señal (como se ilustra en la representación superior a este texto), vinculado al pin digital 3.

El circuito con este programa ha sido probado en placas de prototipado:

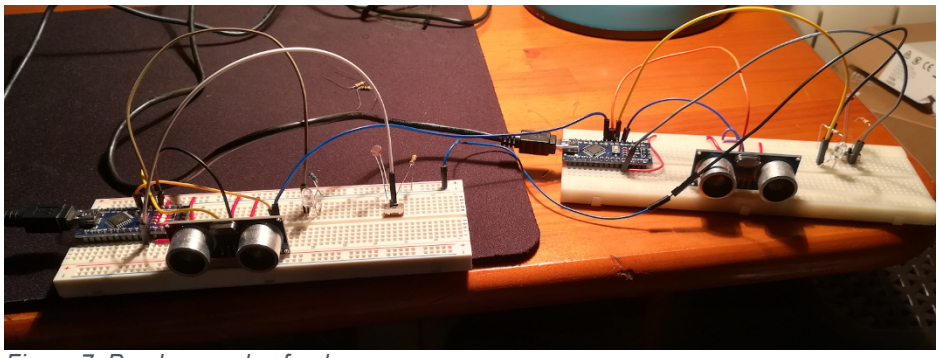

Figura 7: Prueba con dos farolas

En el siguiente enlace, se muestra una grabación realizando esta prueba: https://youtu.be/bCMzwBK8Bv4

1.3.3. Prueba con seis circuitos y un *scalextric*

Tras comprobar que todo funcionaba, se probó a hacer el primer prototipo. Este consiste en conectar seis farolas prototipo en un scalextric. Se comprobó que el sistema de alumbrado reacciona sensiblemente bien al movimiento del automóvil. Puede ser comprobado en el siguiente enlace: https://youtu.be/uf8TS4t5Kbk

1.3.4. Soldadura

Para la soldadura, se utilizaron placas PCB: una para la base, y otra para la parte superior. Se obtuvo el siguiente resultado:

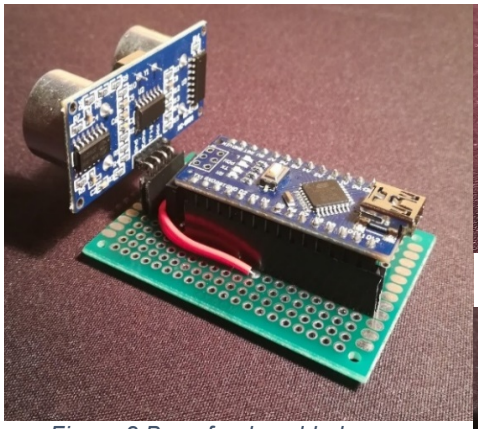

Figura 8:Base farola soldada

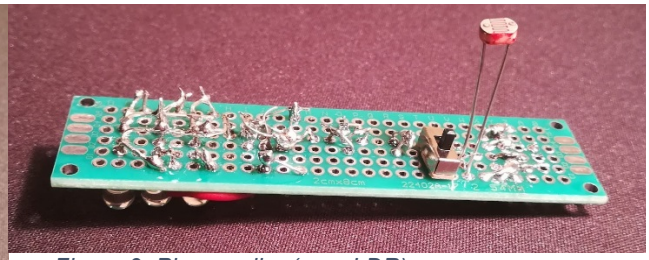

Figura 9: Placa arriba (cara LDR)

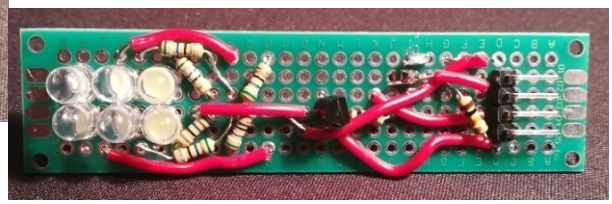

Figura 10: Placa arriba (cara LED's)

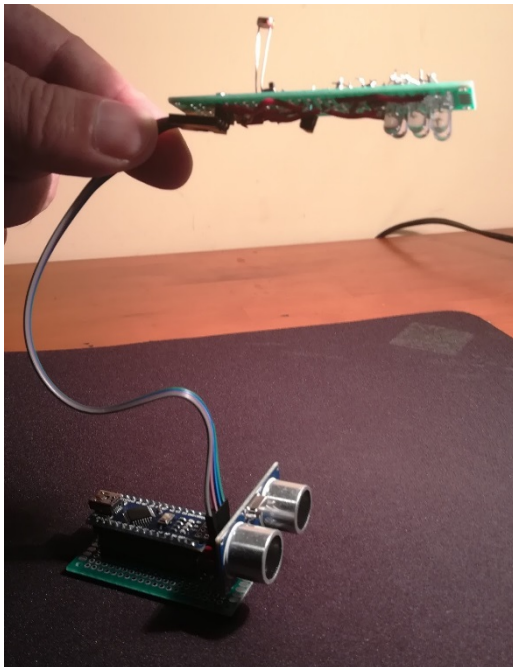

Figura 11: Circuito farola completo

1.3.5. Diseño de la carcasa 3D

La carcasa que recubre el circuito, se diseñó con el software *FreeCad*. La elaboración del boceto 3D se realizó ensamblando y recortando diversas formas geométricas (cubos, cilindros, etc.). Se presentaron algunas dificultades en cuanto a las dimensiones idóneas, pero fueron solventadas con éxito.

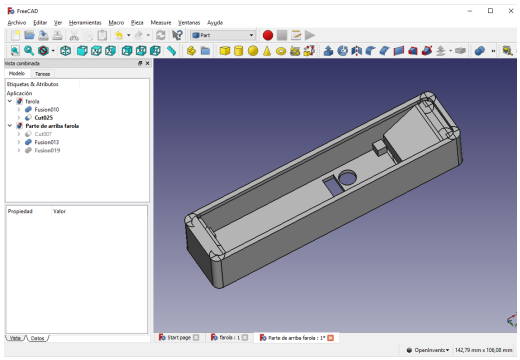

Figura 11: Tapa farola parte de arriba

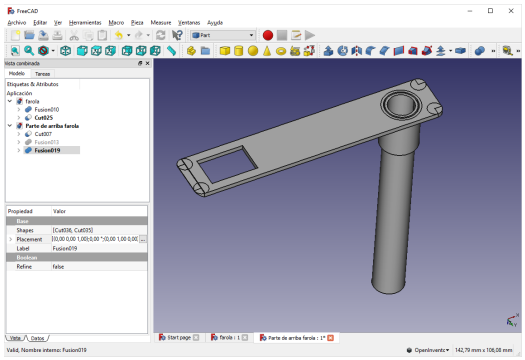

Figura 12: Base + tubo arriba farola

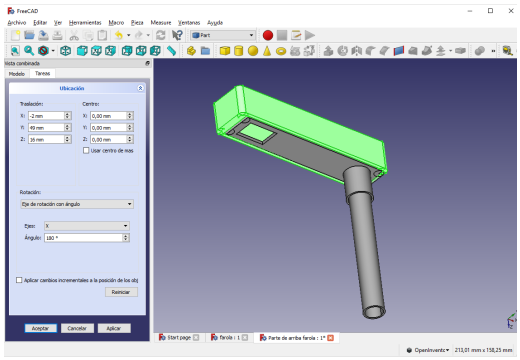

Figura 13: Parte arriba entera

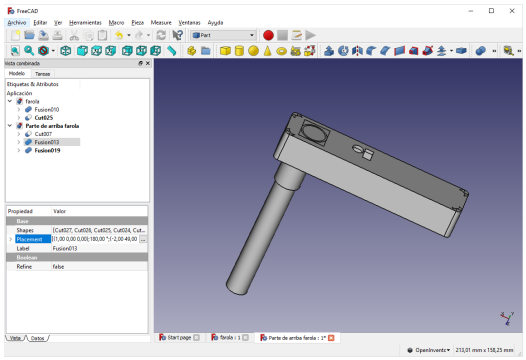

Figura 14: Parte arriba entera

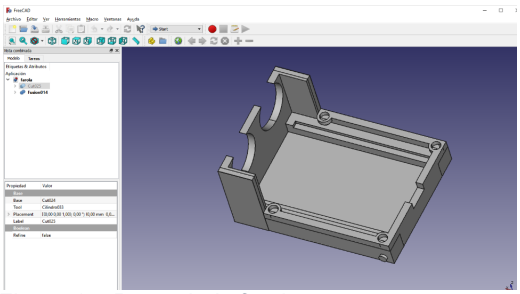

Figura 15: Base abajo farola

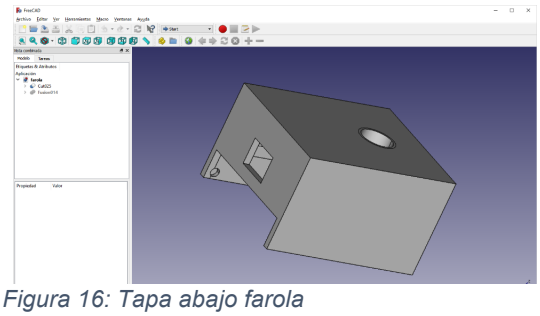

Figura 16: Tapa abajo farola

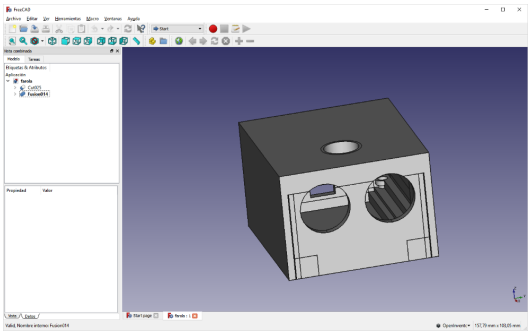

Figura 17: Parte abajo farola completa

Se exportaron los diseños en archivos stl para que en el programa Cura se pudieran archivar en una tarjeta SD. Este programa permite la impresión de los diseños estimando los parámetros de relleno de la pieza, así como las condiciones de impresión.

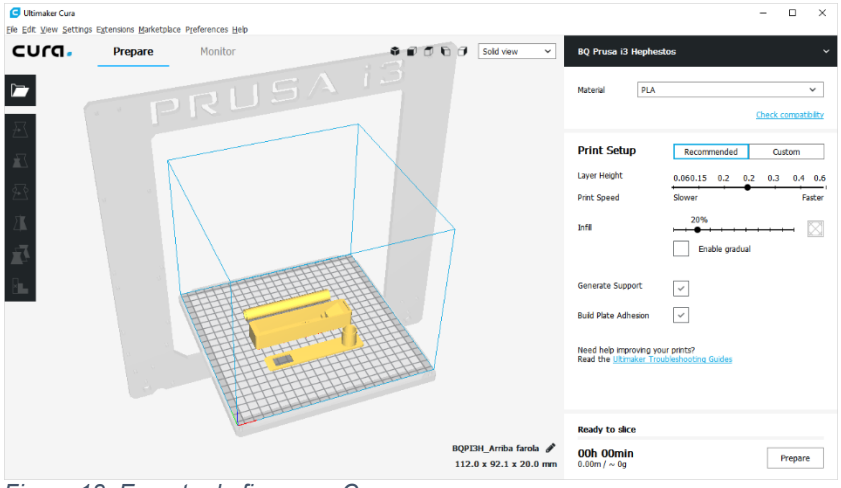

Figura 18: Exportar la figura en Cura

Durante la impresión de las piezas se experimentaron algunos fallos de calibración (imprecisiones de las medidas y ajuste de las tolerancias de la impresora). A continuación, se muestran diversas fotografías del resultado final de las piezas.

Figura 19-24: Farola impresa y con circuitos montada entera

1.4 Resultados

Para finalizar el proyecto, se han verificado los seis circuitos soldados, en carcasas, y en un scalextric. Estos fueron los resultados:

https://www.youtube.com/watch?v=x4S253bkEN8
https://www.youtube.com/watch?v=UcHtQuG9_E4

En los vídeos, se observa que las farolas no se activan con el movimiento del automóvil, si el sistema presenta condiciones de luz ambiente (día). En cambio, en periodo nocturno, se encienden todas a la vez en modo tenue. Cuando pasa el coche, se enciende la farola posterior a ella para que así el coche tenga la luz suficiente para circular por la autovía.

Sin embargo, no todo ha sido tan fácil. Se han presentado una serie de cuestiones conflictivas durante la elaboración del proyecto:

- Se ha apreciado que, si son enfrentadas dos farolas, las ondas de ultrasonidos interfieren entre ellas. Por ello, ha sido colocadas apuntando al exterior. Una solución a este problema para carreteras de doble sentido sería controlar el momento en que cada farola emite las ondas con el sensor de ultrasonidos y así, hacer que nunca coincidan las farolas enfrentadas a la hora de detectar movimiento.

- Las soldaduras han sido notablemente complejas. No ha sido nada grato las repeticiones necesarias por los equívocos cometidos durante el proceso mismo.

- En cuanto al diseño 3D, hemos tenido fallos en medidas que debían de ser muy exactas, y, sobre todo, en el adecuado control de las tolerancias de la impresora para la reproducibilidad del diseño.

- Por último, en ocasiones el bus que conecta todas las farolas se colapsa. Alguna farola no recibe bien el mensaje y no se enciende.

1.5 Conclusión

Este proyecto que hemos realizado es un modelo de pruebas. Lo que pretendemos demostrar con el mismo es que es posible y viable realizar cambios en los sistemas de iluminación en vías urbanas que permitan un ahorro energético. Es un proyecto en fase beta, pero con una motivación clara para ser trasladable a la realidad.

2. Figuras

3. Referencias

- **BBC News:** https://www.bbc.com/mundo/noticias/2011/05/110513_verde_recursos_naturales_lh
- **Código para el ultrasonidos:** https://www.instructables.com/id/Simple-Project-With-the-Ultrasonic-Sensor-HC-SR04-/
- **Cura:** https://ultimaker.com/en/products/ultimaker-cura-software
- **FreeCad:** https://www.freecadweb.org/
- **Fritzing:** http://fritzing.org/home/
- **I2C:**https://aprendiendoarduino.wordpress.com/2017/07/09/i2c/ https://www.luisllamas.es/arduino-i2c/
- **Iluminar varios LED's:** https://forum.arduino.cc/index.php?topic=542073.0
- **LDR's:** https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/
- **PJON:**https://www.pjon.org/ https://www.pjon.org/SoftwareBitBang.php
- **PWM:** https://playground.arduino.cc/ArduinoNotebookTraduccion/Appendix3
- **Simple Timer:** https://playground.arduino.cc/Code/SimpleTimer/