Asociación Cultural Española de Aficionados a los Mecanos

ACEAM

AUTOMOVIL FORD T (1910). Antonio Valero Aicua

Boletín nº 37

Junio 2021

Editorial

Este es el tercer Boletín de ACEAM que se publica durante la pandemia y comprobamos que la Covid no ha podido con nuestra afición. Durante este tiempo hemos construido nuevos modelos, y si bien hemos tenido que suspender las animadas tertulias que manteníamos dos tardes al mes en la cafetería VIPS de la calle Alcalá de Madrid se ha ampliado notablemente el número de tertulianos gracias al grupo de WhatsApp de ACEAM, que aunque virtualmente comentamos nuestros modelos y nos enviamos fotos y videos de los mismos, así como cualquier anuncio que encontramos en internet de piezas y equipos interesantes, entre otras noticias y curiosidades de nuestra afición.

De nuevo invitamos a unirse a ese grupo de WhatsApp a todos los socios que no lo han hecho aún, que a buen seguro aumentaran el interés del mismo.

Seguimos, también, ampliando el contenido de nuestra web www.aceam.org , a la que se han subido los 23 primeros números de nuestro boletín y otros documentos, así como fotografías de modelos de diversos socios.

Estamos tratando de organizar una nueva exposición Meccano para cuando sea posible celebrarla en normalidad sanitaria. Esperamos de los socios colaboración para conseguir financiación y localización de lugares para la exhibición.

Antonio Valero Aicua
Presidente de ACEAM

Contenido

Página

Tres modelos de Fischertechnik Oeco-Energy

Esteban Orozco Vallejo

Introducción

La caja Fischertechnik Profi Oeco Energy trata de energías renovables y permite construir 14 modelos agrupados en tres capítulos : Solar Power, Water Power y Wind Power. Hay también una pequeña caja aneja Fischertechnik Profi Fuel Cell (célula de combustible) que permite producir electricidad. Esta caja Oeco Energy, salida al mercado hace varios años, tiene varios premios internacionales y es utilizada en centros de enseñanza alemanes. Estas dos cajas proporcionan un apreciable conocimiento básico y práctico de las energías renovables.

Describiremos a continuación tres de los modelos que hemos construido : Seguidor solar, Coche solar y Coche eléctrico._

Seguidor solar

Este modelo es muy interesante, pues permite medir el ángulo en horizontal del recorrido del sol en las horas de luz de un día determinado (diferencia de azimuts). Es en sí también un mecanismo para maximizar la producción de energía por el efecto fotovoltaico de un panel solar. La construcción de este modelo ha permitido familiarizarnos con diversos aspectos de un sistema de paneles solares.

Hablaremos primero del entorno astronómico definiendo las palabras cenit y azimut. El cenit es la intersección de la vertical del punto en el que está el observador y la esfera celeste. El azimut es un ángulo en el plano tangente a la esfera terrestre en el punto del observador y se mide a partir de la línea de dicho plano observador- norte en el sentido de las agujas del reloj CW (clock wise). Este ángulo azimutal variable está comprendido entre la línea del norte y la línea intersección del citado plano tangente a la esfera terrestre en el plano cenit-observador-astro. El seguidor solar recorrerá la diferencia entre los ángulos azimutales observador-amanecer (alborada) y observador-puesta del sol (ocaso). Véase la figura 1.

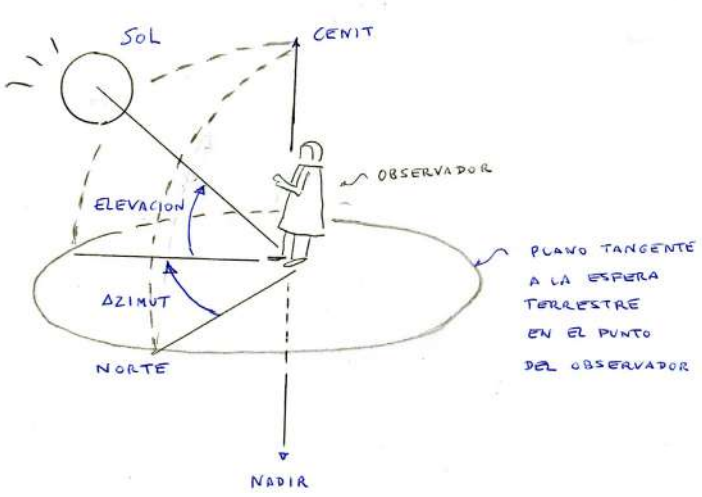


Figura 1

El seguidor solar es también un mecanismo para maximizar (producciçon de energía) el efecto fotovoltaico de un panel solar a lo largo de las horas de sol de un día. El modelo que nos ocupa es uno de los principales tipos de seguidores, llamado “de eje azimutal”, aunque en realidad el panel o paneles solares giran alrededor de un eje vertical observador-cenit. Véase el apartado de Wikipedia “Seguidor solar”.

Describiremos ahora brevemente el modelo. Se utilizan dos paneles solares. Los puntos en que convergen ambos indican siempre la posición del sol, es decir el seguidor gira alrededor de un eje vertical siguiendo el movimiento del sol.

Los dos paneles están conectados de modo anti-paralelo, es decir se une el polo positivo de un módulo solar con el polo negativo del otro módulo solar. Cuando ambos módulos se iluminan con el sol con la misma intensidad se anulan las tensiones generadas y no llega voltaje al motor. Cuando el sol se traslada de posición un módulo se ilumina con mayor intensidad y llega al motor una tensión positiva o negativa. La consecuencia es que el motor gira hasta que la luz incida nuevamente en el plano de separación de ambos módulos. Los paneles del modelo vienen con una inclinación determinada. Evidentemente habría que estudiar la inclinación óptima, que variará según el lugar. Suponemos que la escogida en el modelo está ajustada a algún lugar determinado de Alemania. En las figuras 2 y 3 están dos fotos del modelo. En la figura 3 está incluida un linterna con la cual también funciona el modelo.

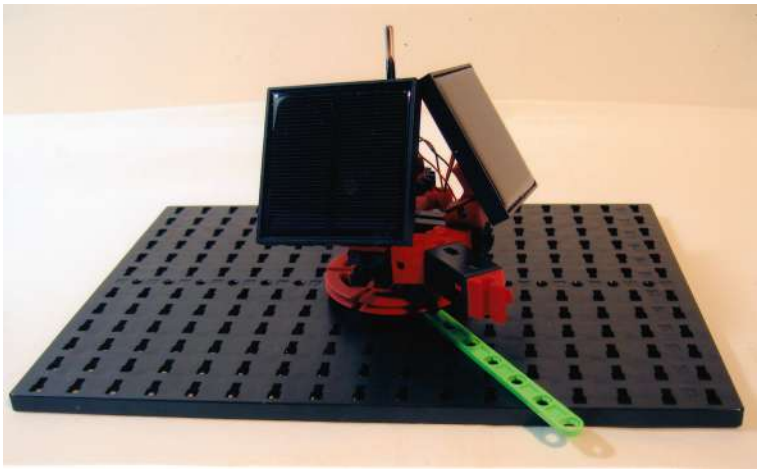


Figura 2

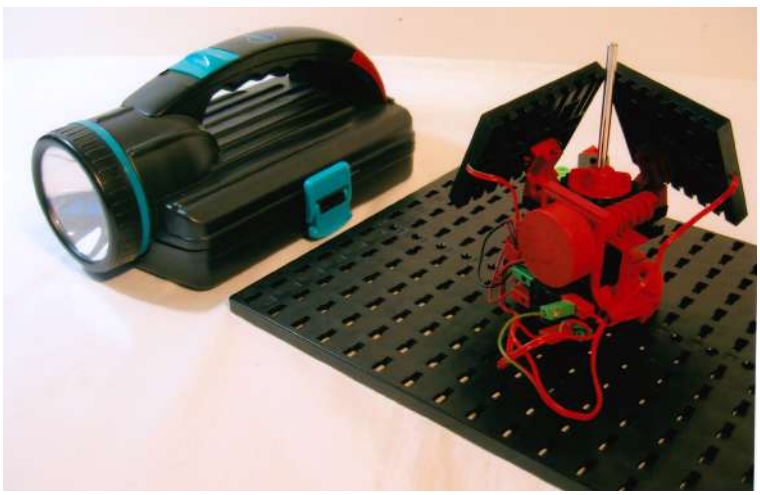


Figura 3

Hablemos ahora del funcionamiento. El modelo funciona perfectamente con el sol, con un foco y con una linterna. Es curioso que no funcione con la luz de un móvil o tableta. Suponemos que la luz que emite el móvil o tableta es LED, lo cual no debe ser compatible o asimilable a un espectro de la luz solar o de la luz del filamento de una bombilla normal.

Figura 4

Colocamos por la mañana el modelo sobre un trípode al lado de una ventana de la fachada interior de mi casa y colocamos una tira verde horizontal hacia la dirección en que sabemos sale el sol. Hay unas horas centrales del día en la que los rayos de sol inciden en la parte sur de nuestra casa y por tanto el modelo no funciona. Pasadas estas horas trasladamos el modelo a una ventana de la fachada exterior (hacia la calle) y el sol sigue iluminando el modelo hasta que se oculta, poniendo ahora hacia el punto del ocaso la citada tira verde horizontal. Recortamos en un papel el sector circular entre los dos citados marcadores (posición inicial y final de la tira verde). Poniendo en Internet "Horas de sol del día" nos dice el ángulo azimutal que puede recorrer el seguidor al día, unos 15º por hora de sol, lo que comprobamos durante varios días. Por ejemplo el 21-11-19 nos dice Internet que el día es de 9,75 horas, lo que da 146,º3, que coincide con el sector de papel indicado, aumentando en él de modo aproximado lo relativo a la altura de los obstáculos a la línea de horizonte. Véase la figura 4.

Cuando está nublado el seguidor se para o incluso retrocede si la luz de la atmósfera es mayor en un punto atrasado (ya hemos dicho que el sentido del giro del sol es el de las agujas del reloj CW). Pero cuando vuelve a salir el sol el seguidor se recupera de una vez y se pone en posición correcta. Se comprueba que hay un determinado umbral en el que, si está muy nublado, el seguidor no gira en absoluto, lo cual nos ilustra de la escasa capacidad de la generación de energía a lo largo del año.

Vamos ahora a los comentarios de tipo energético. Vemos en una revista las horas promedio de funcionamiento al año en España de los diversos modos (mix energético) de producción de energía eléctrica. Frente a las 8.760 horas/año (365x24) en 2019 la energía nuclear ha trabajado 8.700 horas, la eólica marina 5.000 horas, la eólica terrestre 3.500 horas, la fotovoltaica y termo-solar 2.000 horas, siendo la contribución de estas últimas un 3,5% de la energía generada total anual. Añadiremos que, también en el año 2019, el 21% de la energía eléctrica producida ha sido nuclear. Le siguen con un 20% la eólica y el gas natural,

también con un 20%. Siendo el carbón un 10% y la hidráulica otro 10% y en el resto están la cogeneración y otros y la solar con el 3,5% que ya hemos dicho. Estos contactos nuestros en estos meses con la energía fotovoltaica nos ha permitido constatar la escasa disponibilidad de ella, incluso en un país favorable como España. Por lo que deducimos que esta energía es inoperante en la mitad norte de Europa, con muchos meses al año de días cortos en invierno y encima nublados. Ha habido una iniciativa europea para hacer un gigantesco parque solar en el Sahara y traer la energía a Europa pero, tras cuantiosos gastos, se ha abandonado por fin este proyecto.

Como comentario final indico que este sencillo modelo nos ha permitido refrescar determinados conceptos astronómicos y comprobar directamente algunos comportamientos de los sistemas de paneles solares, como efectos de las nubes, umbrales de producción de energía, horas útiles al año, escaso voltaje generado, etc.

Vehículo solar

Se trata de un vehículo accionado simplemente por unos paneles solares que lleva.

Figura 5

Con los dos paneles solares unidos en serie y el motor especial de bajo voltaje el modelo funciona perfectamente. Cada panel emite en corriente rectificada 1 V / 440 mA, con lo cual el conjunto de los dos paneles en serie emite a 2 V / 440 mA.

El vehículo tiene un interruptor, basado en un mini-pulsador, para pararle o ponerle en marcha. También un volante para el cambio de dirección. Está fijada la rueda derecha y no la izquierda. ¿Es esto un diferencial rudimentario o camuflado?. Puesto que el vehículo no tiene control remoto y por tanto no es controlable se me ocurre lo siguiente.
Poniendo la dirección de las ruedas en forma oblicua y atando con un cordón metálico el volante para inmobilizarle se consigue que el coche gire en una circunferencia de unos 60 cm de diámetro. En la noche pararía pero al día siguiente volvería a andar ¡Sería un movimiento casi perpetuo!

En casa, aún con las ventanas abiertas, la trayectoria circular del coche entra en zonas de sombra, con lo cual se para. Bajo al parque que hay detrás de casa y funciona perfectamente sin interrupción. Véase figura 6.

Figura 6

Es evidente que este tipo de vehículo no puede desarrollarse. Sí existen aviones que funcionan con paneles solares, pero vuelan por encima de las nubes y tienen una gran superficie para soportar paneles. Ni vehículo ni aviones podrían funcionar por las noches.

Punto de carga y coche eléctrico

Este modelo tiene dos partes, la primera el punto de carga y la segunda el coche eléctrico.

Punto de carga

El punto de carga utiliza tres paneles solares. Se unen en un circuito anti-paralelo. El polo positivo de un panel se conecta con el polo negativo de otro. En este punto de carga los dos hilos finales del circuito anti-paralelo se conectan a un LED verde y de ahí continúan los dos cables-manguera para conectar con el vehículo y cargarle. Cuando el condensador del coche, del que hablaremos más adelante, esté totalmente cargado se enciende el citado LED verde.

Coche eléctrico

Sobre el modelo de coche solar quitamos los dos paneles y en su lugar ponemos un elemento de carrocería. Instalamos el condensador rápido (la batería del coche), el elemento Goldcap. Este es una pequeña batería recargable que consta de dos elementos de carbón activados, separados entre sí sólo por una fina capa aislante. Su ficha completa es “Goldcap purixell 3,0 V, 10 F, EC(N)”. Hay que conectarle a 3 V cc o menos (en caso contrario hay peligro de explosión). Su capacidad es de 10 F, bastante elevada. Es rápido para cargar el coche

eléctrico. La patilla larga (+) es para enchufarle el pin rojo y la corta (-) para enchufarle el pin verde.

Una vez cargado el coche eléctrico (LED verde encendido) y desenchufado se oprime un pulsador y el coche se pone en marcha. Como antes, está enclavado el volante de dirección y el coche gira en una circunferencia de unos 60 cm de diámetro. También hemos mantenido el citado diferencial encubierto. La carga es rápida y da para un cierto número de vueltas. La ventaja respecto al coche solar es que se puede hacer circular en una habitación sin que le perturben las sombras de los marcos de las ventanas.

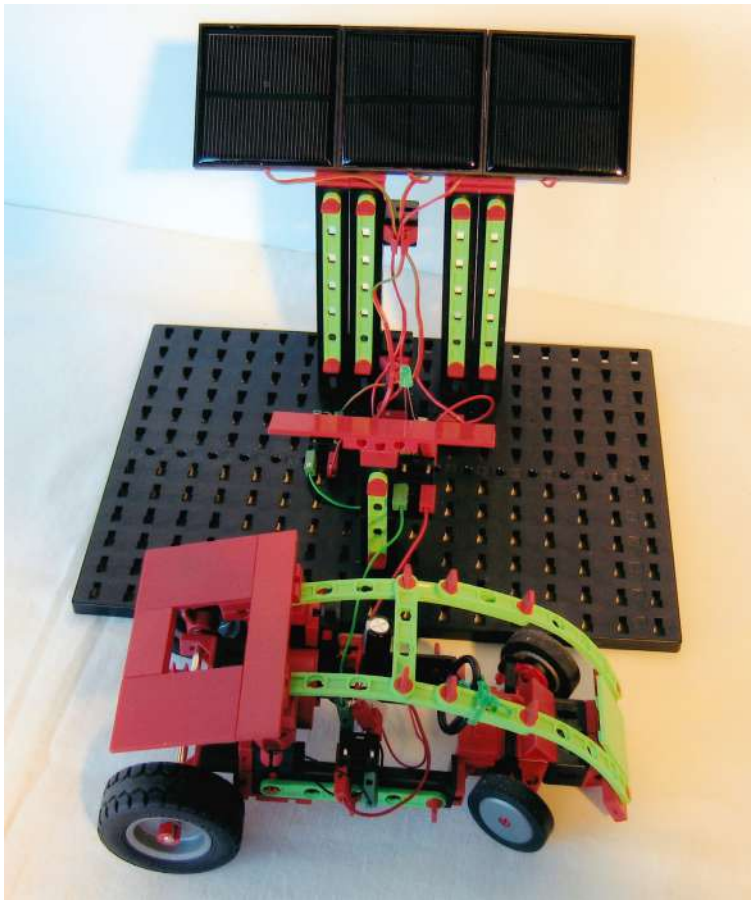


Figura 7

En el modelo aparecen varias de las características de un coche eléctrico. Las electro-mangueras, la batería de carga rápida y la no muy grande autonomía (el número de vueltas que permite cada carga es reducido). El punto de carga no es real (no puede ser un conjunto de paneles). Es simplemente una indicación de que la electricidad debe tener un origen renovable.

Nota final

En la caja Oeco Energy hay también otro coche que lleva aneja una célula de combustible, con sus depósitos de hidrógeno y oxígeno. No le he construido porque al girar el coche puede chocar con algo y hay que tener en cuenta la peligrosidad del hidrógeno.

Es notable ver cómo con estos sencillos modelos y experimentos se avanza en la comprensión de diversos temas, sus aspectos prácticos, características generales, tecnologías aplicables, ventajas e inconvenientes, etc.

Si los juegos de construcción son en general la ingeniería en miniatura esta caja de Fischertechnik es una ingeniería de muy buen diseño y de muy buena calidad.

Joysticks

Jesús Alonso Rodríguez

En el boletín nº 35 de junio de 2020, vimos que un potenciómetro puede ser utilizado como codificador rotativo absoluto. Ahora veremos un tipo de codificador rotativo muy particular que utiliza dos potenciómetros con muy pocos grados de giro.

Si tenemos una palanca que acoplamos a dos potenciómetros, uno que se mueva con el movimiento vertical de la palanca y otro que se mueva con el movimiento horizontal, tendremos lo que se denomina un "Joystick" (pronunciado: "yoistic"). Afortunadamente, no tenemos que rompernos la cabeza pensando una forma de acoplar la palanca a los dos potenciómetros porque ya existe un componente electrónico que lo hace todo; además incorpora unos potenciómetros especiales con muy pocos grados de giro e incluso un pequeño pulsador que se activa cuando presionemos la palanca. En la Figura 1 vemos un joystick de los que se pueden comprar en Amazon o en cualquier tienda china de Internet, por 3 o 4 Euros:

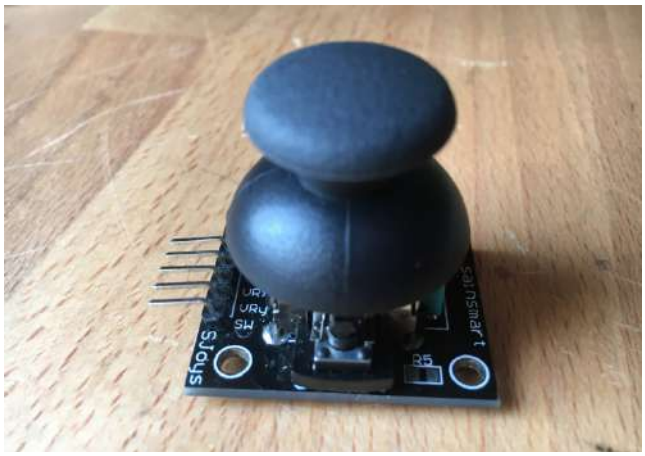


Figura 1

En la Figura 2 vemos el mismo componente con la caperuza quitada. Se puede sustituir la caperuza por un empalmador de ejes de Meccano con un soporte de balaustrada arriba, para hacer un joystick totalmente "meccanero":

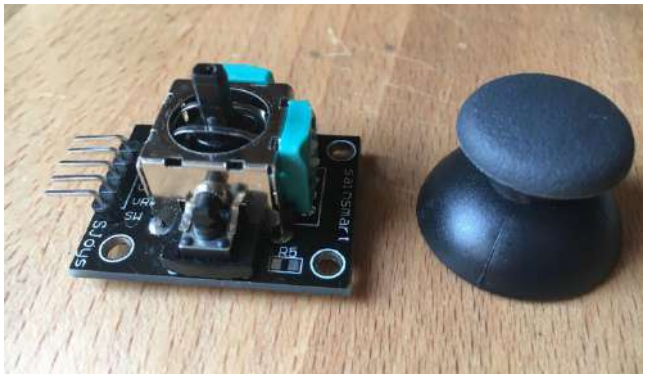


Figura 2

Se trata del tipo de joystick que solemos encontrar en los mandos de las consolas de videojuegos. Vienen montados sobre una plaquita de circuito impreso con cuatro agujeros de 3mm de diámetro para poderla fijar y con cinco puntos de conexión:

- GND : Negativo de la alimentación (masa).
- +5v : Positivo de la alimentación.
- VRx : Pata central del potenciómetro del eje X.
- VRy : Pata central del potenciómetro del eje Y
- SW : Pulsador que conecta con GND al presionar el eje.

Quien haya adquirido el kit de iniciación de Arduino, probablemente se haya encontrado con uno de estos dispositivos. En la Figura 3 vemos el esquema interno de un joystick de este tipo y, a la derecha, el símbolo con el que lo representaremos en nuestros esquemas:

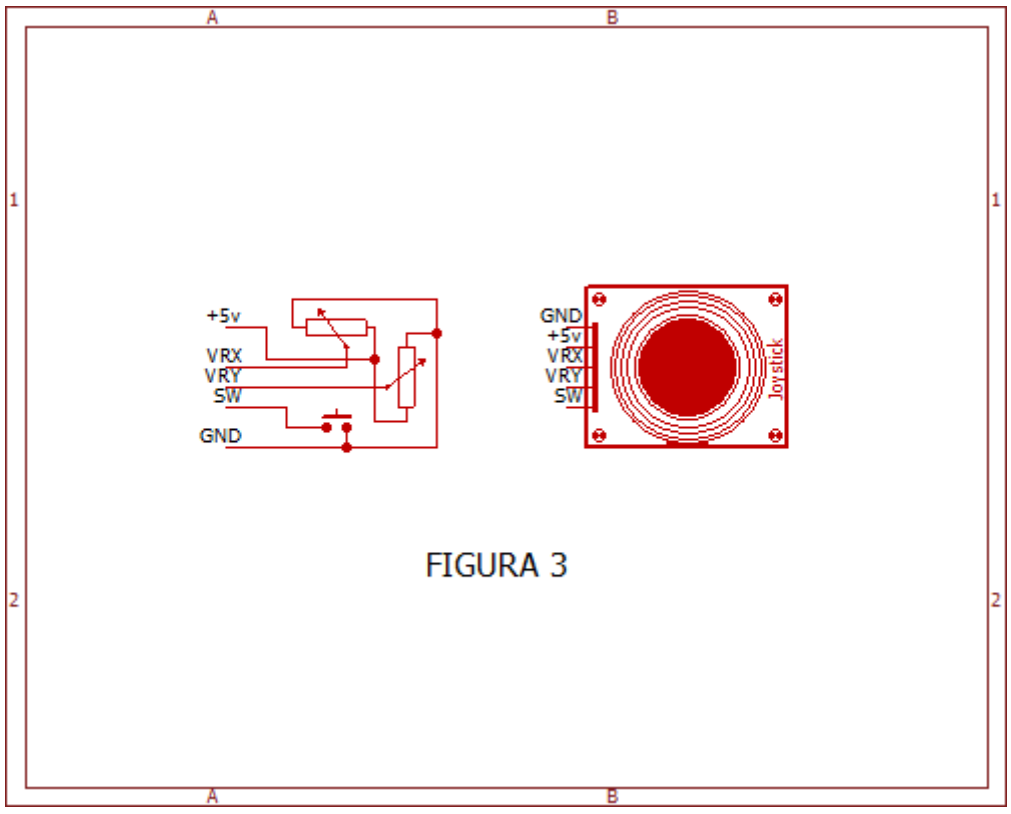


FIGURA 3

Como vemos, el potenciómetro correspondiente al eje X estará en GND (0 voltios) cuando el mando esté a la izquierda y en +5v (5 voltios) cuando el mando esté a la derecha. Igualmente, el potenciómetro correspondiente al eje Y estará en GND (0 voltios) cuando el mando esté arriba (adelante si el joystick está en horizontal) y en +5v (5 voltios) cuando el mando esté abajo (atrás). Estas referencias son válidas si colocamos el joystick como se ve en la Figura 3, es decir, con las conexiones a la izquierda, pero también podemos rotarlo 90 grados a la izquierda y que las conexiones queden para abajo. En ese caso, los ejes X e Y se intercambian y la tensión entregada será mínima (0v.) a la izquierda y abajo y máxima (5v.) a la derecha y arriba.

En la Figura 4, vemos cómo podemos conectar un joystick al Arduino:

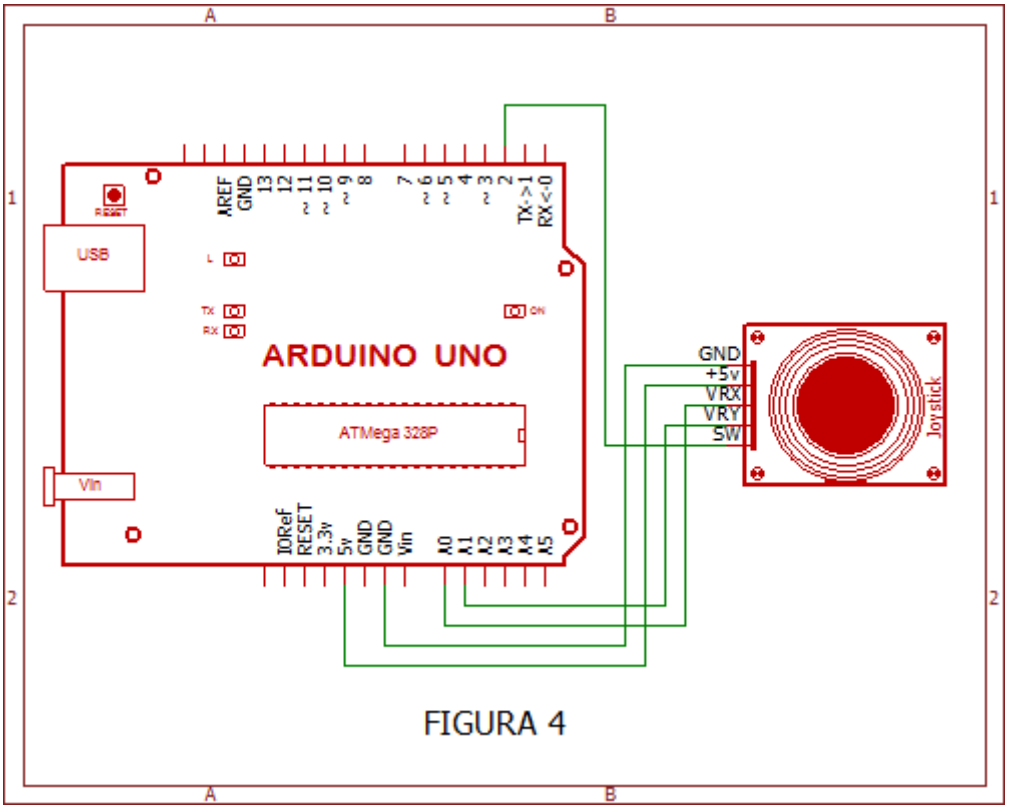


FIGURA 4

Como vemos, conectamos la pata +5v al pin 5v del Arduino, la pata GND a GND, las patas VRx y VRy a dos entradas analógicas; por ejemplo, VRx a A0 y VRy a A1; finalmente, conectamos la pata SW a cualquier pin digital; por ejemplo, el pin 2. Ahora vamos a ver cómo leemos la posición de joystick dentro del programa. Primero, en "void setup()", inicializamos el pin digital como entrada pero con una resistencia a positivo, para que cuando el pulsador no esté pulsado, nos lea una entrada alta:

```
pinMode(2,INPUT_PULLUP);
```

Las entradas analógicas no es necesario inicializarlas. Para leer el eje X haremos:

```
unsigned int ejeX=analogRead(A0);
```

Y para leer el eje Y:

```
unsigned int ejeY=analogRead(A1);
```

Hemos metido los valores en una variable tipo "unsigned int" (16 bits sin signo) porque el valor que nos devolverá analogRead() será un número comprendido entre 0 y 1023; concretamente, "0" para el mando a la izquierda o arriba y "1023" para el mando a la derecha o abajo. Si no tocamos el mando (el joystick tiene un muelle que lo lleva al centro), leeremos un valor de, aproximadamente, "512" en ambos ejes. He puesto "aproximadamente" porque debido a pequeñas imprecisiones, estos números pueden oscilar en uno o dos valores de diferencia entre distintos joysticks.
Para leer el pulsador haremos:

```
boolean pulsador=digitalRead(2);
```

Que nos devolverá un valor "HIGH" si no está pulsado, y un valor "LOW" si lo está. Una pequeña observación para los que se estén iniciando en programación: En el lenguaje "C"

que utiliza Arduino, “HIGH” es lo mismo que “true” y “LOW” es lo mismo que “false”; tan solo son constantes predefinidas para comodidad del programador. “HIGH” y “true” en realidad son “1” y “LOW” y “false” en realidad son “0”. El compilador de “C” considera “true” cualquier cosa que no sea cero. Si hacemos:

```
digitalWrite(2,HIGH);
```

o si hacemos:

```
digitalWrite(2,1);
```

es exactamente lo mismo y funciona exactamente igual, pero la primera forma es más clara y más fácil de leer.

Manejando dos motores

Y ahora, vamos a ver qué podemos hacer con esto. Ya vimos que con un doble puente en “H” podíamos manejar dos motores. Vamos a utilizar el módulo con un L298 que vimos en el boletín de junio de 2020 (nº 35):

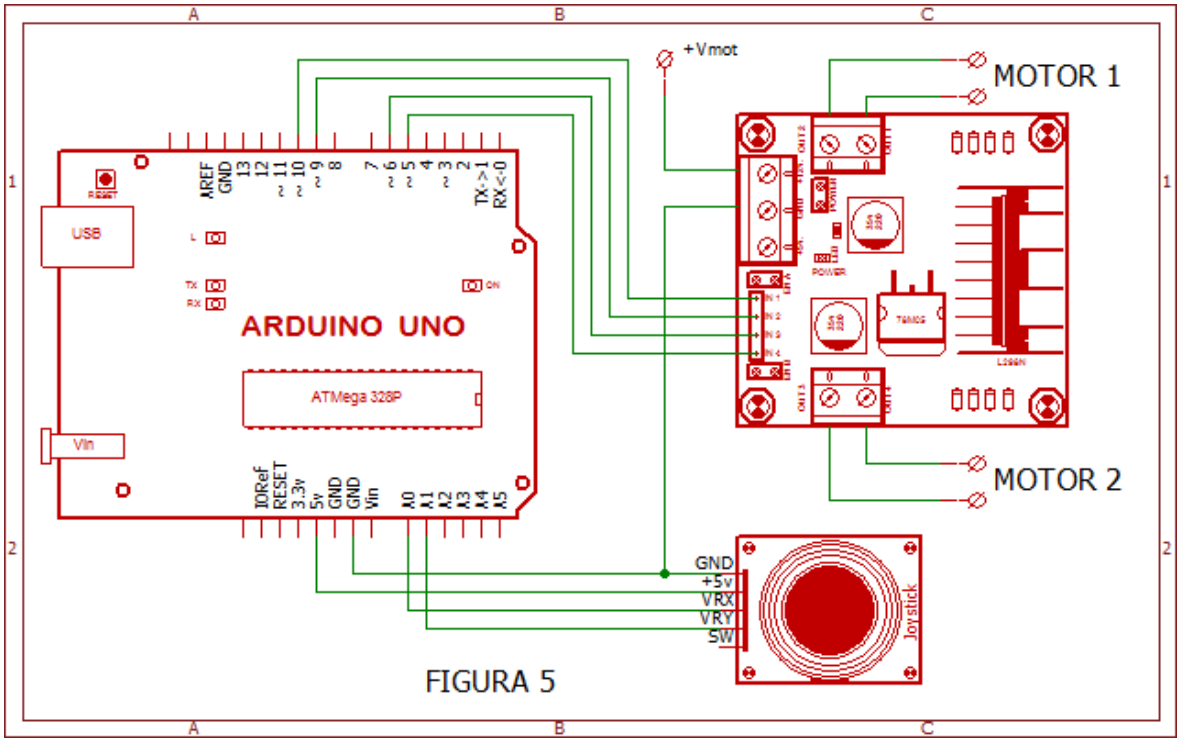


FIGURA 5

De momento, prescindimos del pulsador. Vamos a controlar dos motores con el joystick, de manera que uno sea controlado con el movimiento en el eje X y el otro con el movimiento en el eje Y. Cada motor estará parado cuando el joystick esté en el centro y se moverá en uno u otro sentido según movamos el joystick. En ambos casos, la velocidad de cada motor será proporcional a lo alejado que esté el joystick de su posición central.

En la entrada marcada “+Vmot” meteremos el positivo de la fuente de alimentación (o batería) que nos suministre el voltaje para los motores (9 voltios, 12 voltios, etc.) No olvidemos que el negativo de esta fuente también tiene que ir conectado a la masa del Arduino. El programa que tendremos que cargar en el Arduino será el siguiente:

```
// Mueve dos motores siguiendo a un joystick.

// CONSTANTES:
#define EJE_X A0 // Joystick eje X.
#define EJE_Y A1 // Joystick eje Y.
#define MOTOR_1_AV 10 // Avance motor 1.
#define MOTOR_1_RE 9 // Retroceso motor 1.
#define MOTOR_2_AV 6 // Avance motor 2.
#define MOTOR_2_RE 5 // Retroceso motor 2.
// FIN DE CONSTANTES.

// VARIABLES GLOBALES:
unsigned int ejeX; // Lectura del joystick.
unsigned int ejeY;
byte veloc1; // Velocidad de motores.
byte veloc2;
byte sentido1; // Sentido giro motores.
byte sentido2;
// FIN DE VARIABLES GLOBALES.

// CÓDIGO:
void setup() {
  pinMode(MOTOR_1_AV,OUTPUT);
  pinMode(MOTOR_1_RE,OUTPUT);
  pinMode(MOTOR_2_AV,OUTPUT);
  pinMode(MOTOR_2_RE,OUTPUT);
}

void loop() {
  // Lee el joystick:
  ejeX=analogRead(EJE_X);
  ejeY=analogRead(EJE_Y);

  // Calcula velocidades y sentidos:
  if(ejeX<500){
    sentido1=MOTOR_1_RE;
    veloc1=map(ejeX,499,0,0,255);
  }
  else if(ejeX>524){
    sentido1=MOTOR_1_AV;
    veloc1=map(ejeX,525,1023,0,255);
  }
  else{
    sentido1=0;
  }

  if(ejeY<500){
    sentido2=MOTOR_2_AV;
    veloc2=map(ejeY,499,0,0,255);
  }
  else if(ejeY>524){
    sentido2=MOTOR_2_RE;
    veloc2=map(ejeY,525,1023,0,255);
  }
  else{
    sentido2=0;
  }
```

```
  // Mueve los motores:
  if(sentido1!=0){
    analogWrite(sentido1,veloc1);
  }
  else{
    analogWrite(MOTOR_1_AV,0);
    analogWrite(MOTOR_1_RE,0);
  }

  if(sentido2!=0){
    analogWrite(sentido2,veloc2);
  }
  else{
    analogWrite(MOTOR_2_AV,0);
    analogWrite(MOTOR_2_RE,0);
  }

  // Espera 0,1 segundos:
  delay(100);
}
// FIN DE CÓDIGO.

// Fin de programa.
```

Recordemos que todas las líneas que empiezan por "//" son comentarios; se pueden quitar y el programa funcionará igual, pero recomiendo encarecidamente teclearlas, ya que ayudan a entender cómo funciona el programa.

Empezamos por definir las constantes y variables que necesitaremos. A continuación y dentro de "void setup()", inicializamos las salidas (esta inicialización no es imprescindible puesto que las vamos a utilizar como salidas analógicas, pero es preferible hacerlo aunque sólo sea por claridad del programa).

Dentro de "void loop()", leemos el joystick, calculamos sentido y velocidad para cada motor y los movemos en consecuencia. He definido una zona neutra en el centro del recorrido de cada eje, entre 500 y 524, donde los motores estarán parados. A partir de ahí, valores menores de 500 los moverán en un sentido y valores mayores de 524 los moverán en sentido contrario (recordemos que el joystick nos entrega, para cada eje, valores entre 0 y 1023). Para calcular la velocidad de movimiento, utilizamos la función "map()", que funciona igual de bien con proporcionalidad directa que con proporcionalidad inversa. Finalmente, ponemos en marcha los motores utilizando la función "analogWrite()". Al final de cada pasada del bucle, nos detenemos una décima de segundo para que no se nos solapen las órdenes de escritura analógica con su ejecución. Fijaos que para cambiar el sentido de giro de un motor, tenemos que pasar por la zona neutra de cada eje; esto se hace así para proteger a los motores contra inversiones de giro repentinas, que no les vienen nada bien.

Manejando dos servos

Ahora, vamos a ver cómo podemos manejar dos servos. Si se trata de servos de giro continuo, funcionarán igual que los dos motores del ejemplo anterior, mientras que si se trata de servos normales, sus respectivas posiciones seguirán los movimientos del joystick en uno y otro eje. A continuación vemos el esquema de conexión:

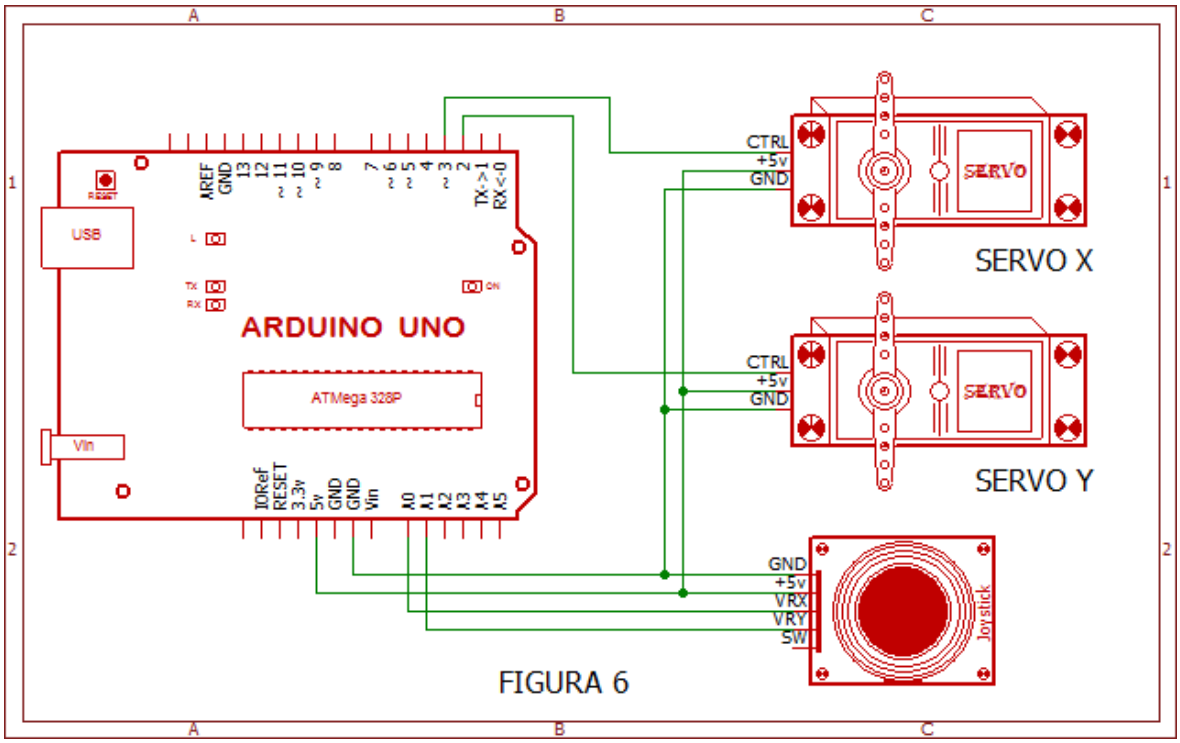


FIGURA 6

Para no complicar mucho el circuito, he alimentado los dos servos desde la salida "5v" del Arduino; esto funcionará bien con microservos, pero para servos más grandes, es preferible alimentarlos aparte como se explicaba en el artículo sobre servos (Boletín nº 36 de diciembre de 2020). El programa para hacer funcionar esto sería:

```
// Mueve dos servos siguiendo a un joystick.

// BIBLIOTECAS:
#include <Servo.h>
// FIN DE BIBLIOTECAS.

// CONSTANTES:
#define EJE_X A0 // Joystick eje X.
#define EJE_Y A1 // Joystick eje Y.
#define SERVO_X 3 // Control SERVO X.
#define SERVO_Y 2 // Control SERVO Y.
// FIN DE CONSTANTES.

// OBJETOS:
Servo ServoX;
Servo ServoY;
// FIN DE OBJETOS.

// VARIABLES GLOBALES:
unsigned int ejeX; // Lectura del joystick.
unsigned int ejeY;
byte anguloX; // Ángulos de los servos.
byte anguloY;
// FIN DE VARIABLES GLOBALES.

// CÓDIGO:
void setup() {
  ServoX.attach(SERVO_X);
```

```
  ServoY.attach(SERVO_Y);
}

void loop() {
  // Lee el joystick:
  ejeX=analogRead(EJE_X);
  ejeY=analogRead(EJE_Y);

  // Calcula los ángulos:
  anguloX=map(ejeX,0,1023,0,180);
  anguloY=map(ejeY,0,1023,0,180);

  // Mueve los servos:
  ServoX.write(anguloX);
  ServoY.write(anguloY);

  // Espera 0,1 segundos:
  delay(100);
}
// FIN DE CÓDIGO.

// Fin de programa.
```

Aquí vemos claramente cómo el manejo de servos es mucho más fácil que el de motores convencionales. Cómo indicaba en el artículo sobre servos, utilizamos la biblioteca "Servo.h", incluida en el IDE de Arduino. Se crea un objeto de tipo "Servo" para cada uno de los servos y se conectan ("attach()") a sus respectivos pines de control. A partir de ahí, el cálculo del ángulo de giro se realiza con una función "map()". Si queremos invertir el sentido de movimiento de, por ejemplo, el joystick X, nos bastará con invertir los dos últimos argumentos de su función "map()" correspondiente:

```
anguloX=map(ejeX,0,1023,180,0);
```

Cuando se use la biblioteca "Servo.h", hay que tener en cuenta que esta biblioteca utiliza el timer 1 del procesador, por lo que no es compatible con salidas PWM en los pines 9 y 10.

Con lo visto hasta aquí, podemos controlar cualquier montaje de Meccano con uno o dos joysticks para mover dos o cuatro motores o servos, siempre que los joysticks formen parte del montaje o exista una manguera de cables que los una. En el próximo artículo, veremos cómo hacer esto mismo a través de un enlace de radio y desarrollaremos un "Telemando Digital Universal para Meccano". Como siempre, dudas y preguntas al chat de WhatsApp o a: jalonsovivienda@gmail.com

Telemando digital universal

Jesús Alonso Rodríguez

Una vez que hemos visto cómo utilizar cualquier tipo de motor en Meccano, cómo emplear codificadores rotativos y cómo utilizar joysticks, sólo nos queda ver cómo manejar todo esto a distancia y con la misma fiabilidad que si tuviéramos un cable conectado a nuestro modelo.

El objetivo de este proyecto es desarrollar un sistema de telemando que sea lo suficientemente completo y tenga el suficiente alcance como para que un mismo aparato nos sirva para manejar cualquier modelo de Meccano y, al mismo tiempo, lo suficientemente fiable para que la respuesta sea tan segura como si estuviera conectado al modelo mediante un cable. En la Figura 1 vemos el aspecto que puede tener nuestro transmisor una vez terminado.

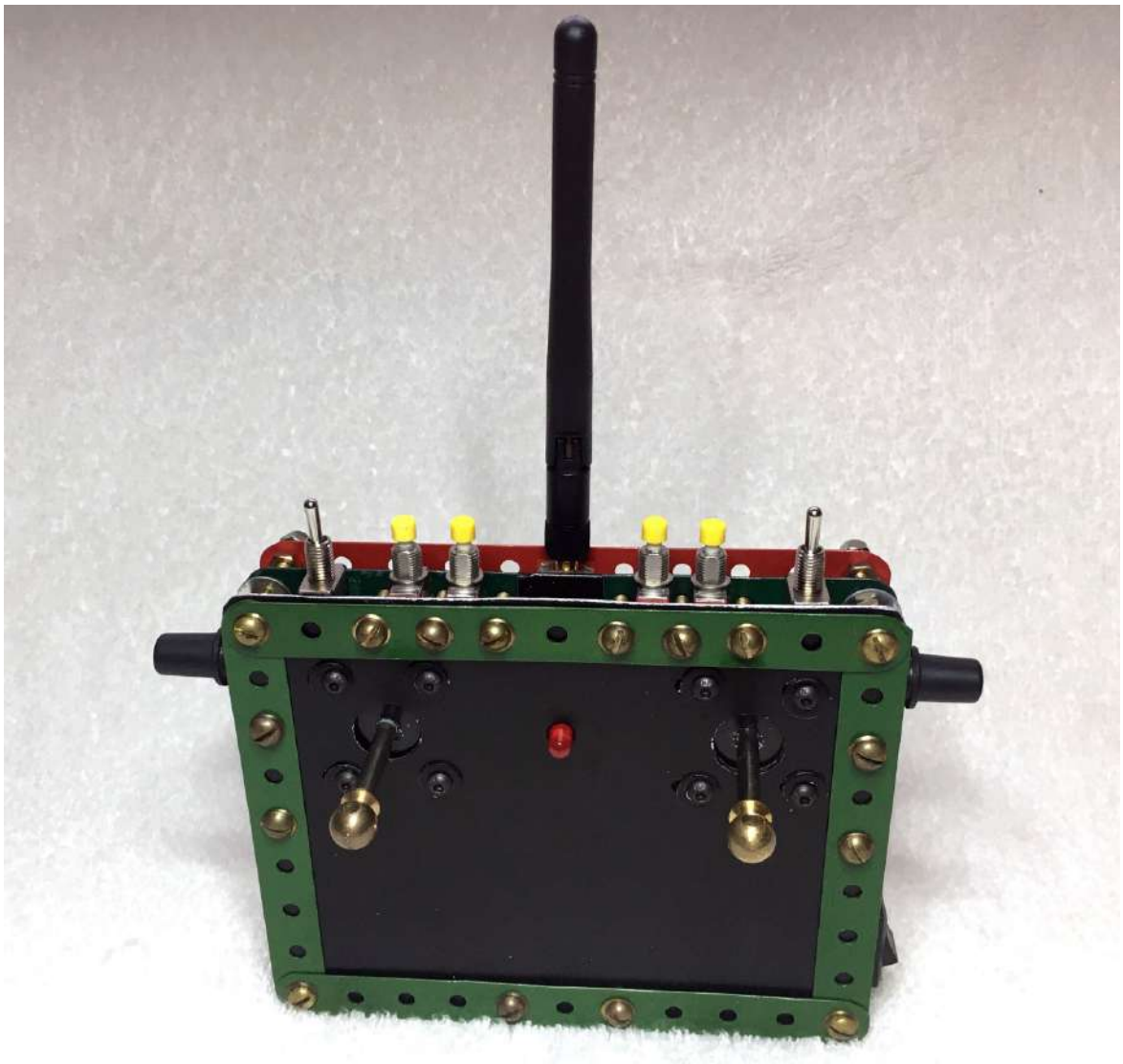


Figura 1

En primer lugar, hay que tener en cuenta que este va a ser el proyecto más complejo de los que hemos acometido hasta ahora, así que es muy conveniente que quien pretenda seguirlo esté familiarizado con la plataforma Arduino. Para ello, lo mejor es comprar el kit de iniciación y desarrollar todos los experimentos que incluye, con lo que se conseguirá irse introduciendo paulatinamente en esta tecnología. También es muy conveniente haber leído todos los artículos que he publicado hasta ahora sobre electrónica ya que, a través de ellos, he ido explicando los fundamentos teóricos y prácticos necesarios para entender este proyecto:

- **UTILIZACIÓN DE MOTORES PASO-A-PASO EN MECCANO** (Boletín 18 – diciembre de 2011)
- **REGULAR LA VELOCIDAD DE UN MOTOR** (Boletín 26 – diciembre de 2015
- **ROBOT SEGUIDOR DE LÍNEA** (Boletín 27 – junio de 2016)
- **CONTROL ELECTRÓNICO DE MOTORES: EL PUENTE EN "H"** (Boletín 35 – junio de 2020)
- **CODIFICADORES ROTATIVOS** (Boletín 35 – junio de 2020)
- **MOTORES PASO-A-PASO** (Boletín 36 – diciembre de 2020)
- **UTILIZANDO SERVOS EN MECCANO** (Boletín 36 – diciembre de 2020)
- **JOYSTICKS** (Boletín 37 – junio de 2021)

No hace falta ser un técnico en electrónica para realizar este proyecto. Quien no desee profundizar en su funcionamiento puede limitarse a copiar los esquemas de conexión y los programas y tendrá su telemando, pero también espero que este artículo sirva a quien esté interesado en aprender algo de electrónica para adentrarse en un terreno fascinante que le abrirá un mundo de posibilidades; por ello, intentaré dar todas las explicaciones necesarias sobre el funcionamiento del telemando; quien no esté interesado puede saltárselas.

Banda de frecuencias utilizada

A la hora de desarrollar un sistema de telemando, lo primero que hay que decidir es en qué frecuencia se va a operar. La utilización del espectro radioeléctrico, al ser este un bien de uso público, está sujeta a autorización administrativa, así que no podemos ponernos a emitir en cualquier frecuencia con el riesgo de provocar interferencias en otros servicios autorizados que estén operando en esa misma frecuencia. Existen, no obstante, una serie de bandas de frecuencia de uso libre en las que no es necesario obtener una autorización administrativa; estas bandas se denominan "bandas ISM" (Industrial, Scientific and Medical). La emisión en estas bandas es libre pero quien opere en ellas debe aceptar cualquier interferencia producida por otro que también esté operando en la misma banda. Las más utilizadas hoy en día son la banda de 433 Megahercios y la de 2,4 Gigahercios (2.400 Megahercios).

La primera se usa habitualmente para los telemandos que abren puertas de garaje, las llaves electrónicas de los coches y la mayoría de trasmisores de radiocontrol para modelismo. Nosotros utilizaremos la segunda: 2,4GHz. Se trata de una banda correspondiente al rango de las microondas; de hecho, los magnetrones de los hornos de microondas emiten en una frecuencia de esta banda. También funcionan en esta banda los dispositivos de interconexión inalámbrica de ordenadores mediante tecnología Wi-Fi que casi todos tenemos en casa y los sistemas "Bluetooth" aunque estos últimos, debido a su corto alcance, no son tan preocupantes en cuanto a interferencias.
La banda de 2,4 GHz incluye las frecuencias comprendidas entre los 2.400 MHz y los 2.525

MHz. Si consideramos un ancho de banda de modulación de 1 MHz, tendremos 125 canales. Los dispositivos Wi-Fi utilizan los 100 primeros así que nosotros, para no interferir con la Wi-Fi de casa, utilizaremos uno de los 25 últimos. En el software que muestro aquí, he utilizado el canal 110 correspondiente a la frecuencia de 2.510 MHz (2,51 GHz), emitiendo con una potencia aproximada de 150 milivatios en antena. En estas condiciones, el alcance teórico máximo es de 1.100 metros en una situación ideal. En la vida real, tenemos garantizado un alcance superior a 100 metros en terreno despejado y la cobertura total de nuestra vivienda. Durante las pruebas, el sistema funcionó perfectamente desde el interior de un bajo hasta un segundo piso, atravesando dos plantas y, en el exterior, desde una distancia de 50m hasta el interior de la casa. Otra de las ventajas de esta banda es que, al tratarse de frecuencias muy altas, las antenas son bastante pequeñas, lo que nos viene de perlas para integrarlas en nuestros modelos. Modularemos la señal en frecuencia mediante una técnica conocida como “FSK” (Frequency Shift Keying o Codificación por Desplazamiento de Frecuencia) que es, en transmisión digital, el equivalente a una modulación en frecuencia (FM) en transmisión analógica.

Componentes utilizados

Como no es cosa de ponerse aquí a diseñar un transmisor y un receptor de microondas, vamos a apoyarnos en algún módulo que exista ya disponible en el mercado, que sea fácil de conseguir y que tenga un precio razonable. Tras varias pruebas, he decidido optar por un módulo denominado: “NRF24L01+LNA+PA” que incluye todo lo necesario, incluso la antena, se puede comprar en Amazon y cuesta entre 3 y 5 Euros por unidad. Utilizaremos uno en el transmisor y uno en cada receptor. En la Figura 2 se puede ver el aspecto que tiene.

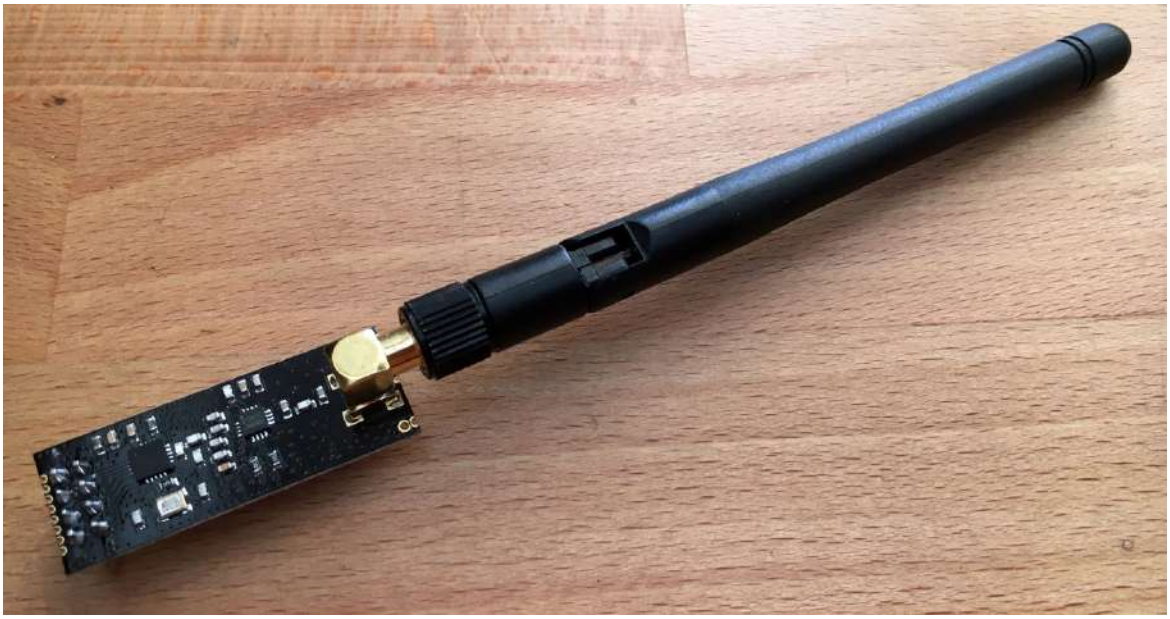


Figura 2

Este módulo se basa en el chip NRF24L01 y además, incorpora un amplificador de bajo ruido (LNA) para la recepción y un amplificador de potencia (PA) para la transmisión. Se trata de un “transceptor”, es decir, un dispositivo que puede tanto emitir como recibir pero no ambas cosas a la vez. Cada módulo puede funcionar como transmisor o como receptor y cambiar su rol en cualquier momento. Este modo de comunicación se denomina “semi-duplex”.

Una particularidad importante de este módulo es que se alimenta a 3,3 voltios y no a los 5 voltios a los que se alimentan el procesador del Arduino y la mayoría de circuitos digitales; de hecho, si lo alimentamos a 5 voltios se destruirá. Por ello, es muy recomendable utilizar el pequeño módulo adaptador que se muestra en la Figura 3 y que también se puede comprar en Amazon por un par de Euros. Este adaptador incluye un reductor de voltaje a 3,3 voltios, lo que nos da una enorme libertad a la hora de alimentar nuestro transmisor.

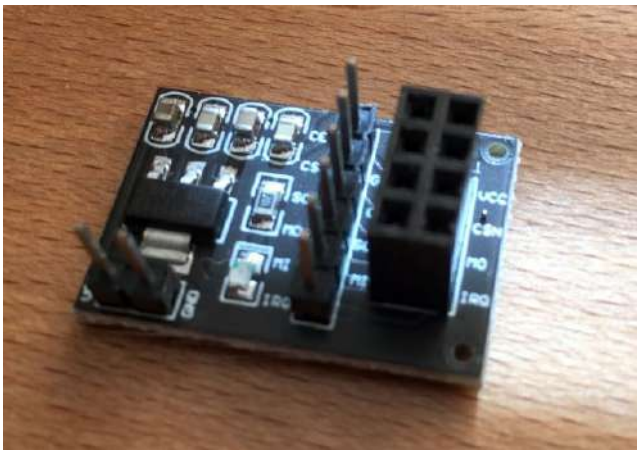


Figura 3

Funcionamiento

Cada décima de segundo, el transmisor realizará una transmisión; si el receptor la ha recibido correctamente, le enviará una respuesta de confirmación que llamamos “Acknowledge” o abreviado: “ACK”. Si el transmisor no recibe el ACK, volverá a emitir la información cada 250 microsegundos y un máximo de 5 veces. Si recibe el ACK, nos lo indicará apagando un LED durante 50 milisegundos. De esta forma, cuando el enlace entre transmisor y receptor funcione correctamente, veremos parpadear la lucecita roja del transmisor a razón de 10 parpadeos por segundo; si no hay enlace, la veremos encendida fija.

Para que el receptor sepa que ha recibido la transmisión correctamente, el chip NRF24L01 añade cierta información redundante a los códigos que nosotros le mandamos enviar, componiendo lo que se denomina una “trama”. Cada una de estas tramas se compone de preámbulo, dirección, carga útil y CRC. El preámbulo consiste en una serie de unos y ceros alternados que ayudan al receptor a ajustar niveles de ganancia. A continuación viene una dirección de 5 bytes (40 bits) que le indica al receptor si la trama está destinada a él o no. Después de la dirección va la carga útil (llamada “payload”) que es la información que realmente queremos enviar (luego hablaremos de ella). Finalmente se envía un byte que es el resultado de realizar una operación lógica XOR bit a bit entre todos los bytes precedentes; este byte se denomina “Comprobación de Redundancia Cíclica” o CRC y sirve para que el receptor pueda detectar si se ha perdido algún bit durante la transmisión y, por tanto, si el mensaje se ha recibido correctamente o no.

Como se ve, se trata de un enlace bidireccional donde el receptor está constantemente confirmándole al transmisor la correcta recepción de los códigos transmitidos. De esta forma, mientras veamos parpadear el LED del transmisor, podemos estar seguros de que la comunicación se está realizando con total fiabilidad.

Esta forma tan segura de establecer un enlace está definida en la capa 2 (denominada “capa de enlace”) del modelo OSI (“Open Systems Interconection”) definido por ISO (“Organización Internacional de Normalización”) y está en la base de todas las comunicaciones que se realizan a través de redes informáticas. Ningún telemando comercial para modelismo garantiza una comunicación tan segura.

Diseño del transmisor

Hemos dicho que queremos que nuestro transmisor valga para manejar cualquier modelo que hagamos con Meccano, desde una grúa estática o un brazo robótico hasta un vehículo que se mueva. Para ello, vamos a diseñar un mando muy versátil que contendrá dos joysticks, dos potenciómetros (uno en cada lateral), dos conmutadores y cuatro pulsadores, más los dos pulsadores de los joysticks. De esta forma, tendremos lo que en un telemando se denominan 6 canales analógicos y 8 canales digitales de telemando. Ojo: no confundir con los canales de radio que se corresponden con frecuencias y de esos sólo utilizamos uno, es decir, nuestros 6 canales analógicos y 8 canales digitales de telemando se envían por un único canal de radio. En las Figuras 4 y 5 podemos ver el esquema teórico y el diagrama de cableado del transmisor. He decidido representar ambos diagramas para que aquellos que estén iniciándose en la electrónica puedan ver la correspondencia entre las dos formas de representar un circuito.

Esquema teórico:

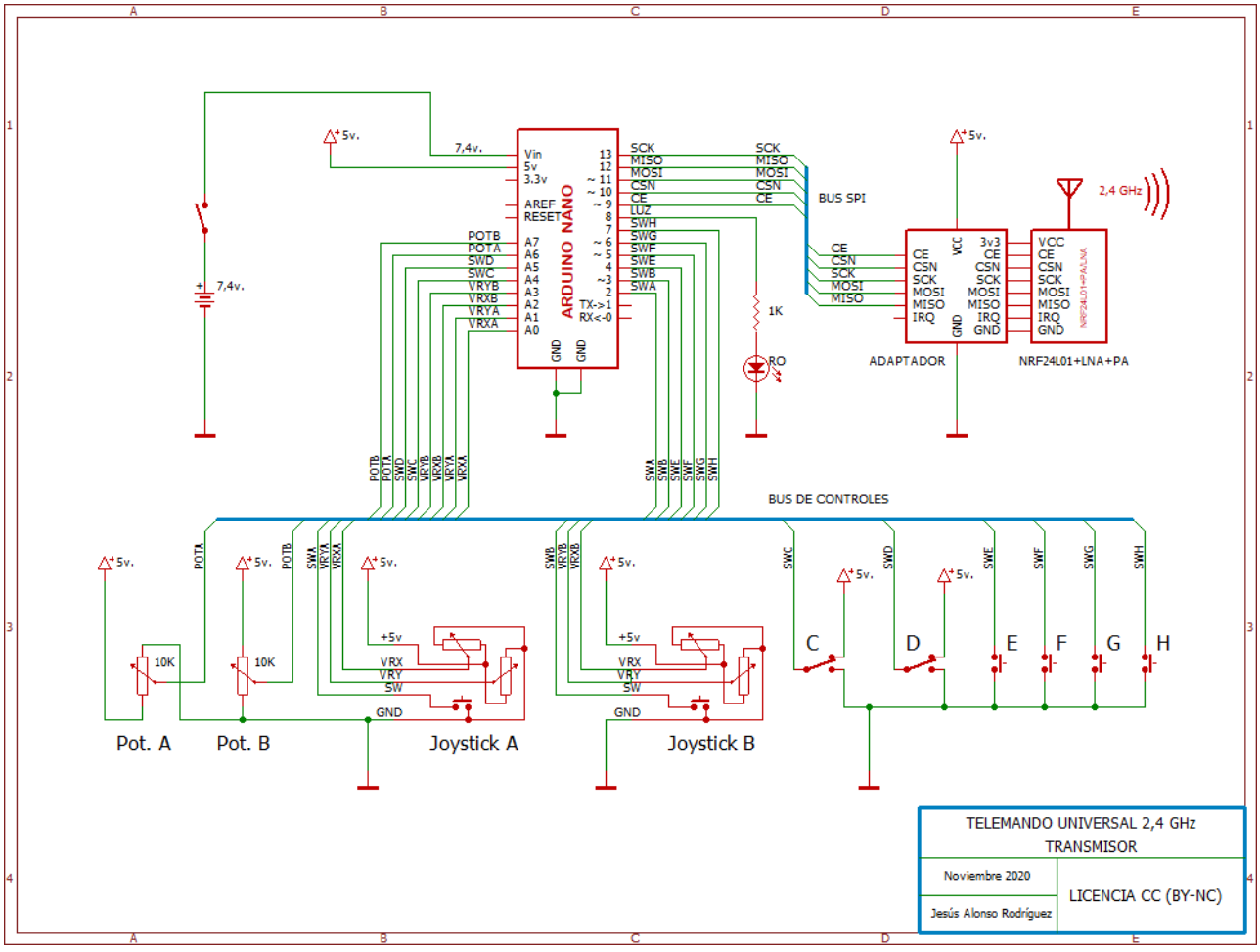


Figura 4

Diagrama de cableado

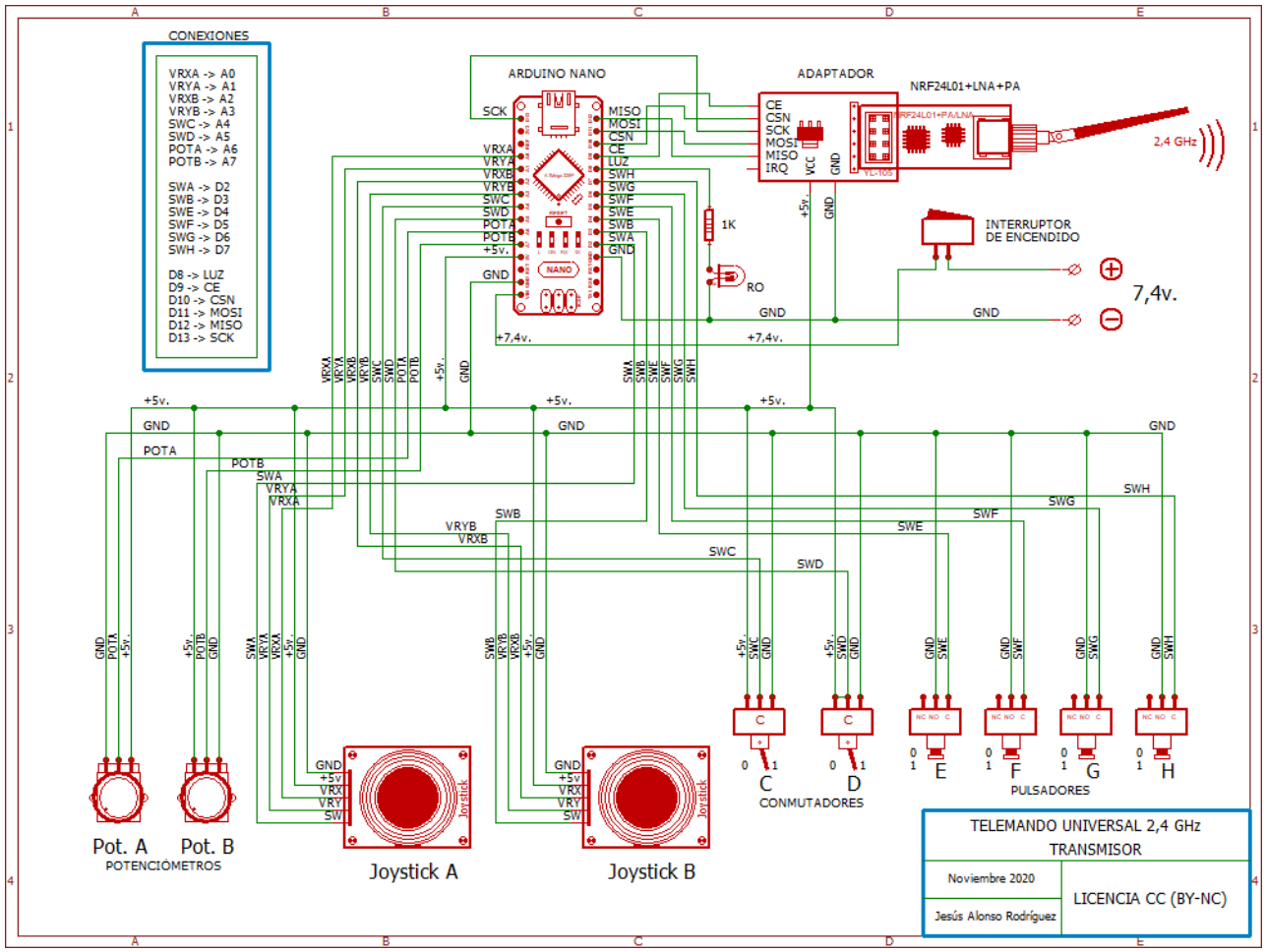


Figura 5

Como se aprecia en los diagramas, conectamos los cuatro ejes de los joysticks a cuatro entradas analógicas de un Arduino Nano, los dos conmutadores a otras dos entradas analógicas (que redefiniremos como entradas digitales en el programa) y los dos potenciómetros a las dos últimas entradas analógicas. Es imprescindible utilizar un Arduino Nano, ya que tiene 8 entradas analógicas, frente a sólo 6 en el Arduino Uno, además es más pequeño y nos permite construir un mando más compacto.

Los dos pulsadores de los joysticks y los cuatro pulsadores restantes los conectamos a los pines digitales 2 a 7 configurados como “INPUT PULLUP”. El pin 8 lo configuraremos como “OUTPUT” y le conectaremos el diodo LED que nos va a indicar la existencia de enlace con el receptor. Finalmente, utilizamos los pines 9 a 13 para comunicarnos con el módulo transceptor a través del protocolo “SPI” (Serial Peripheral Interface).

Payload

Cómo ya hemos explicado, denominamos “Payload” o carga útil a la información que realmente queremos transmitir. El módulo que vamos a utilizar puede manejar payloads de hasta 32 bytes; nosotros sólo necesitamos enviar 7 bytes (56 bits). Cada byte contiene

8 bits y cada bit puede ser “0” o “1”. Al agrupar 8 bits podemos representar, en base 2, cualquier número entero positivo comprendido entre “0” y “255” en decimal.

Utilizaremos el primer byte para representar la posición del joystick A (el izquierdo) en el eje “X”, de forma que enviaremos un “0” si el joystick está totalmente a la izquierda y un “255” si está totalmente a la derecha. En puntos intermedios se enviarán valores intermedios. El segundo byte representará la posición del joystick A en el eje “Y”, de forma que enviaremos un “0” si está abajo y un “255” si está totalmente arriba. De esta forma, si el joystick está en el centro, se enviarán dos códigos “128”. El tercer y cuarto byte será lo mismo para el joystick B (el derecho). El quinto byte representará la posición del potenciómetro A (el izquierdo) y contendrá un “0” si el potenciómetro está en el extremo en sentido horario y un “255” si está en el extremo en sentido antihorario. De igual forma, el sexto byte representará la posición del potenciómetro B (el derecho) y contendrá un “0” si el potenciómetro está en el extremo en sentido antihorario y un “255” si está en el extremo en sentido horario. Por tanto, estos primeros seis bytes codificarán los seis canales analógicos de nuestro mando. El séptimo y último byte lo vamos a utilizar para codificar los 8 canales digitales. Sin pulsar ningún pulsador, todos los bits serán “1”. El pulsador del joystick izquierdo pondrá a “0” el primer bit, el del joystick derecho pondrá a “0” el segundo bit. Los conmutadores izquierdo y derecho pondrán a cero, respectivamente, el tercer y cuarto bit cuando estén hacia abajo. Finalmente, los cuatro pulsadores restantes pondrán a cero cada uno de los últimos cuatro bits cuando estén pulsados. Resumiendo:

PAYLOAD: 7 Bytes

Byte 1: Joystick A - Eje X Byte 2: Joystick A - Eje Y Byte 3: Joystick B - Eje X Byte 4: Joystick B - Eje Y Byte 5: Potenciómetro A Byte 6: Potenciómetro B Byte 7: Controles digitales: - Bit 0: Pulsador Joystick A - Bit 1: Pulsador Joystick B - Bit 2: Conmutador C - Bit 3: Conmutador D - Bit 4: Pulsador E - Bit 5: Pulsador F - Bit 6: Pulsador G - Bit 7: Pulsador H JOYSTICKS: Eje X: - A la izquierda: 0 - A la derecha: 255	Eje Y: - Abajo o atrás: 0 - Arriba o delante: 255 POTENCIÓMETROS: Izquierdo: - Extremo horario: 0 - Extremo antihorario: 255 Derecho: - Extremo antihorario: 0 - Extremo horario: 255 Conmutadores: - Arriba: 1 - Abajo: 0 Pulsadores: - Sin pulsar: 1 - Pulsado: 0

Es muy importante tener presente siempre este esquema de “payload” cuando diseñemos cada uno de los receptores que han de recibir estos códigos para actuar sobre el modelo. En la Figura 6 vemos los componentes que vamos a utilizar colocados sobre dos placas de 11 x 5 agujeros solapadas en una línea de agujeros, lo que nos determinará el tamaño de nuestro transmisor: 11 x 9 agujeros o 5 ½ x 4 ½ pulgadas.

Como se ve en la imagen, utilizamos una mini placa "breadboard" para enchufar el Arduino Nano y simplificar las conexiones para tener que soldar lo menos posible. También se puede ver que utilizamos una batería Li-Po de 7,4 voltios y 1.100 miliamperios hora. Con esta batería totalmente cargada y debido al bajo consumo del transmisor, este puede funcionar sin interrupción durante aproximadamente 17 horas. En el prototipo, he utilizado dos potenciómetros muy pequeñitos procedentes de desguace, pero también se pueden utilizar dos potenciómetros normales de 10K (10.000 Ohmios), buscando la forma de acoplarlos en el chasis de Meccano. Obsérvese que los joysticks están girados 90 grados a la izquierda y las patas quedan en la parte de abajo, con los que los ejes "X" e "Y" quedan intercambiados.

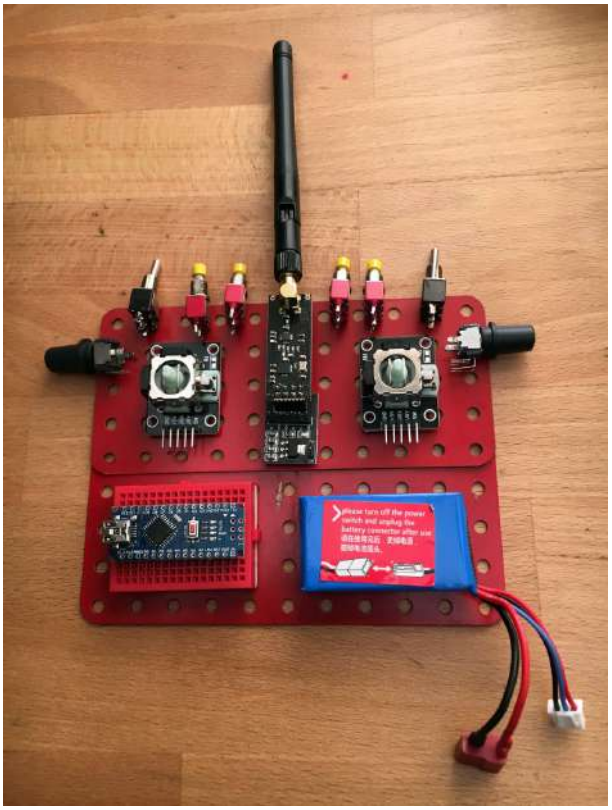


Figura 6

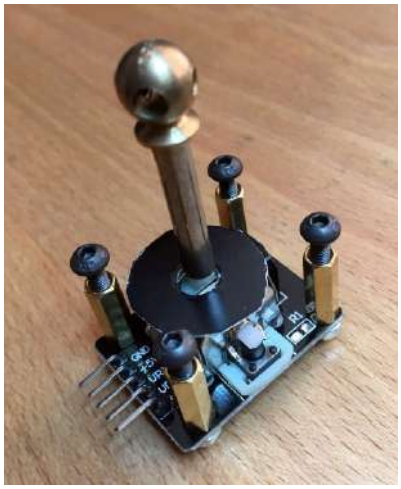


Figura 7

En la Figura 7 vemos cómo adaptar un empalmador de ejes y un soporte de balaustrada a los joysticks para darles un aspecto "meccanero". Las cuatro torretas para atornillarlo también se pueden conseguir por Internet o en cualquier tienda de componentes electrónicos.

Finalmente, en la Figura 8 vemos el transmisor totalmente montado y cableado, antes de ponerle las dos placas de abajo que hacen de tapa. Para la parte de arriba, yo he optado por fabricarme una placa especial en aluminio de 1mm. y pintarla de negro mate, pero también se podría hacer con piezas estrictamente de Meccano. En la Figura 9 vemos el aspecto final visto desde arriba:

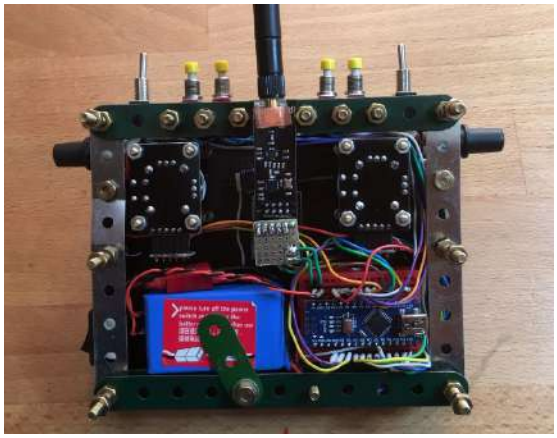


Figura 8

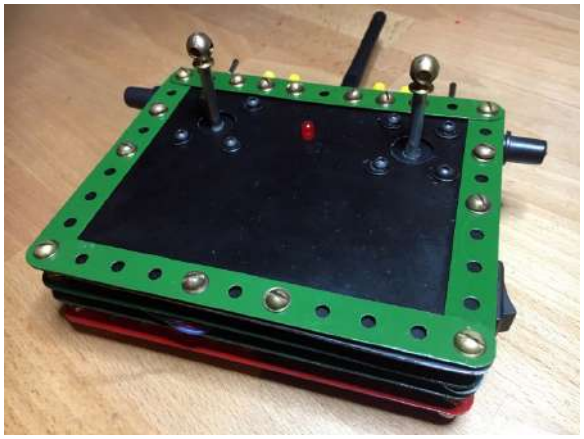


Figura 9

Software

Y ahora que ya tenemos todo montado, vamos a hacer que funcione. Este sería el programa que hay que cargar en el Arduino Nano:

```
// Telemando Digital Universal Meccano.
// Transmisor 2,4 GHz
//
// Versión: 1.0
// Noviembre de 2020
// (c) Jesús Alonso Rodríguez
// Licencia: Creative Commons (by-nc).
// Esta obra está licenciada bajo la Licencia Creative Commons
// Atribución-NoComercial 4.0 Internacional.
// Para ver una copia de esta licencia, visita
// http://creativecommons.org/licenses/by-nc/4.0/.

// COMPROBADO.

// BIBLIOTECAS:
#include <SPI.h>
#include <RF24.h>
// FIN DE BIBLIOTECAS.

// CONSTANTES:
#define VRXA A0
#define VRYA A1
#define VRXB A2
#define VRYB A3
#define SWC A4
#define SWD A5
#define POTA A6
#define POTB A7
#define SWA 2
#define SWB 3
#define SWE 4
#define SWF 5
#define SWG 6
#define SWH 7
#define LUZ 8
#define CE_PIN 9
#define CSN_PIN 10
// SPI_MOSI 11
// SPI_MISO 12
// SPI_SCK 13
// FIN DE CONSTANTES.

// OBJETOS:
RF24 radio(CE_PIN, CSN_PIN);
// FIN DE OBJETOS.

// VARIABLES GLOBALES:
int ejes[6];
byte direccion[5] = {0, 0, 0, 1, 5};
byte TX_buffer[7];
// FIN DE VARIABLES GLOBALES.
```

```
// CÓDIGO:
void setup() {
  pinMode(SWA, INPUT_PULLUP);
  pinMode(SWB, INPUT_PULLUP);
  pinMode(SWE, INPUT_PULLUP);
  pinMode(SWF, INPUT_PULLUP);
  pinMode(SWG, INPUT_PULLUP);
  pinMode(SWH, INPUT_PULLUP);
  pinMode(SWC, INPUT);
  pinMode(SWD, INPUT);
  pinMode(LUZ, OUTPUT);
  digitalWrite(LUZ,HIGH);
  radio.begin();
  radio.setChannel(110);
  radio.setPALevel(RF24_PA_MAX);
  radio.stopListening();
  radio.openWritingPipe(direccion);
  radio.setPayloadSize(7);
  radio.setRetries(0,5); // 250us/5 retries
}

void loop() {
  ejes[0] = analogRead(VRXA);
  ejes[1] = analogRead(VRYA);
  ejes[2] = analogRead(VRXB);
  ejes[3] = analogRead(VRYB);
  ejes[4] = analogRead(POTA);
  ejes[5] = analogRead(POTB);
  for (int i = 0; i < 6; i++) {
    TX_buffer[i] = map(ejes[i], 0, 1023, 0, 255);
  }
  bitWrite(TX_buffer[6], 0, digitalRead(SWA));
  bitWrite(TX_buffer[6], 1, digitalRead(SWB));
  bitWrite(TX_buffer[6], 2, digitalRead(SWC));
  bitWrite(TX_buffer[6], 3, digitalRead(SWD));
  bitWrite(TX_buffer[6], 4, digitalRead(SWE));
  bitWrite(TX_buffer[6], 5, digitalRead(SWF));
  bitWrite(TX_buffer[6], 6, digitalRead(SWG));
  bitWrite(TX_buffer[6], 7, digitalRead(SWH));
  if(radio.write(TX_buffer,7)){
    digitalWrite(LUZ, LOW);
  }
  delay(50);
  digitalWrite(LUZ, HIGH);
  delay(50);
}
// FIN DE CÓDIGO.

// Fin de programa.
```

Vamos a analizar el programa porque tiene algunas sutilezas:
Antes de nada, a quien haya adquirido el Arduino Nano en una tienda china o en Amazon, probablemente le haya llegado con el chip CH340 en vez del FT232RL para hacer la comunicación por USB. En ese caso, lo primero que hay que hacer es descargarse el driver de este chip de la dirección:
https://github.com/HobbyComponents/CH340-Drivers/blob/master/CH341SER/SETUP.EXE

Empecemos por las bibliotecas: vemos que utilizamos dos. La primera “SPI.h” forma parte del IDE de Arduino y no es necesario descargársela. La segunda “RF24.h” habrá que descargársela desde:
https://www.arduinolibraries.info/libraries/rf24

Se recomienda descargar e instalar la versión más reciente. Para instalarla, desde el IDE de Arduino, ir a “Programa”, “Incluir librería”, “Añadir biblioteca .Zip” y buscar el archivo ZIP descargado (si se está trabajando en Windows, normalmente estará en la carpeta “Descargas”). Luego habrá que cerrar y volver a abrir el IDE antes de empezar a teclear el programa.

Después de las bibliotecas, definimos las constantes. Las tres últimas, que son necesarias para la comunicación SPI, no hace falta definirlas ya que la biblioteca “SPI.h” se encarga de ello, pero se han puesto como comentarios para mayor claridad. Véase aquí la correspondencia entre la definición de constantes y los diagramas esquemáticos que representan el conexionado.

A continuación, creamos el objeto “radio” asociado a la biblioteca “RF24.h” que será el que utilicemos para manejar el transceptor. Después creamos tres variables globales: un vector de 6 enteros sin signo (16 bits) que utilizaremos para leer los ejes de los joysticks y los potenciómetros, un vector de 5 bytes donde pondremos la dirección (yo he utilizado como dirección mi número de socio de ACEAM, pero se puede poner cualquiera; el único requisito es que el transmisor y los receptores tengan la misma dirección) y un vector de 7 bytes que contendrá el “payload” a enviar.

Y ya entramos al código: primero configuramos los pines, luego encendemos la luz y, a continuación, pasamos a configurar el transceptor. Este paso es muy importante y ha requerido varias horas de experimentación para afinarlo, así que voy a comentar cada línea: La configuración del transceptor la hacemos a través de métodos asociados al objeto “radio”. Un método es como una función pero asociada a un objeto. Obviamente, no hay que definirlos porque ya están definidos en la biblioteca.

En la primera línea “`radio.begin();`”, inicializamos el funcionamiento del transceptor. En la segunda línea le indicamos el canal en el que vamos a transmitir; lógicamente, emisor y receptores deberán operar en el mismo canal. Yo he elegido el canal 110; se puede elegir cualquiera entre el 1 y el 125, pero recomiendo no utilizar los 100 primeros ya que podrían interferir con el Wi-Fi de casa. También recomiendo, en las exposiciones, que cada socio utilice un canal diferente para no interferirnos entre nosotros; ya sabéis cual utilizo yo.

La siguiente línea le indica al módulo que transmita a la potencia máxima (aproximadamente, 150 milivatios) para conseguir el máximo alcance. A continuación, la orden “`radio.stopListening();`” coloca el transceptor en modo transmisión (como si pulsáramos el botón “PTT” en un walky-talky). La siguiente línea le indica la dirección que tiene que añadir al construir la trama. A continuación, le indicamos la longitud de nuestro payload; podríamos no hacerlo pero entonces siempre emitiría un payload de 32 bytes, rellenando con ceros y haciendo que la trama fuera innecesariamente larga. Finalmente, fijamos el intervalo entre reintentos de transmisión, cuando no se reciba ACK, en 250 microsegundos y le decimos que, como máximo, realice 5 reintentos. En realidad, no sería necesario hacer reintentos de transmisión, ya que estamos mandando tramas continuamente, pero lo correcto es hacerlo así para cumplir con lo especificado en la capa 2 del modelo OSI. Limitamos a 5 reintentos

para que no se solape con la siguiente trama a enviar 100 milisegundos después.

Y una vez configurada la "radio", entramos en el bucle "`void loop()`" que se ejecutará continuamente. Primero leemos los valores analógicos que nos entregan los ejes de los joysticks y de los potenciómetros, sobre el vector "ejes[]". Luego los transferimos a los primeros seis bytes del vector "TX_buffer[]" que contendrá la payload; aquí hay que tener en cuenta que la lectura analógica nos entrega números comprendidos entre "0" y "1023" y necesitamos números comprendidos entre "0" y "255", por eso usamos la función "map()", aunque también podríamos, simplemente, dividir por cuatro. A continuación, vamos poniendo a "1" o a "0" cada bit del séptimo byte, en función del estado de los pulsadores y conmutadores.

Una vez que tenemos la payload completa, utilizamos el método "`radio.write`" para transmitirla. Este método nos devolverá un "true" si la transmisión se ha realizado con éxito (o sea, si ha recibido un ACK) y un "false" en caso contrario; por eso, lo ponemos dentro de una estructura "if()" y, en caso afirmativo, apagamos la luz. Luego esperamos 50 milisegundos, la encendemos y esperamos otros 50 milisegundos. Como todo lo anterior ocurre muy rápido, transmitiremos una trama cada, aproximadamente, 100 milisegundos, o sea, 10 tramas por segundo que es más que suficiente para que nuestros modelos estén totalmente controlados.

Y hasta aquí, las instrucciones para construir un transmisor. Ahora vamos a ver cómo construir los receptores.

Receptor 1: de prueba

Mientras no tengamos un receptor funcionando, no veremos parpadear la luz del transmisor, ya que no habrá nadie contestándole con ACKs. Vamos a empezar por un receptor muy sencillito que nos permitirá ver, en el ordenador, los códigos que está enviando el transmisor. El plano de cableado sería el que se muestra en la Figura 10:

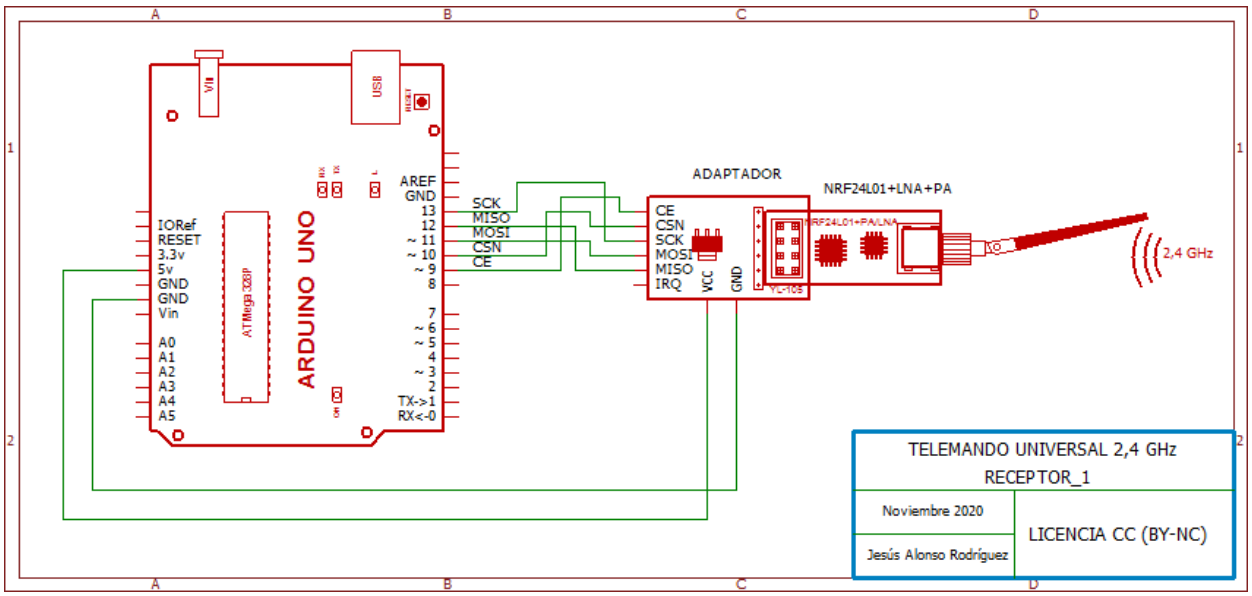


Figura 10

Y este sería el programa a cargar en el Arduino Uno para que funcione:

```
// Telemando Universal Meccano.
// Receptor_1 2,4 GHz
//
// Muestra los datos recibidos en el Monitor Serie.
//
// Versión: 1.0
// Noviembre de 2020
// (c) Jesús Alonso Rodríguez
// Licencia: Creative Commons (by-nc).
// Esta obra está licenciada bajo la Licencia Creative Commons
// Atribución-NoComercial 4.0 Internacional.
// Para ver una copia de esta licencia, visita
// http://creativecommons.org/licenses/by-nc/4.0/.

// COMPROBADO.

// BIBLIOTECAS:
#include <SPI.h>
#include <RF24.h>
// FIN DE BIBLIOTECAS.

// CONSTANTES:
#define CE_PIN 9
#define CSN_PIN 10
// FIN DE CONSTANTES.

// OBJETOS:
RF24 radio(CE_PIN,CSN_PIN);
// FIN DE OBJETOS.

// VARIABLES GLOBALES:
byte direccion[5]={0,0,0,1,5};
byte RX_buffer[7];
// FIN DE VARIABLES GLOBALES.

// CÓDIGO:
void setup() {
  radio.begin();
  radio.setPALevel(RF24_PA_MAX);
  radio.setChannel(110);
  radio.setPayloadSize(7);
  radio.setRetries(0,5); // 250us/5 retries.
  radio.openReadingPipe(1,direccion);
  radio.startListening();
  Serial.begin(9600);
  Serial.println("Empezamos...");
}

void loop() {
  if(radio.available()){
    radio.read(RX_buffer,7);
    Serial.print("AX:");
    Serial.print(RX_buffer[0]);
    Serial.print(" AY:");
    Serial.print(RX_buffer[1]);
    Serial.print(" BX:");
```

```
      Serial.print(RX_buffer[2]);
      Serial.print(" BY:");
      Serial.print(RX_buffer[3]);
      Serial.print(" POT_A:");
      Serial.print(RX_buffer[4]);
      Serial.print(" POT_B:");
      Serial.print(RX_buffer[5]);
      Serial.print(" Switches:");
      for(byte i=0;i<8;i++){
        Serial.print(bitRead(RX_buffer[6],i));
      }
      Serial.println(' ');
    }
}
// FIN DE CÓDIGO.
// Fin de programa.
```

Cómo se puede ver, la inicialización es muy similar a la del transmisor, solo que ponemos la radio en recepción con: "`radio.startListening();`". A partir de ahí, vamos recibiendo las tramas e imprimiendo las "payloads" recibidas en el monitor serie (se abre en "Herramientas", "Monitor Serie" y deberá estar configurado a 9600 baudios, que es la configuración por defecto). Lógicamente, este receptor hay que hacerlo funcionar conectado al ordenador. Una vez que lo tengamos funcionando, podemos dedicarnos a "jugar" con los controles del transmisor e ir viendo cómo cambian los datos recibidos en la pantalla del ordenador.

Receptor 2: mover cuatro servos

Ahora vamos a complicarlo un poquito más y mover cuatro servos con los dos joysticks. El diagrama de conexiones se muestra en la Figura 11:

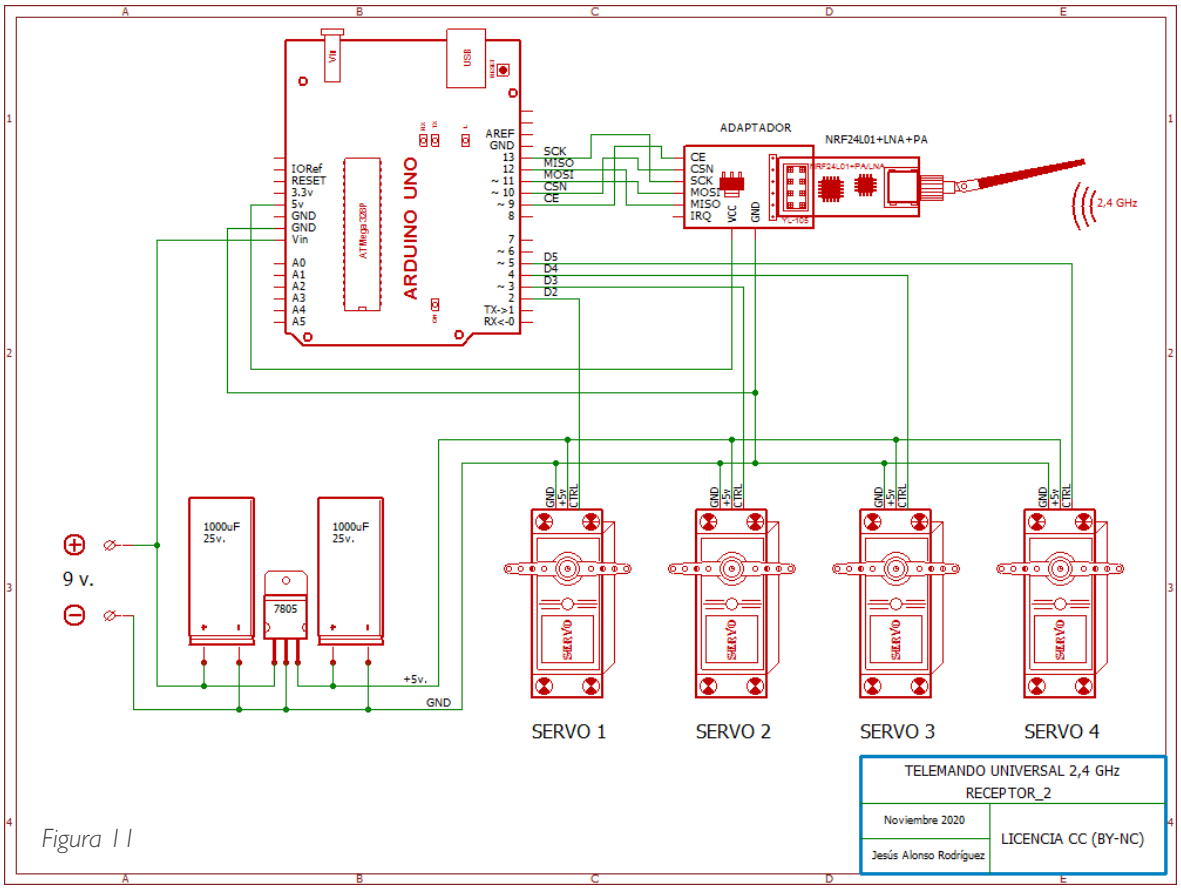


Figura 11

Y este sería el programa para hacerlo funcionar:

```
// Telemando Universal Meccano.
// Receptor_2 2,4 GHz
//
// Maneja cuatro servos desde los joysticks.
//
// Versión: 1.0
// Noviembre de 2020
// (c) Jesús Alonso Rodríguez
// Licencia: Creative Commons (by-nc).
// Esta obra está licenciada bajo la Licencia Creative Commons
// Atribución-NoComercial 4.0 Internacional.
// Para ver una copia de esta licencia, visita
// http://creativecommons.org/licenses/by-nc/4.0/.

// COMPROBADO.

// BIBLIOTECAS:
#include <SPI.h>
#include <RF24.h>
#include <Servo.h>
// FIN DE BIBLIOTECAS.

// CONSTANTES:
#define CE_PIN 9
#define CSN_PIN 10
#define SERVO_1_PIN 2
#define SERVO_2_PIN 3
#define SERVO_3_PIN 4
#define SERVO_4_PIN 5
// FIN DE CONSTANTES.

// OBJETOS:
RF24 radio(CE_PIN,CSN_PIN);
Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;
// FIN DE OBJETOS.

// VARIABLES GLOBALES:
byte direccion[5]={0,0,0,1,5};
byte RX_buffer[7];
// FIN DE VARIABLES GLOBALES.

// CÓDIGO:
void setup() {
  radio.begin();
  radio.setPALevel(RF24_PA_MAX);
  radio.setChannel(110);
  radio.setPayloadSize(7);
  radio.setRetries(0,5); // 250us/5 retries.
  radio.openReadingPipe(1,direccion);
  radio.startListening();
  servo1.attach(SERVO_1_PIN);
  servo2.attach(SERVO_2_PIN);
  servo3.attach(SERVO_3_PIN);
```

```
  servo4.attach(SERVO_4_PIN);
}

void loop() {
  if(radio.available()){
    radio.read(RX_buffer,7);
    servo1.write(map(RX_buffer[0],0,255,0,180));
    servo2.write(map(RX_buffer[1],0,255,0,180));
    servo3.write(map(RX_buffer[2],0,255,0,180));
    servo4.write(map(RX_buffer[3],0,255,0,180));
  }
}
// FIN DE CÓDIGO.

// Fin de programa.
```

Si se quiere que uno de los servos funcione al revés, se pueden invertir los dos últimos argumentos de su sentencia “map” correspondiente; por ejemplo:

```
servo1.write(map(RX_buffer[0],0,255,180,0));
```

Receptor 3: mover una grúa de cuatro motores

Ahora, vamos a manejar una grúa con cuatro motores. Utilizaremos los módulos de control de motores basados en L298. Vemos el diagrama de conexiones en la Figura 12:

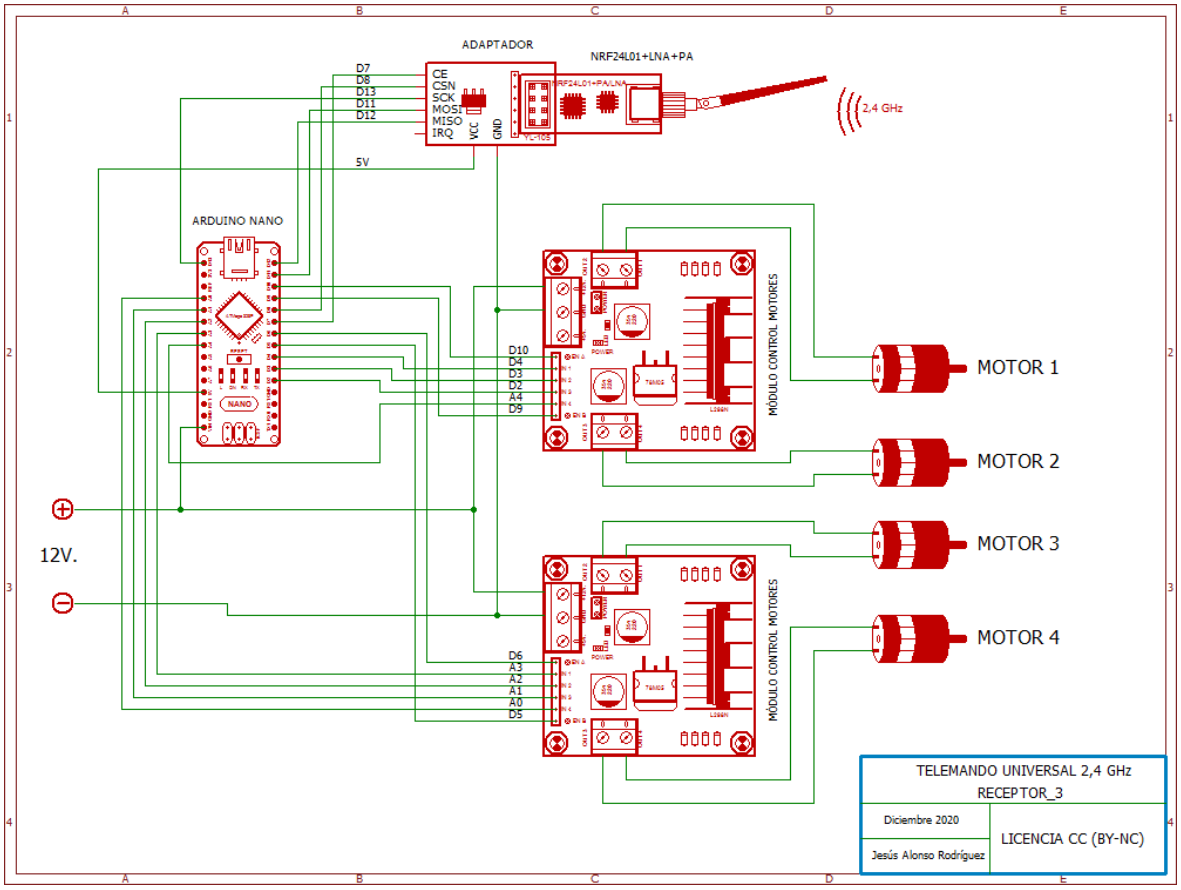


Figura 12

Vemos en el diagrama que, para cada motor, utilizamos un pin para que gire en un sentido, otro pin para que gire en otro sentido y un tercer pin para controlar su velocidad; este último ha de ser uno de los que puedan generar señal PWM y lo conectamos por la entrada "enable" correspondiente. De esta forma, podremos controlar la velocidad de giro de cada uno de los cuatro motores independientemente. Cómo vimos en el artículo sobre los joysticks, controlaremos dos motores con cada joystick, de forma que giren en uno u otro sentido dependiendo de a qué lado movamos el joystick y su velocidad sea tanto mayor cuanto más alejado esté el joystick de su centro. Como no tenemos tantos pines digitales, "reciclaremos" cinco de las entradas analógicas y las convertiremos en salidas digitales (no os preocupéis, lo hace el programa). Este sería el programa para hacerlo funcionar:

```
// Telemando Universal Meccano.
// Receptor_3 2,4 GHz
//
// Maneja cuatro motores desde los joysticks.
//
// Versión: 1.0
// Diciembre de 2020
// (c) Jesús Alonso Rodríguez
// Licencia: Creative Commons (by-nc).
// Esta obra está licenciada bajo la Licencia Creative Commons
// Atribución-NoComercial 4.0 Internacional.
// Para ver una copia de esta licencia, visita
// http://creativecommons.org/licenses/by-nc/4.0/.

// COMPROBADO.

// BIBLIOTECAS:
#include <SPI.h>
#include <RF24.h>
// FIN DE BIBLIOTECAS.

// CONSTANTES:
#define CE_PIN 7
#define CSN_PIN 8
#define M1_AV 4
#define M1_RE 3
#define M1_VE 10
#define M2_AV 2
#define M2_RE A4
#define M2_VE 9
#define M3_AV A3
#define M3_RE A2
#define M3_VE 6
#define M4_AV A1
#define M4_RE A0
#define M4_VE 5
#define NEUTRO_MIN 123 // Neutro mínimo de cada joystick.
#define NEUTRO_MAX 133 // Neutro máximo de cada joystick.
// FIN DE CONSTANTES.

// OBJETOS:
RF24 radio(CE_PIN,CSN_PIN);
// FIN DE OBJETOS.

// VARIABLES GLOBALES:
```

```
byte direccion[5]={0,0,0,1,5};
byte RX_buffer[7];
unsigned long failSafe;
// FIN DE VARIABLES GLOBALES.

// CÓDIGO:
void setup() {
  pinMode(M1_AV,OUTPUT);
  pinMode(M1_RE,OUTPUT);
  pinMode(M1_VE,OUTPUT);
  pinMode(M2_AV,OUTPUT);
  pinMode(M2_RE,OUTPUT);
  pinMode(M2_VE,OUTPUT);
  pinMode(M3_AV,OUTPUT);
  pinMode(M3_RE,OUTPUT);
  pinMode(M3_VE,OUTPUT);
  pinMode(M4_AV,OUTPUT);
  pinMode(M4_RE,OUTPUT);
  pinMode(M4_VE,OUTPUT);
  radio.begin();
  radio.setPALevel(RF24_PA_MAX);
  radio.setChannel(110);
  radio.setPayloadSize(7);
  radio.setRetries(0,5); // 250us/5 retries.
  radio.openReadingPipe(1,direccion);
  radio.startListening();
}

void loop(){
  if(radio.available()){
    failSafe=millis();
    radio.read(RX_buffer,7);

    // Mueve Motor 1:
    if(RX_buffer[0]<NEUTRO_MIN){
      byte veloc=map(RX_buffer[0],NEUTRO_MIN,0,0,255);
      analogWrite(M1_VE,veloc);
      digitalWrite(M1_AV,LOW);
      digitalWrite(M1_RE,HIGH);
    }else if(RX_buffer[0]>NEUTRO_MAX){
      byte veloc=map(RX_buffer[0],NEUTRO_MAX,255,0,255);
      analogWrite(M1_VE,veloc);
      digitalWrite(M1_AV,HIGH);
      digitalWrite(M1_RE,LOW);
    }else {
      digitalWrite(M1_VE,LOW);
      digitalWrite(M1_AV,LOW);
      digitalWrite(M1_RE,LOW);
    }

    // Mueve Motor 2:
    if(RX_buffer[1]<NEUTRO_MIN){
      byte veloc=map(RX_buffer[1],NEUTRO_MIN,0,0,255);
      analogWrite(M2_VE,veloc);
      digitalWrite(M2_AV,LOW);
      digitalWrite(M2_RE,HIGH);
    }else if(RX_buffer[1]>NEUTRO_MAX){
      byte veloc=map(RX_buffer[1],NEUTRO_MAX,255,0,255);
      analogWrite(M2_VE,veloc);
```

```
      digitalWrite(M2_AV,HIGH);
      digitalWrite(M2_RE,LOW);
    }else {
      digitalWrite(M2_VE,LOW);
      digitalWrite(M2_AV,LOW);
      digitalWrite(M2_RE,LOW);
    }
    // Mueve Motor 3:
    if(RX_buffer[2]<NEUTRO_MIN){
      byte veloc=map(RX_buffer[2],NEUTRO_MIN,0,0,255);
      analogWrite(M3_VE,veloc);
      digitalWrite(M3_AV,LOW);
      digitalWrite(M3_RE,HIGH);
    }else if(RX_buffer[2]>NEUTRO_MAX){
      byte veloc=map(RX_buffer[2],NEUTRO_MAX,255,0,255);
      analogWrite(M3_VE,veloc);
      digitalWrite(M3_AV,HIGH);
      digitalWrite(M3_RE,LOW);
    }else {
      digitalWrite(M3_VE,LOW);
      digitalWrite(M3_AV,LOW);
      digitalWrite(M3_RE,LOW);
    }
    // Mueve Motor 4:
    if(RX_buffer[3]<NEUTRO_MIN){
      byte veloc=map(RX_buffer[3],NEUTRO_MIN,0,0,255);
      analogWrite(M4_VE,veloc);
      digitalWrite(M4_AV,LOW);
      digitalWrite(M4_RE,HIGH);
    }else if(RX_buffer[3]>NEUTRO_MAX){
      byte veloc=map(RX_buffer[3],NEUTRO_MAX,255,0,255);
      analogWrite(M4_VE,veloc);
      digitalWrite(M4_AV,HIGH);
      digitalWrite(M4_RE,LOW);
    }else {
      digitalWrite(M4_VE,LOW);
      digitalWrite(M4_AV,LOW);
      digitalWrite(M4_RE,LOW);
    }
  }
  else{
    // Entra en "FailSafe" en un segundo:
    if((millis()-failSafe)>1000){
      digitalWrite(M1_AV,LOW);
      digitalWrite(M1_RE,LOW);
      digitalWrite(M1_VE,LOW);
      digitalWrite(M2_AV,LOW);
      digitalWrite(M2_RE,LOW);
      digitalWrite(M2_VE,LOW);
      digitalWrite(M3_AV,LOW);
      digitalWrite(M3_RE,LOW);
      digitalWrite(M3_VE,LOW);
      digitalWrite(M4_AV,LOW);
      digitalWrite(M4_RE,LOW);
      digitalWrite(M4_VE,LOW);
    }
  }

} // Fin de void loop().
```

```
// FIN DE CÓDIGO.

// Fin de programa.
```

Aquí la cosa ya se va complicando un poquito. Hemos definido una zona neutra en el centro de los joysticks para que los motores estén parados cuando los joysticks estén en el centro. Hay cuatro bloques de código muy similares, uno para cada motor. Cuando un joystick sale de su zona neutra, se mira en qué sentido se ha movido para saber en qué sentido hay que hacer girar al motor y cuanto está desplazado desde el centro para saber a qué velocidad. También hemos definido una tercera variable como "unsigned long" (32 bits sin signo) a la que llamamos "failSafe" y que cargamos con el contador de milisegundos del procesador "millis()". Cada vez que recibimos correctamente una trama, recargamos esta variable. Cada vez que no recibimos una trama, la comparamos con el estado actual del contador y, si ha pasado más de un segundo, paramos todos los motores. Esta es una función de seguridad que debe incorporar cualquier sistema controlado por radio para detener los motores si se pierde el enlace durante más de un segundo. Se suele conocer como "Fail Safe" (Seguro ante Fallo).

Receptor 4: teledirigir un vehículo

Y ya por último, vamos a ver cómo podemos utilizar nuestro Telemando Digital Universal para dirigir un vehículo. Suponemos que tenemos un motor de tracción cuya velocidad y sentido controlamos con el movimiento en el eje "Y" del joystick A (el izquierdo). También tenemos un motor para mover la dirección, acoplado a un potenciómetro, que actúa como codificador rotativo y nos cierra el bucle, que movemos con el eje "X" del joystick B (el

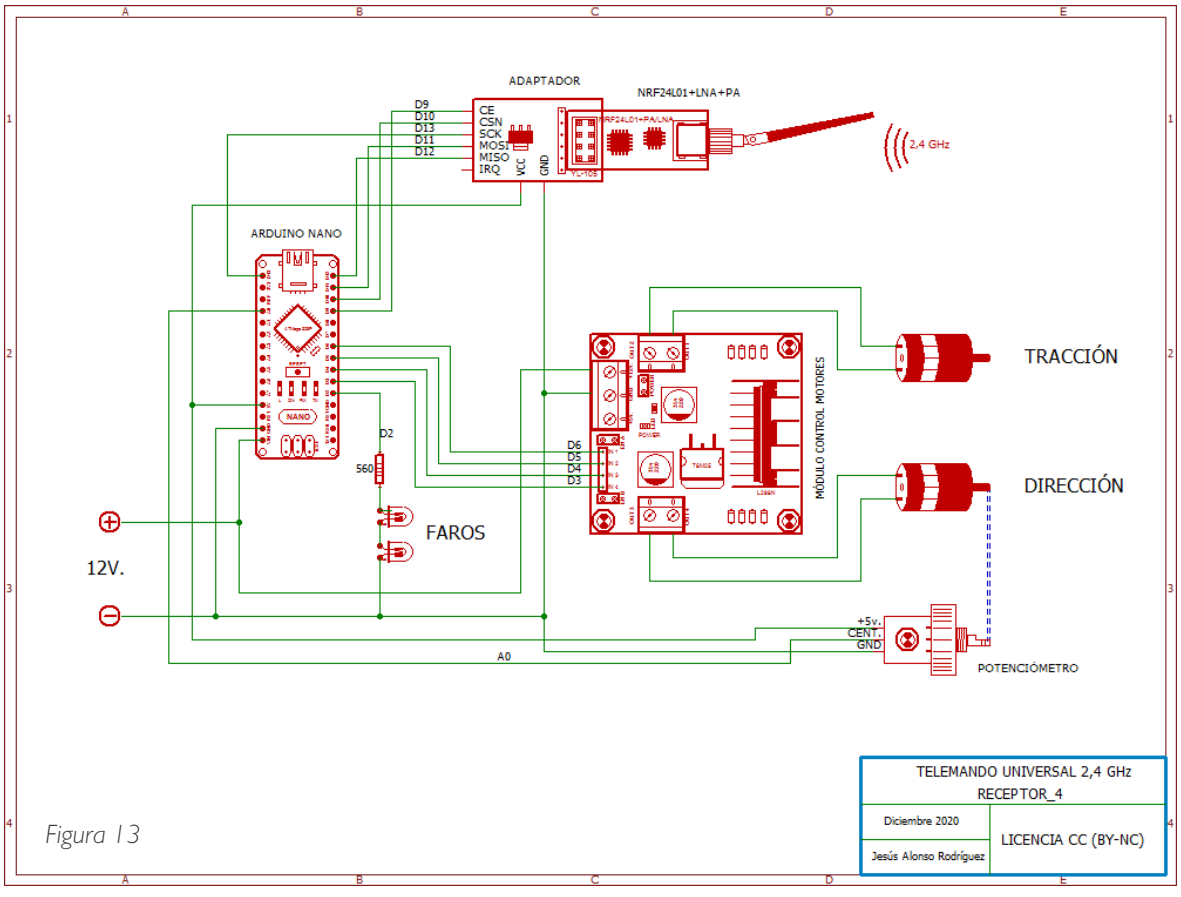


Figura 13

derecho). También le pondremos unos faros que encenderemos con el potenciómetro A (el izquierdo) o con el conmutador izquierdo. Vemos el diagrama de conexiones en la Figura 13.

Y este es el programa que le hace funcionar:

```
// Telemando Universal Meccano.
// Receptor_4 2,4 GHz
//
// Maneja un vehículo.
//
// Versión: 1.0
// Diciembre de 2020
// (c) Jesús Alonso Rodríguez
// Licencia: Creative Commons (by-nc).
// Esta obra está licenciada bajo la Licencia Creative Commons
// Atribución-NoComercial 4.0 Internacional.
// Para ver una copia de esta licencia, visita
// http://creativecommons.org/licenses/by-nc/4.0/.

// COMPROBADO.

// BIBLIOTECAS:
#include <SPI.h>
#include <RF24.h>
// FIN DE BIBLIOTECAS.

// CONSTANTES:
#define CE_PIN 9
#define CSN_PIN 10
#define MOT_A 6 // Motor avance.
#define MOT_R 5 // Motor retroceso.
#define DIR_I 4 // Dirección a izquierda.
#define DIR_D 3 // Dirección a derecha
#define FAR 2 // Faros.
#define POS_DIR A0 // Posición de la dirección.
#define DIR_IZQ 250 // Valores de ejemplo, los
#define DIR_CEN 470 // verdaderos hay que determinarlos
#define DIR_DER 690 // para cada modelo.
#define NEUTRO 0 // Punto muerto.
#define ADELANTE 1 // Marcha adelante.
#define ATRAS 2 // Marcha atrás.
#define NEUTRO_MIN 123 // Neutro mínimo de cada joystick.
#define NEUTRO_MAX 133 // Neutro máximo de cada joystick.
#define UMBRAL 200
// FIN DE CONSTANTES.

// OBJETOS:
RF24 radio(CE_PIN,CSN_PIN);
// FIN DE OBJETOS.

// VARIABLES GLOBALES:
byte direccion[5]={0,0,0,1,5};
byte RX_buffer[7];
unsigned long failSafe;
int dir=DIR_CEN;
int angulo=analogRead(POS_DIR);
```

```
byte mar=NEUTRO;
int velocidad=0;
boolean faros=false;
// FIN DE VARIABLES GLOBALES.

// CÓDIGO:
void setup() {
  pinMode(DIR_I,OUTPUT);digitalWrite(DIR_I,LOW);
  pinMode(DIR_D,OUTPUT);digitalWrite(DIR_D,LOW);
  pinMode(MOT_A,OUTPUT);digitalWrite(MOT_A,LOW);
  pinMode(MOT_R,OUTPUT);digitalWrite(MOT_R,LOW);
  pinMode(FAR,OUTPUT);digitalWrite(FAR,LOW);
  radio.begin();
  radio.setPALevel(RF24_PA_MAX);
  radio.setChannel(110);
  radio.setPayloadSize(7);
  radio.setRetries(0,5); // 250us/5 retries.
  radio.openReadingPipe(1,direccion);
  radio.startListening();
}

void loop(){
  if(radio.available()){
    failSafe=millis();
    radio.read(RX_buffer,7);
    // Fija velocidad:
    if(RX_buffer[1]<NEUTRO_MIN){
      mar=ATRAS;
      velocidad=map(RX_buffer[1],NEUTRO_MIN,0,0,255);
    }else if(RX_buffer[1]>NEUTRO_MAX){
      mar=ADELANTE;
      velocidad=map(RX_buffer[1],NEUTRO_MAX,255,0,255);
    }else{
      mar=NEUTRO;
      velocidad=0;
    }
    // Fija dirección:
    if(RX_buffer[2]<NEUTRO_MIN){
      dir=map(RX_buffer[2],NEUTRO_MIN,0,DIR_CEN,DIR_IZQ);
    }else if(RX_buffer[2]>NEUTRO_MAX){
      dir=map(RX_buffer[2],NEUTRO_MAX,255,DIR_CEN,DIR_DER);
    }else{
      dir=DIR_CEN;
    }
    // Enciende/apaga faros:
    if((RX_buffer[4]>UMBRAL)||(bitRead(RX_buffer[6],2)==0)){
      faros=true;
    }else{
      faros=false;
    }
  }else{
    // Entra en "FailSafe" en un segundo:
    if((millis()-failSafe)>1000){
      mar=NEUTRO;
      velocidad=0;
      dir=DIR_CEN;
      faros=false;
    }
  }
```

```
  if(mar==ADELANTE){
    analogWrite(MOT_A,velocidad);
    }
  if(mar==ATRAS){
    analogWrite(MOT_R,velocidad);
    }
  if(mar==NEUTRO){
    analogWrite(MOT_A,0);
    analogWrite(MOT_R,0);
  }
  digitalWrite(FAR,faros);
  angulo=analogRead(POS_DIR);
  if(angulo<dir){digitalWrite(DIR_D,HIGH);}
  else{digitalWrite(DIR_D,LOW);}
  if(angulo>dir){digitalWrite(DIR_I,HIGH);}
  else{digitalWrite(DIR_I,LOW);}
} // Fin de void loop().

// FIN DE CÓDIGO.

// Fin de programa.
```

El potenciómetro que mueve la dirección (entre 5K y 100K) hay que conectarlo de manera que dé el valor más bajo cuando la dirección esté a la izquierda y el más alto cuando esté a la derecha (el valor de DIR_CEN será la media entre los dos). Los valores de las constantes DIR_IZQ, DIR_CEN y DIR_DER habrá que determinarlos experimentalmente, ya que cambian para cada modelo. Por supuesto que esto se podría hacer con un servo, pero se pierde la posibilidad de que gire el volante al moverse la dirección, que queda muy real. Típicamente, el motor moverá la columna de la dirección que, a través de un sinfín y un piñón, moverá la biela que actúa sobre las manguetas. El potenciómetro deberá acoplarse solidariamente con este piñón, de forma que su movimiento seguirá el movimiento de las manguetas. Finalmente, en este tipo de aplicación es todavía más importante implementar la función “Fail Safe” para evitar que el vehículo se nos escape si sale de la zona de cobertura.

Hasta aquí la explicación del Telemando Digital Universal. Espero que resulte útil, no sólo para hacer magníficos modelos radiocontrolados sino también para aprender un poquito de electrónica.

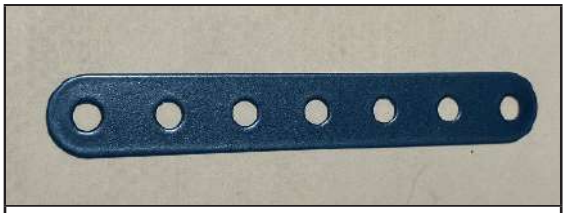
Se trata de un proyecto bastante complejo y es perfectamente normal que no se entienda completamente en una primera lectura. El que esté interesado tendrá que aplicarse un poquito al estudio, pero estoy seguro de que el resultado valdrá la pena. Como de costumbre, estaré encantado de resolver dudas y preguntas en el chat de Watsapp o en la dirección de correo: **jalonsovivienda@gmail.com**

Las nuevas piezas metalicas Meccano desde 1983

Antonio Valero Aicua

Desde que cerraron las fábricas inglesas de Liverpool y la española de Barcelona en 1980, solo se fabricaba Meccano original en Francia. Desde entonces han venido apareciendo muchas piezas nuevas metálicas y también de plástico, papel, goma y madera. En esta ocasión me voy a centrar en las nuevas piezas metálicas que han sido muy numerosas y muchas de ellas muy prácticas e interesantes.

Tiras

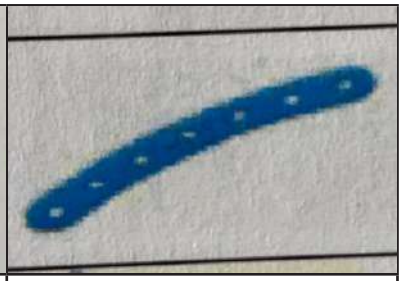


B666 Tira de 8 agujerosw

Tiras curvas

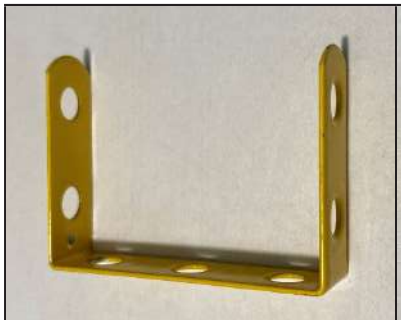
B205 Tira curva 8 agujeros sin escalón

C667 Tira curva 4 agujeros



C258 Tira curva 7 ag.

Tiras dobladas



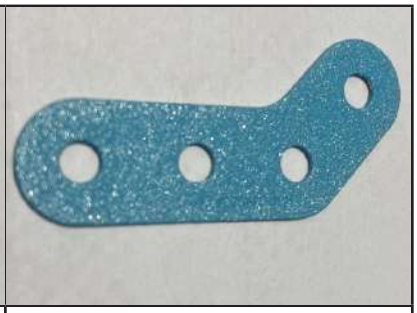
46 a Tira doblada 3 x 2 ag.

48 e Tira doblada 3 juntos x1 ag.

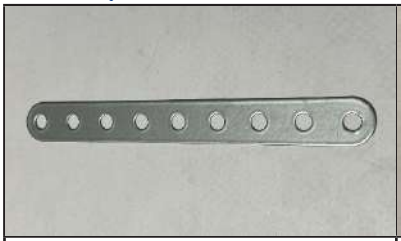
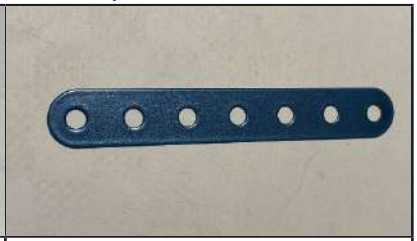
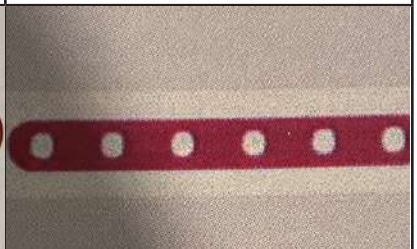
47 b Tira doblada 7 x 3 ag, 3 juntos

	
11b Tira doblada 5 x 1 en ag centrales	Tira doble dobladura 2 ag.

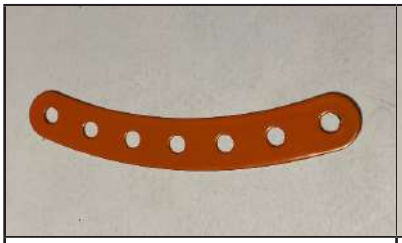
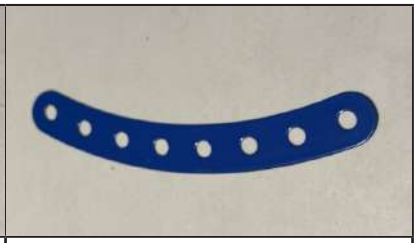
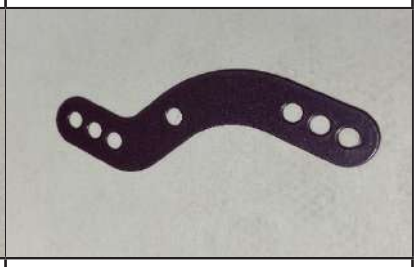
Tiras especiales

	
133b Tira en ángulo recto	133c Tira en ángulo obstuso

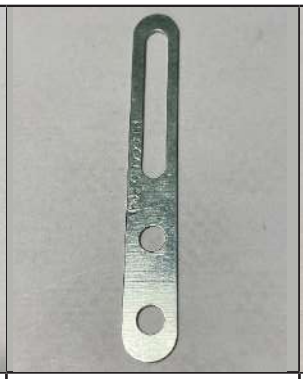
Tiras flexibles (con efecto memoria)

	
B482 Tira 9 ag	B488 Tira 7 ag
	
B487 Tira 5 ag	C190 Tira estrecha 6 ag

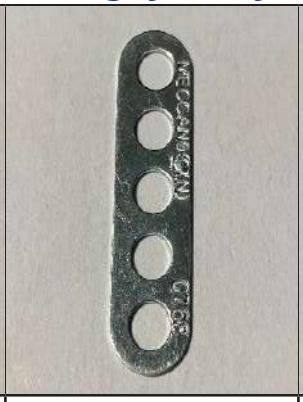
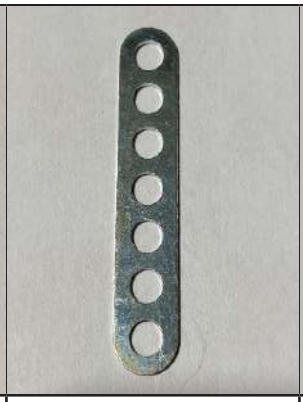
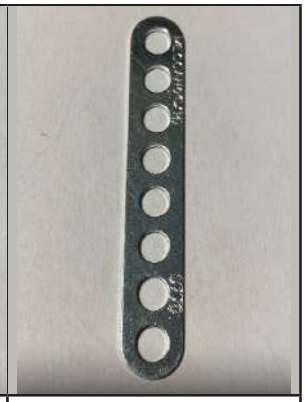
Tiras curvas flexibles

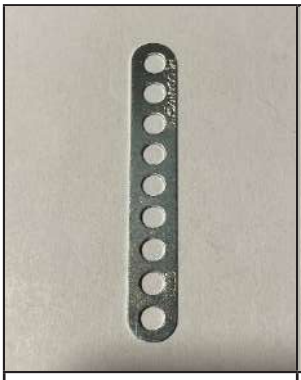
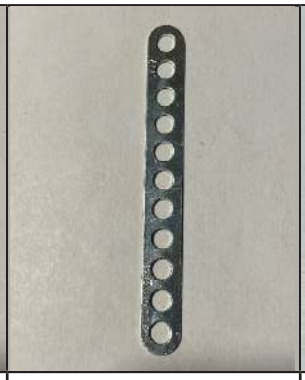
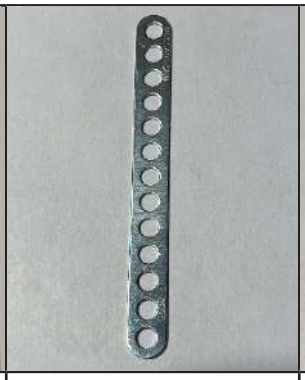
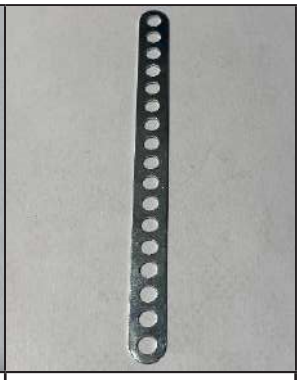
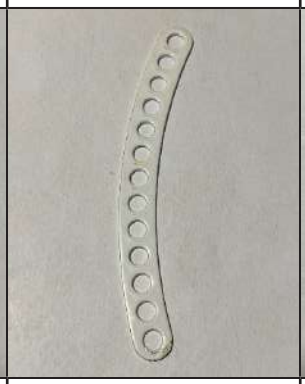
	
B856 7 ag.	B589 8 ag
	
C947 5 ag	B572 Inversa 7 ag

Tiras estrechas

			
806b Tira 2 ag	C329 Tira 3 ag	B698 Tira con muesca	Tira doblada 5 x 1 ag

Tiras estrechas de agujeros juntos

			
C329 3 ag	C768 5 ag	C769 7 ag	C770 8 ag

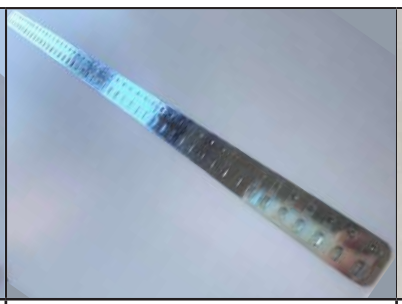
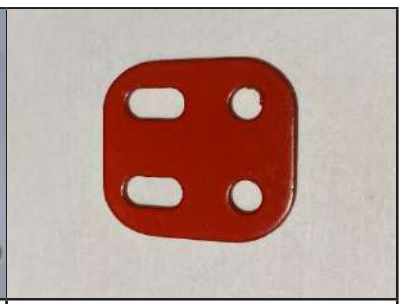
			
C771 9 ag	C772 11 ag	C773 13 ag	C774 17 ag
			
C777 Curva de 9 ag	C779 Curva 13 ag	C776 ángulo recto 5x3	C775 Obtuso

Viguetas

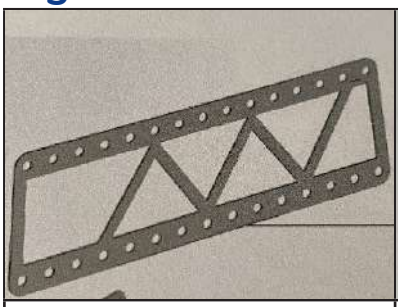
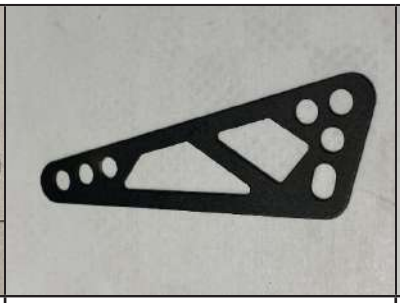
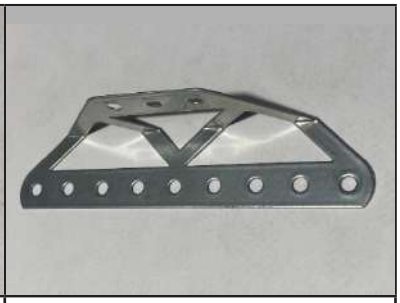
Vigueta angular


9L Angular 2 ag

Viguetas planas

		
103t Plana 37 ag.	103r Plana 49 ag.	103 L Plana 2 ag.

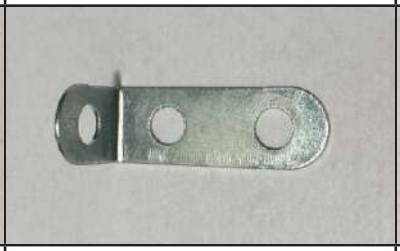
Viguetas caladas

		
C699 15 ag brazos simples	C472 Triangular	C371 Doblada

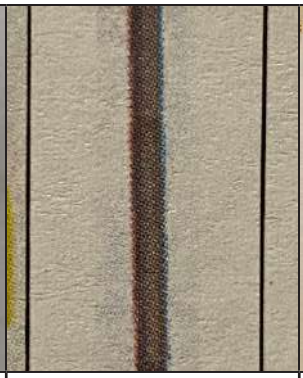
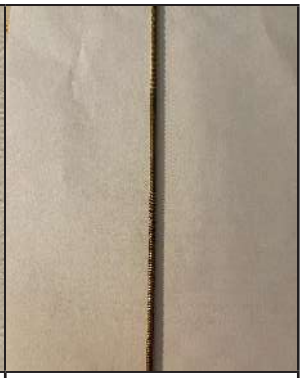
Soportes


12 d Angular obtuso 1 ag.

Soportes estrechos

		
811 Doble	C330 Angular	C483 Obtuso
		
812d Angular 2 ag obtuso	812b Angular 2 ag	825a Inverso
		
811a Doble 2 ag	C331 Doble obtuso	825 inverso obtuso

Ejes

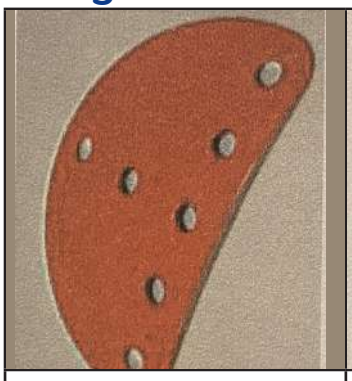
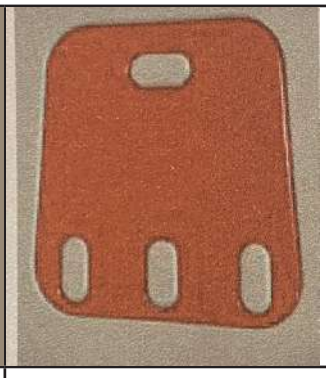
			
Ejes triangulares de 2,5 3 3,8, 5 6 7,5 9 10 11,5,13 cm 318a, 317, 316b, 316a, 316, 315b, 315a, 31	115d Clavija larga tuerca	B917 Eje rosca ambos lados	78c Eje roscado 24 cm

Engranajes

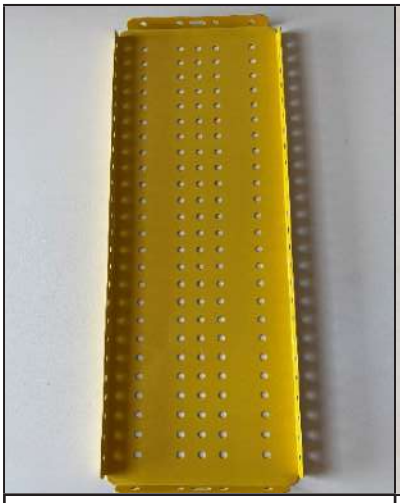
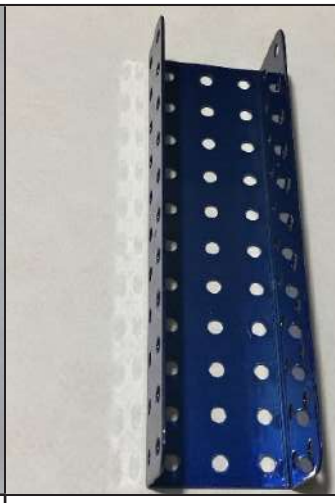
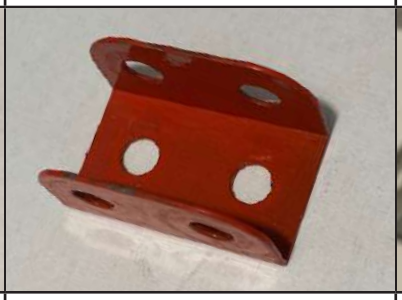
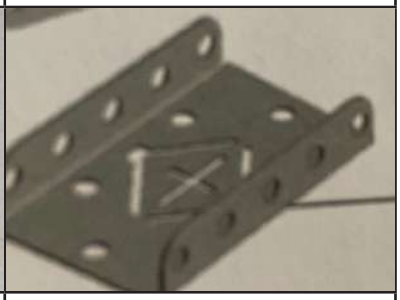
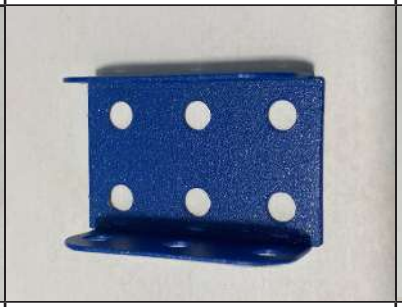
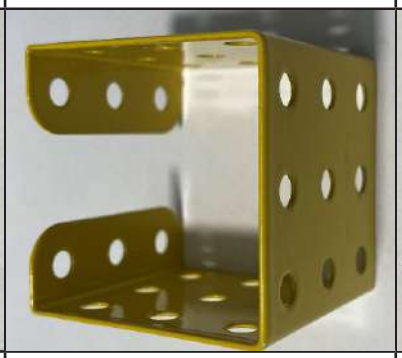
	
26 n Piñon 11 dientes	25 c Rueda 25 dientes

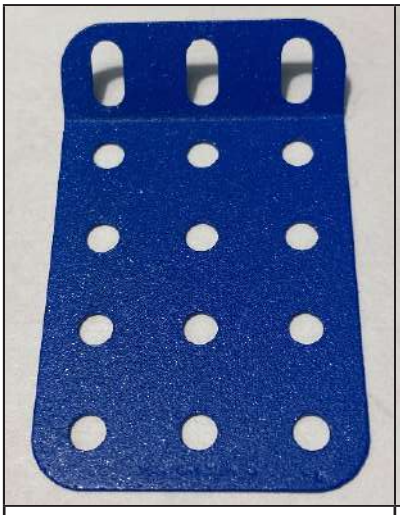
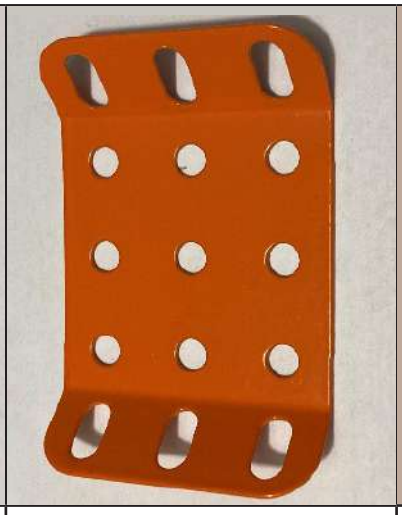
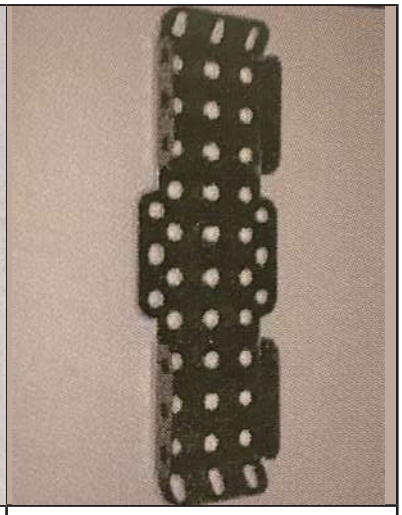
Placas

Placas rígidas

		
C616 Placa Semicircular curvada	C617 Placa 3 x 1 ag	C618 Placa especial

Placas rebordeadas

		
236 Placa tapa 25 x 5 x 1 ag	160d Placa 11 x 3 x 1 ag	51 f Placa 5 x 3 x 1 ag
		
45a Placa 2 ag, y lengüetas	51c Placa 2 ag.	C700 Placa especial con ranuras
		
51 b Placa 3 x 3 x 1 ag	51 a Placa 3 x 2 x 1 ag	51 d Placa 3 x 2 x 2 ag
		
51 e Placa 5 x 3 x 1 ag corto	160g Placa 3 x 3 ag doble	C369 Placa trapezidal 5 y 2 ag

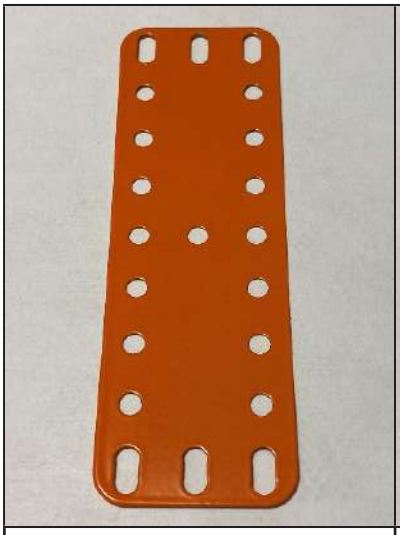
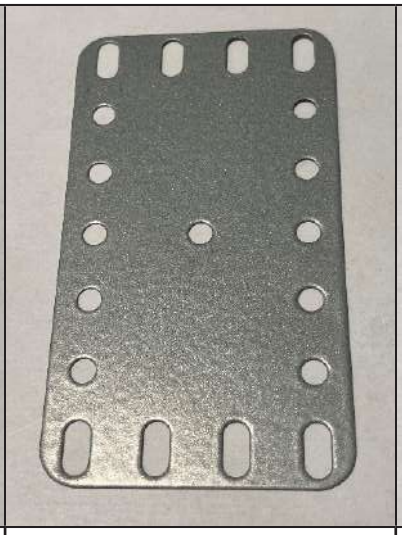
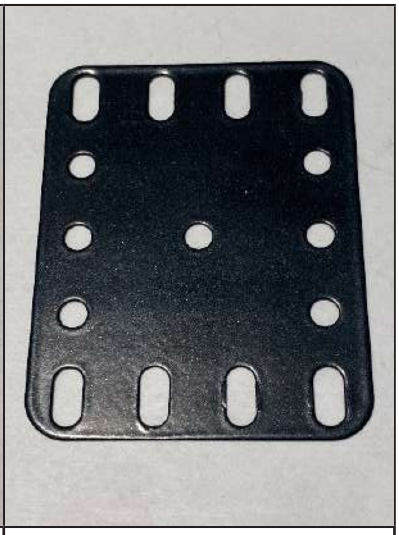
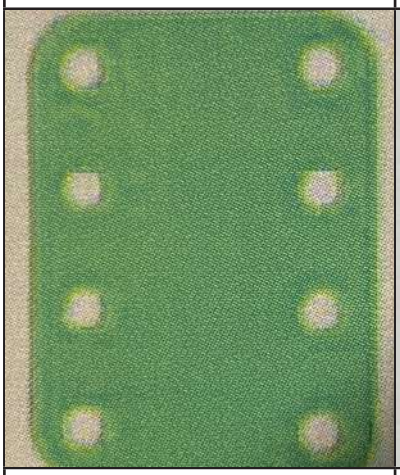
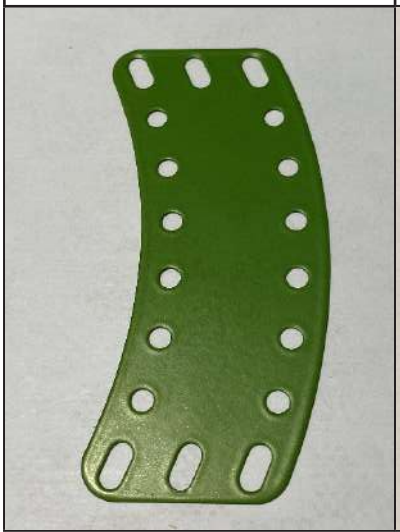
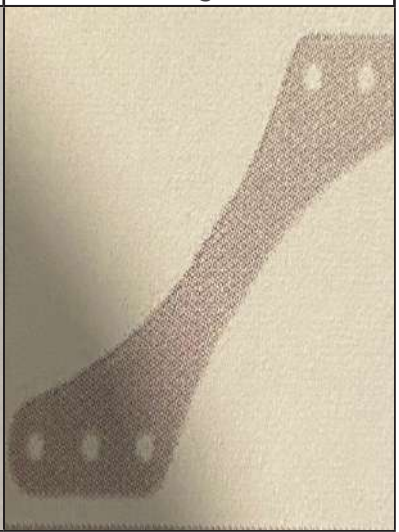
		
73 a Placa reborde obtuso 3 x 4 ag	51 g Placa rebordes obtusos 3 x 3 ag	C045 Placa chasis

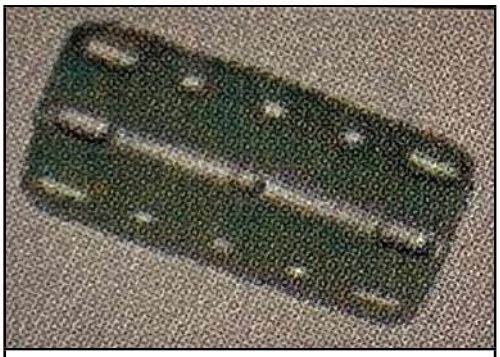
Placas triangulares abombadas

		
C323 Placa 1 x 2 ag	B684 Placa 2 x 3 ag	B484 Placa 3 x 3 ag

	
B695 Placa 3 x 4 ag	B970 Placa trapezoidal 5 y 3 ag

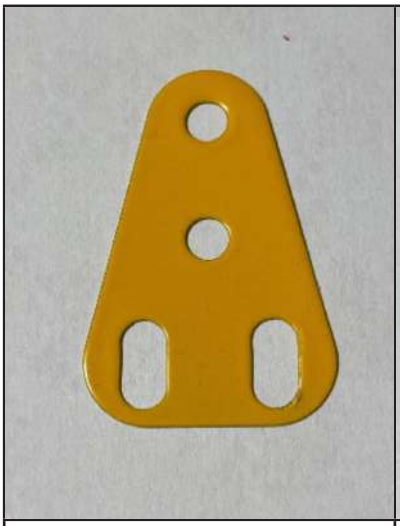
Placas flexibles

		
B664 Placa 9 x 3 ag	B664 Placa 9 x 3 ag	B665 Placa 5 x 4 ag
		
B583 Placa 7 x 4 ag	200d Placa curva 9 x 3 ag	C243 Placa curva de 7 x 2 ag
		
B709 Placa curva 8 x 3 ag	B969 Placa inversa	C240 Placa inversa estrecha



C061 placa en u 5 x 3 ag

Placas flexibles triangulares



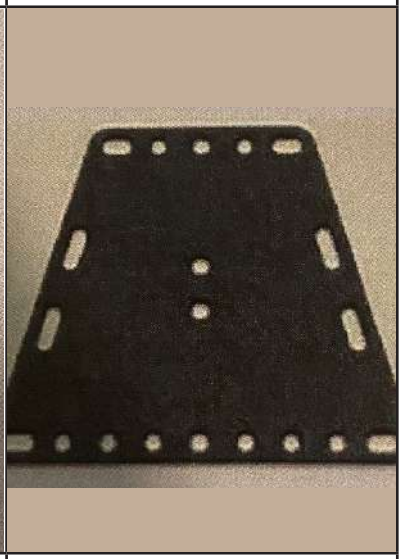
B608 Placa 2 x 2 ag

B581 Placa 2 x 1 ag

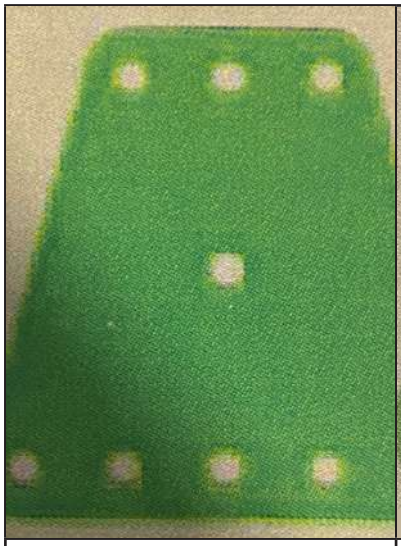
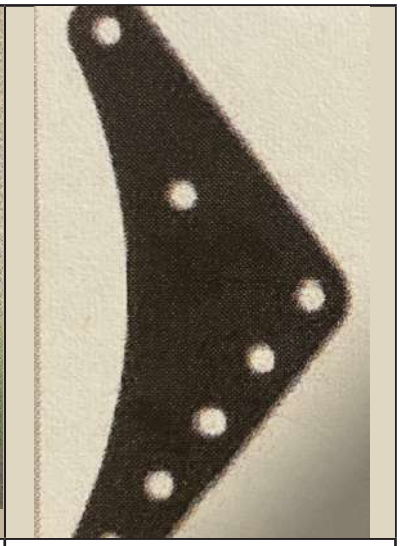
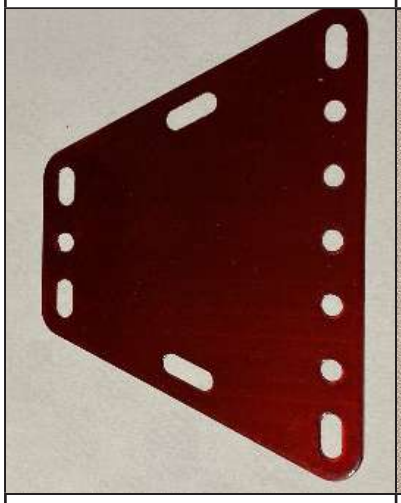
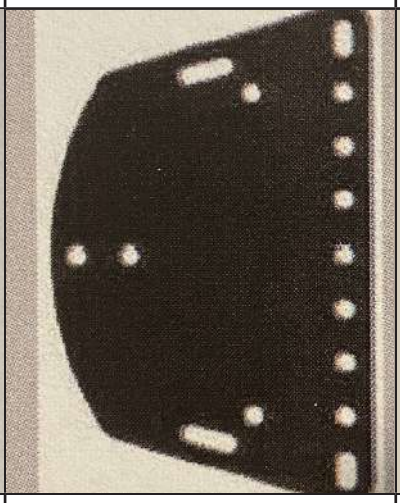
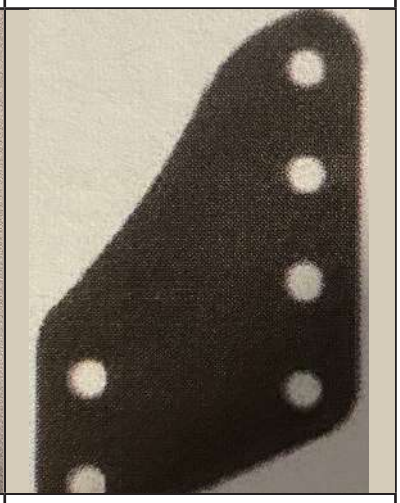
B582 Placa 3 x 2 ag

C191 Placa 5 x 3 ag

C192 Placa 7 x 2 ag



B489 Placa 9 x 5 ag

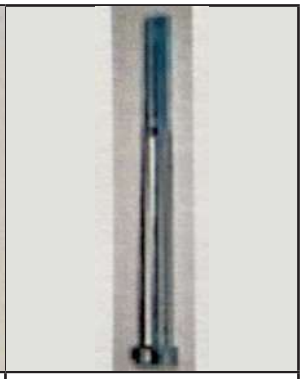
		
B863 Placa 5 x 3 ag central	*B865 Cantonera 3 x 3 ag*	*C241 Cantonera 5 x 2 ag*
		
B480 Placa 7 x 3 ag	*C244 Placa especial*	*C242 Placa especial*

Varios

		
120 d Amortiguador	*B923 Muelle comprensión corto*	*120dr Muelle comprensión grande*

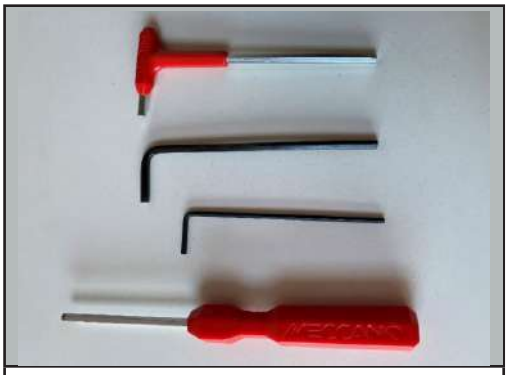
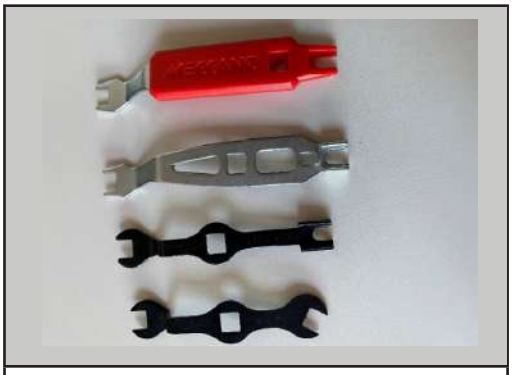
		
B663 Cono abierto	B673 Cono	216a Cilindro abierto
		
213D Empalme ejes corto	64a Acoplamiento roscado corto	B741 Cono hélice avión
		
187g Tapa cubo pequeño	Polea 38 mm sin buje	162d Tapa caldera borde bajo
		
C951 Sección caldera	537a Imán	B708 Cable para comandar

Tornillos y Tuecas

			
Tonillos negros allem 6,9,12,19,30 y tuerca. *37,111c, 111a, 111,111e,111d*	*C935 B696 Tornillos pivotantes allem 31 y 40 mm*	*Tornillos pivotantes allem 16,20,25,35,40 mm.* *147d, 147f, 147g B256*	*M312 45 y 66 mm*

		
Tornillos allem *6,9,12,19,25,28 mm* *37,111c, 111a, 111,111e,111d* *Tuerca cuadrada*	*37h Tuerca autoblocante*	*69 c Invisible allem*

Herramientas

	
C994, 36 d, 36 c Llaves allem	*34 C088 C935 C993 B499 Llaves tuercas cuadradas*

Algunas de estas nuevas piezas están contenidas en los modernos equipos para modelos concretos como automóviles radiocontrolados, torre Eiffel, grúa de obra, edificios y otros, pero son totalmente compatibles con el resto de las piezas Meccano.

Además de las piezas que se muestran existen las mismas en otros colores, algunos metalizados.

Hay vendedores de piezas sueltas, todos por internet, que tienen estas nuevas piezas. Por el contrario Meccano France no vende al público piezas sueltas desde hace bastante tiempo.

Automóvil aleman antiguo. Modelo Benz. Año 1904.

José Enríquez Victoria

Karl Benz desarrolló su primer automóvil en 1885 al instalar un motor monocilíndrico de 954 c.c. y 0.75 caballos de potencia en un chasis diseñado para este motor. La velocidad máxima que alcanzaba era de 16 km/h.

Al año siguiente, el 29 de enero de 1886 Karl Benz estaba construyendo lo que sería el primer automóvil propulsado por un motor de combustión interna a gasolina: El BENZWAGEN.

En este mismo año, su modelo recibió el número 37435 de patente en la Oficina Imperial Alemana de Patentes de Berlín, quedando de este modo registrado su vehículo de tres ruedas. También en 1886, e independientemente de Benz, Gottlieb Daimler construía un coche motorizado. Ambos acontecimientos marcan la historia de 130 años de éxitos de Mercedes-Benz.

La primera aparición pública de este vehículo de tres ruedas fue el 3 julio de 1886 en Ringstrasse, en la localidad alemana de Mannheim.

Dos meses más tarde, en agosto de 1886, Bertha, la mujer de Benz, acompañada de dos de sus hijos (Foto nº 1) sin decir nada a su marido viajó desde Mannheim hasta Pforzheim. Hasta ese momento solo se había probado el vehículo en trayectos cortos hasta que Bertha hiciese ese viaje de 104 kilómetros y que ha pasado a la historia como el primer viaje más largo en coche de la industria automovilística.

Foto nº 1: Bertha y sus hijos al inicio del viaje.

Bertha (1849- 1944) se casó con Karl Benz en 1872 y destinó parte de su fortuna para financiar la nueva empresa Benz y Cia. Los primeros años del matrimonio fueron muy duros, incluso llegaron a pasar hambre.

Como curiosidad 16 años mas tarde Emilia de Pardo Bazán, considerada la mejor novelista española de sigo XIX, decidió subirse a un automóvil convirtiéndose así en la primera mujer española en conducir.

¿Quién era Karl Benz?

(Karl Benz, Kalsruhe, Alemania 1844 – Landenburg 1929), fue un ingeniero alemán que diseñó el primer automóvil impulsado por un motor de combustión interna (1885).

Al igual que otros pioneros de la automoción (Henry Ford, Giovanni Agnelli, André Citroën, Louis Renault) fundó y dirigió su propia empresa, (la casa Benz radicaba en Mannheim), desde la que contribuyó al despegue de la industria automovilística en la segunda etapa de la Revolución Industrial.

Hijo de un conductor ferroviario, Karl Benz realizó en 1877 sus primeros experimentos sobre el motor de combustión. En 1883 fundó en Mannheim la Benz y Cia.
El primer coche fabricado por la empresa fue un triciclo al que llamó MOTORWAGEN, que actualmente se conserva en Munich.

Foto nº 2

El prototipo Benz era una especie de carro con tres ruedas. La dirección se controlaba mediante un sistema de piñón y barra de cremallera. Contaba con una única marcha y la fuerza se transmitía del motor a las ruedas mediante el uso de una cadena. (Foto nº 2).

En 1893, la casa Benz y Cia diseñó y fabricó con características similares al anterior, su primer vehículo de cuatro ruedas.

Posteriormente en 1926 la casa Benz y Cia se fusionó con la Daimler Motoren de Gottlieb Daimler, creando la firma productora de Mercedes-Benz que ha llegado hasta nuestros días.

El modelo Meccano

Algunas consideraciones sobre el modelo:

En la foto se ve que el modelo está en alzado, pero lo he probado sin él y funciona perfectamente, teniendo en cuenta que la rueda delantera hacia adelante gira sin control y se atasca, con lo cual hay que sujetarla firmemente.

En las exposiciones donde el espacio es muy limitado es conveniente poner los modelos en alzado.

Fotos nº 3 y 4º. Vista general

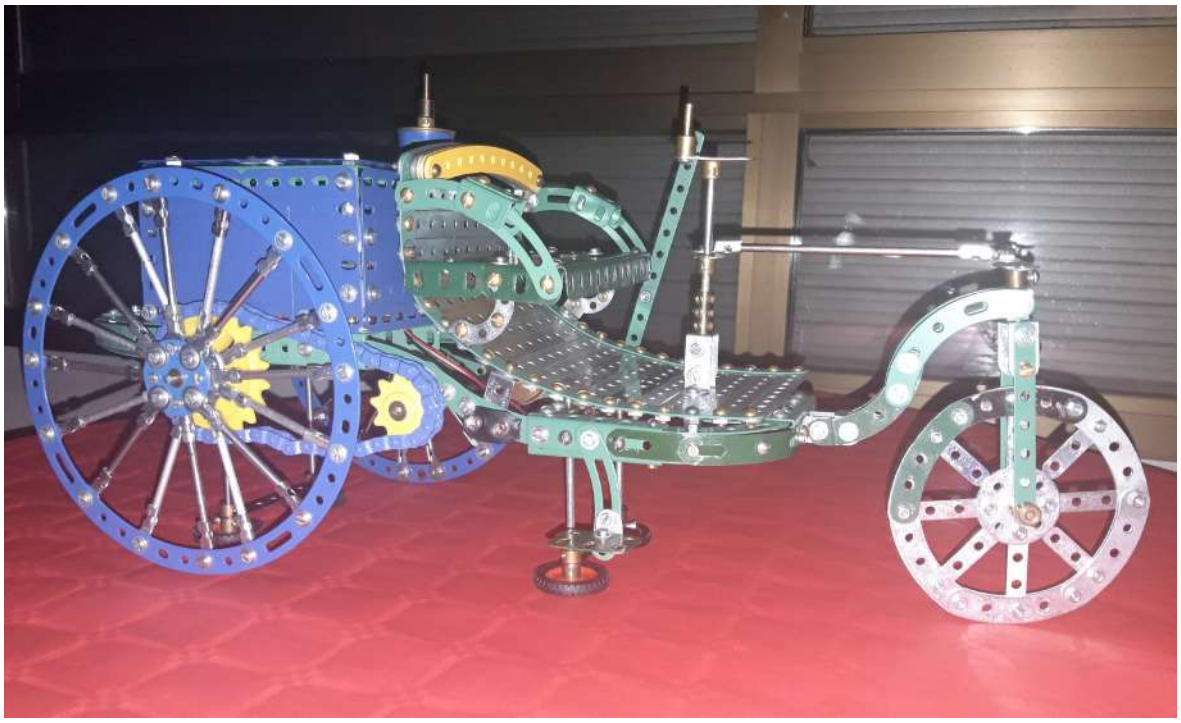


Foto nº 3

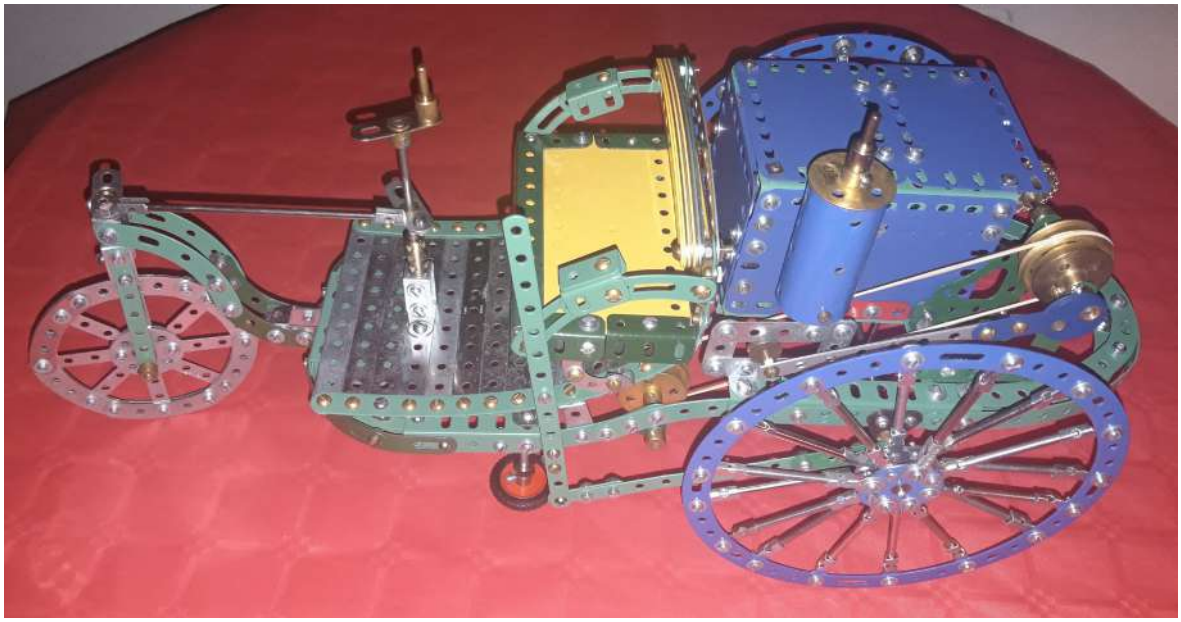


Foto nª 4

En estas dos vistas generales se aprecia la similitud del modelo Meccano con el original. Es un modelo muy vistoso y bastante fácil de hacer. Lo único que hay que tener son las tiras circulares (Nº de Meccano 145) para poder hacer las ruedas.

Fotos nº 5 y 6. Ruedas Motrices

Si nos fijamos en la fotos nº 5 y 6 Ruedas Motrices es lo más complejo del modelo sin serlo demasiado. He usado dos ruedas con buje de 8 ag. por un lado para formar 8 radios y por el otro lado lo mismo y en su eje ponerlas de manera que no coincidan y así tener 16 radios. En esta misma foto se ve la cadena y sus ruedas. He puesto cadena y ruedas de plástico, pues al ser más gruesas que las metálicas van mejor para este modelo.

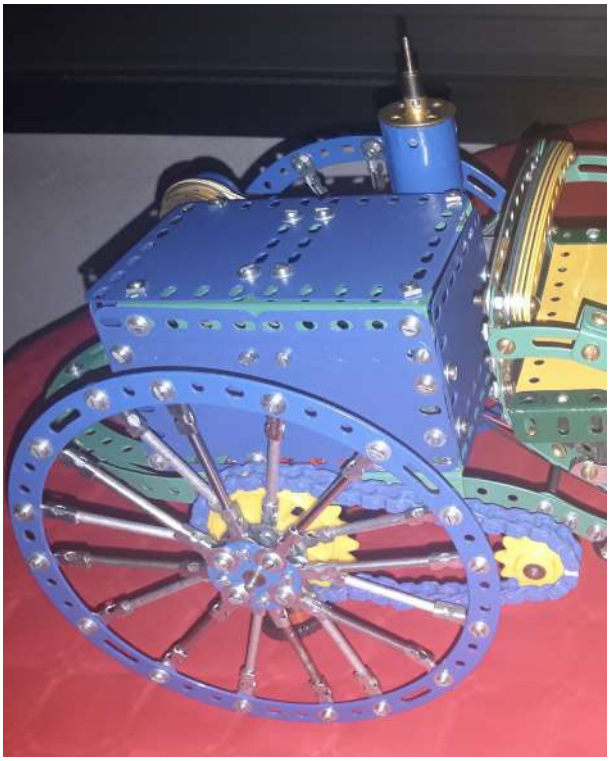


Foto nª 5

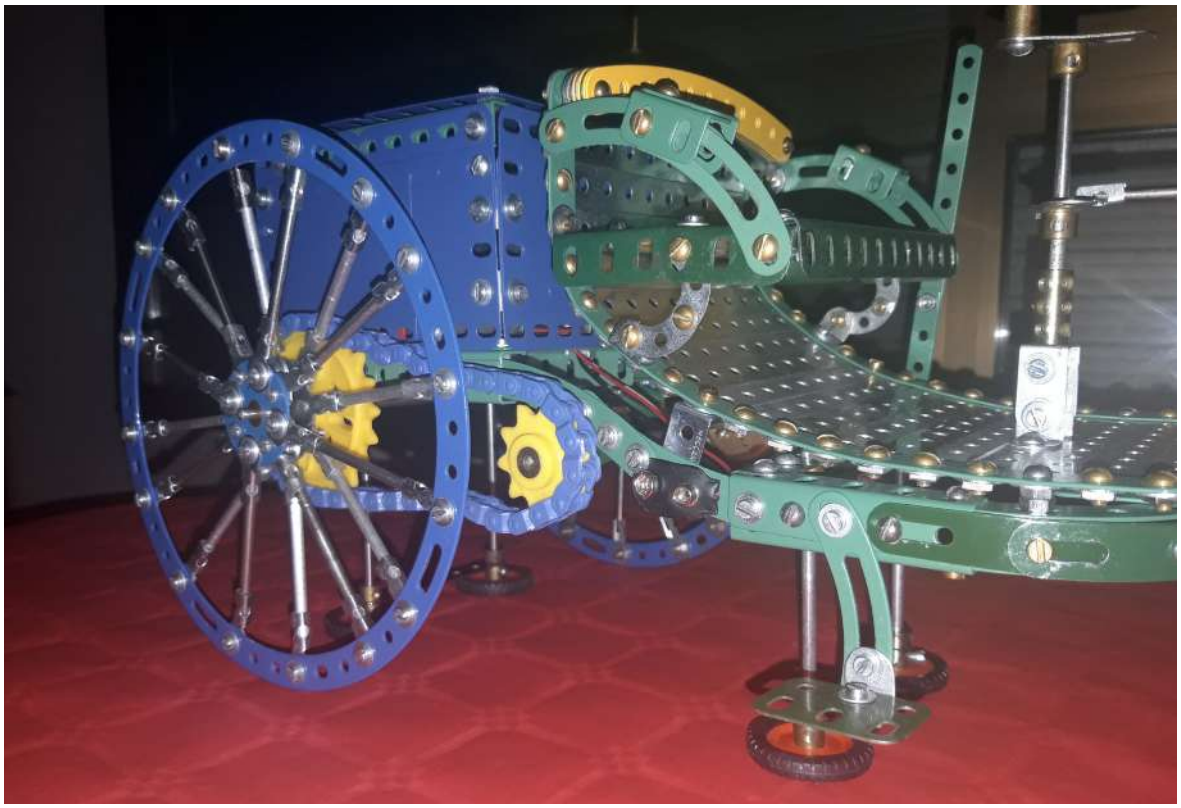


Foto nº 6

Foto nº7. La transmisión del motor a las ruedas
El movimiento de vaivén lo produce una excéntrica de 2 pasos
Ya que tengo poco espacio para hacer un recorrido más grande he utilizado dos ruedas rebordeadas de 28 mm. y en el centro una polea de 38 mm, consiguiendo de este modo una eficiente transmisión. Una goma elástica cierra este movimiento.

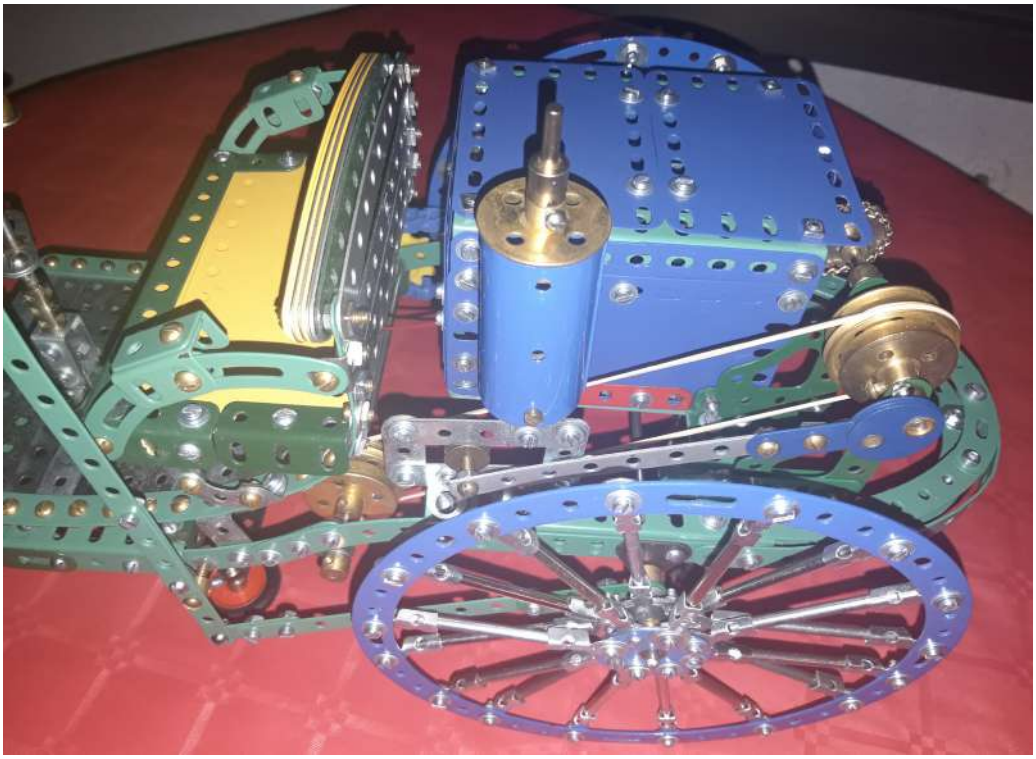


Foto nº 7

Foto nº 8. La rueda delantera

La rueda delantera como se ve en la imagen no forma un circulo, lo ideal sería haber puesto una tira circular de ese tamaño o aproximadamente, pero carecía de ella así como de una tira curvada de color cinc, a pesar de ello el modelo se resiente poco.

Foto nª 8

Foto nº 9. El chasis

Por último una foto del chasis por si alguno tiene la tentación de construir este modelo, también se ve el motor que he puesto.

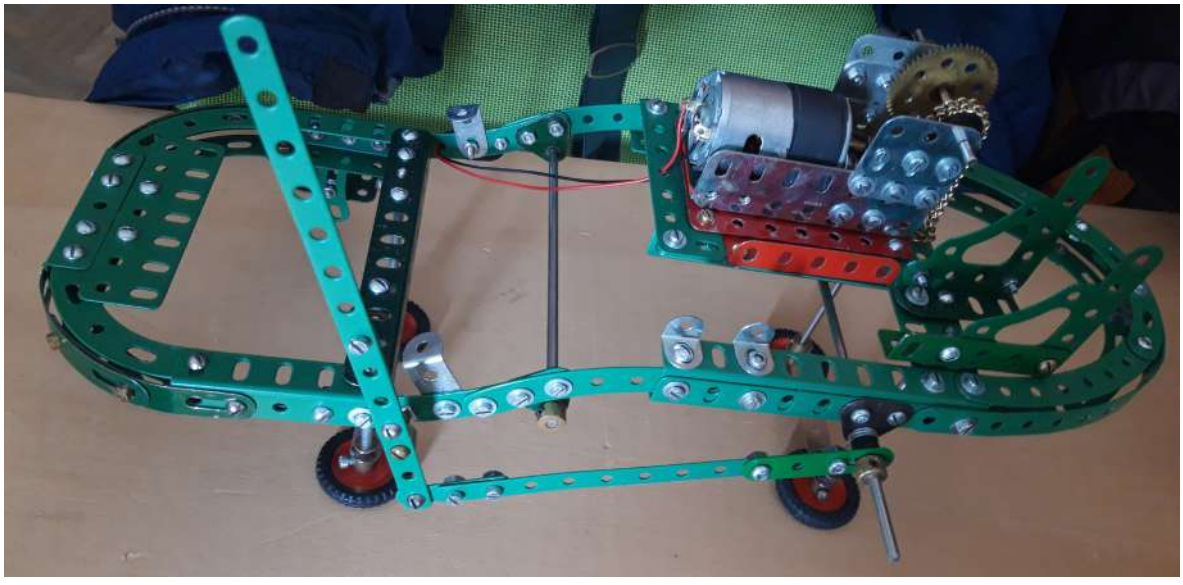


Foto nª 9

EQUIPO TRONICO "TITANIC"