



industriales
etsii

**Escuela Técnica
Superior
de Ingeniería
Industrial**

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

**Escuela Técnica Superior de Ingeniería
Industrial**

Sistema de Automatización basado en LEGO EV3

TRABAJO FIN DE GRADO

**GRADO EN INGENIERÍA INDUSTRIAL ESPECIALIZADA EN
ELECTRÓNICA Y AUTOMÁTICA**

Autor: Iván Fernández Herrero
Director: Héctor Puyosa Piña

Cartagena, 18/04/2017



**Universidad
Politécnica
de Cartagena**

AGRADECIMIENTOS.

Con este trabajo de fin de grado finaliza una etapa de mi vida y comienza una nueva, comencé esta carrera sin saber si la iba a terminar. En aquel entonces, ingeniería me daba mucho respeto debido a que nunca he sido un alumno de sobresalientes ni siquiera en las famosas asignaturas de matemáticas y física que son en las que más nos basamos a la hora de enfrentarnos a una carrera tal, como es ingeniería.

Después de 4 duros e intensos años he descubierto que las barreras nos las ponemos nosotros mismos y que con motivación y esfuerzo podemos conseguir cualquier objetivo que nos propongamos. Estos años han influido enormemente en mi perspectiva de ver el mundo, se puede decir que la ingeniería ha despertado en mi la curiosidad de todo lo que puedo percibir y no puedo evitar hacerme esa pregunta típica de cualquier ingeniero “¿Por qué esto es así?” e investigar sobre ello o incluso mejorarlo.

Todo esto no hubiera sido posible sin el apoyo, energía y tiempo que me ha brindado mi familia, ya que sin ellos estar hoy donde estoy no hubiera sido posible.

También quiero agradecer a mis compañeros de universidad por todo el apoyo y ayuda recibida y todas las aventuras que hemos vivido durante estos años, particularmente a mis amigos Antonio Francisco, Varinia, Alejandro Cutillas, Ana y Héctor que, aunque este año vayamos terminando nuestra carrera, no tengo la menor duda que vamos a seguir en contacto.

Además, quiero agradecer a mi tutor de proyecto Héctor Puyosa Piña, por su ayuda y dedicación en la elaboración del presente proyecto, a Pablo por su ayuda en el laboratorio de automática y a otros profesores que han influido en mi como José Alfonso Vera, Ana Toledo, Antonio Guerrero y Francisco Ortiz.

Muchas gracias a todos,

Iván Fernández.

Contenido

1. Introducción.....	7
1.1. La Automatización.....	7
1.2. LEGO MINDSTORMS	8
1.2.1. Características físicas del LEGO EV3.....	9
1.3. RobotC.....	11
1.4. Arduino.....	12
1.4.1. Características físicas de Arduino.....	12
1.5. Objetivos del proyecto.....	13
1.5.1. El proyecto.....	13
2. Estado del Arte.....	14
2.1. Protocolo de comunicación I ² C.....	14
2.1.1. El Protocolo I ² C	14
2.1.2. Características del Protocolo I ² C.....	14
2.1.3. Funcionamiento del Protocolo I ² C.....	15
2.1.4. Aplicaciones del Protocolo I ² C.....	17
2.2. Comunicación entre Arduino y LEGO EV3.....	17
2.3. Programación con LEGO EV3.....	18
2.4. Programación con RobotC.....	20
2.5. Programación con Arduino.....	21
2.5.1. Introducción a la programación con Arduino.....	21
2.5.2. Librería wire() de Arduino.....	22
2.5.3. Librería Servo() de Arduino.....	23
3. El proyecto.....	25
3.1. Interfaz de comunicación entre LEGO EV3 y Arduino.....	25
3.1.1. Opción 1: Comprar un adaptador.....	27
3.1.2. Opción 2: Cortar un cable de LEGO.....	28
3.1.3. Opción 3: Adaptar un cable RJ12.....	30
3.1.4. Fabricar el adaptador nosotros mismos.....	33
3.2. Programación mediante LEGO.....	35
3.2.1. Primeros pasos.....	35
3.2.2. EL bloque EV3_I2C.....	37
3.2.3. Construcción de bloques Arduino.....	39

3.2.4.	Programación de los bloques de Arduino con LEGO MINDSTORMS.	44
3.3.	Programación mediante RobotC.	51
3.3.1.	Primeros pasos.	51
3.3.2.	Programación de RobotC.....	53
3.4.	Otros lenguajes de programación.	59
3.5.	Programación de Arduino.....	60
3.5.1.	Consideraciones previas.....	60
3.5.2.	Código de Arduino.....	61
3.5.3.	Uso del serial en Arduino.	70
4.	Aplicaciones.....	71
4.1.	Aumentar las entradas y salidas de LEGO EV3 con sensores y actuadores de LEGO Mindstorms y Arduino.	71
4.1.1.	Sensor táctil NXT.....	71
4.1.2.	Sensor de sonido NXT.....	72
4.1.3.	Sensor de luminosidad NXT.....	73
4.1.4.	Sensor ultrasónico NXT.....	73
4.1.5.	Sensores y actuadores de LEGO EV3.	74
4.1.6.	Motores.	74
4.2.	Maqueta utilizando sensores y actuadores de bajo coste.	76
5.	Conclusión.	79
6.	Ampliaciones.	81
6.1.	Comunicación inalámbrica entre Arduino y LEGO.....	81
6.2.	Comunicación de sensores EV3 con Arduino.	81
7.	Referencias.....	83
8.	Bibliografía	84
9.	Anexos.....	88
9.1.	Anexo I.	88
9.2.	Anexo II.	96

Figura 1: Pirámide de la Automatización Industrial.	7
Figura 2: Set LEGO MINDSTORMS	8
Figura 3: Bloque RCX, NXT y EV3 respectivamente.	9
Figura 4: Sensores y actuadores.	9
Figura 5: Arduino UNO.....	12
Figura 6: Envío de datos Maestro-Esclavo.....	15
Figura 7: Esquema eléctrico para conectar a LEGO a Arduino, la resistencia de pull-up es de 82 K Ω	16
Figura 8: Caracterización de los bits en un mensaje I2C.	16
Figura 9: Envío de datos esclavo-Maestro.....	17
Figura 10: Bloques de acción.	18
Figura 11: Bloques de flujo.	19
Figura 12: Bloques de sensores.	19
Figura 13: Bloques de operaciones.....	19
Figura 14: Bloques avanzados.	20
Figura 15: Interfaz de usuario del software RobotC.....	20
Figura 16: Ejemplos de funciones en RobotC.....	21
Figura 17: Interfaz de usuario de RobotC.....	21
Figura 18: Estructura básica de programación Arduino.	22
Figura 19: Conector RJ12.	25
Figura 20: Adaptador LEGO-Arduino desarrollado por Dexter Industries.	27
Figura 21: Diseño circuito LEGO-Arduino.	28
Figura 22: Conector LEGO-Arduino 1.	29
Figura 23: Conector LEGO-Arduino 2.	29
Figura 24: Cable RJ12 estándar (izquierda) Cable RJ12 LEGO (derecha).....	30
Figura 25: Dirección de corte para obtener la pestaña.....	30
Figura 26: Lijamos la pestaña..	31
Figura 27: Comprobamos que encaja en el conector LEGO.....	31
Figura 28: Pegamos la pestaña con la posición corregida.....	32
Figura 29: Diferencia entre el cable RJ12 adaptado (izquierda) y el cable LEGO (derecha).	32
Figura 30: Diferencia entre conectores: conector LEGO (izquierda) conector RJ12 (derecha).....	33
Figura 31: Conector RJ12 con plástico retirado.....	33
Figura 32: Conexión entre el cable de LEGO y conector RJ12.....	34
Figura 33: placa de comunicación entre LEGO EV3 y Arduino.	34
Figura 34: Comunicación LEGO Arduino.....	34
Figura 35: Conexión entre Arduino y LEGO	35
Figura 36: Importar bloques en LEGO MINDSTORMS 1.	36
Figura 37: Importar bloques en LEGO MINDSTORMS 2.	36
Figura 38: Ilustración 29: Importar bloques en LEGO MINDSTORMS 3.	36
Figura 39: Bloque I2C desarrollado por Dexter Industries.	37
Figura 40: Bloque I2C.....	37
Figura 41: Bloque I2C Write 8 bytes.	38

Figura 42: Bloque I2C Read 1 byte.....	39
Figura 43: Bloque de tiempo.	39
Figura 44: Construcción de un bloque LED.....	40
Figura 45: Ruta para crear un bloque personalizado.	40
Figura 46:Constructor de mi bloque.....	40
Figura 47: Configuración del parámetro.	41
Figura 48:Imágenes para los bloques en LEGO MINDSTORMS	42
Figura 49: Icono programación.....	42
Figura 50: Paleta de bloques.	42
Figura 51: Bloque de programación LED.	43
Figura 52: Diseño del bloque LED 1.	43
Figura 53: Diseño del bloque LED 2	43
Figura 54: Bloque Arduino LED 1.	43
Figura 55: Bloque Arduino LED 2.	44
Figura 56: Programación bloque LED.	44
Figura 57: Ejemplo bloque LED.....	45
Figura 58: Programación bloque sensor.....	45
Figura 59: Bloque interruptor.....	46
Figura 60: Ejemplo bloque sensor 1.	46
Figura 61: Ejemplo bloque sensor 2.	47
Figura 62: Programación sensor de infrarrojo.....	48
Figura 63: Ejemplo programación bloque infrarrojo.....	48
Figura 64: Programación Servomotor	49
Figura 65: Ejemplo Servomotor.	49
Figura 66: Programación Motor CC.	50
Figura 67: Error a la hora de programar motores 1.	50
Figura 68: Error a la hora de programar motores 2.	50
Figura 69: Ejemplo de programación de motor 1.	51
Figura 70: Ejemplo de programación de motor 2.	51
Figura 71: Elección del bloque EV3 en RobotC.....	52
Figura 72: Actualización firmware RobotC	52
Figura 73: Instalación de la máquina virtual de RobotC.....	53
Figura 74: Protocolo comunicación I ² C RobotC.....	54
Figura 75: Matriz de instrucciones Arduino mediante programación LEGO.....	62
Figura 76: Ejemplo Comunicación serial.....	70
Figura 77: Esquema electrónico Sensor táctil.	71
Figura 79: Esquema electrónico Sensor de sonido.....	72
Figura 81:Esquema electrónico Sensor de luminosidad.	73
Figura 83: Esquema electrónico principal del Sensor ultrasónico.	74
Figura 84: Sistema de automatización basado en sensores y actuadores de bajo coste.	77
Figura 85: Conexión de un LED a Arduino.	88
Figura 86: Conexión sensor de temperatura a Arduino	89
Figura 87: Conexión servomotor a Arduino.	90

Figura 88:Puente en H.	90
Figura 89: L293D	91
Figura 90: Conexión del encapsulado L293D para motores inferiores a 5V	92
Figura 91: Conexión del encapsulado L293D para motores superiores a 5V.....	93

1. Introducción.

1.1. La Automatización.

Se entiende como Automatización Industrial como la aplicación de diferentes tecnologías para controlar y monitorear un proceso, o dispositivo capaz de regular y cumplir funciones repetitivas de forma automática, reduciendo al mínimo la intervención humana.

El objetivo de esta tecnología es generar la mayor cantidad de producto, en el menor tiempo posible, reduciendo costes y garantizando una uniformidad en la calidad de dicho producto. Además, permite la reducción de costo en operadores humanos, o el remplazo de estos, en tareas repetitivas o de alto riesgo.

Todo esto es posible gracias a los Sistemas de Control, que son organizaciones de equipos e instrumentos, que combinados con procedimientos algorítmicos trabajan en un entorno con propósitos previamente establecidos. Es decir, se encargan de la observación del proceso, controlar las variables a automatizar, el procesamiento de la información y la corrección de los elementos terminales para conseguir lo deseado.

La Automatización Industrial comprende un conjunto de funcionamiento y conexión de varias tecnologías como la de instrumentación, neumática, electrotecnia, robótica, telemetría etc. La integración y desarrollo de estas tecnologías está haciendo posible una evolución notoria, y queda representada en la llamada "Pirámide de automatización", que recoge los cinco niveles tecnológicos que se pueden encontrar en un entorno industrial:

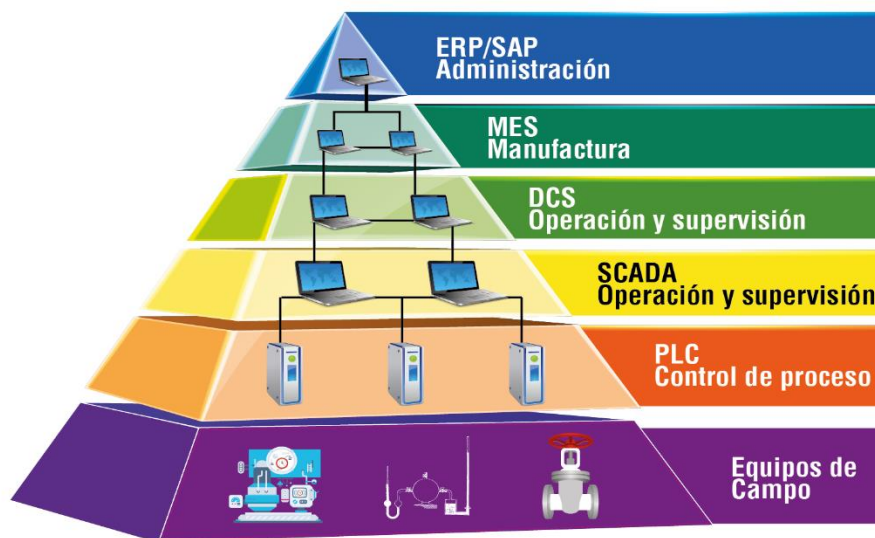


Figura 1: Pirámide de la Automatización Industrial.

- Primer nivel o "nivel de campo" incluye los dispositivos físicos presnetes en la industria, como los actuadores y sensores.
- El segundo nivel o "nivel de control" incluye los dispositivos controladores como ordenadores, PLCs, PIDs, etc.

- El "nivel de supervisión" (tercer nivel) corresponde a los sistemas de supervisión, control y adquisición de datos (SCADA).
- En un nivel superior o "nivel de planificación" se encuentran los sistemas de ejecución de la producción (MES).
- La cúspide de la pirámide ("nivel de gestión") la componen los sistemas de gestión integral de la empresa (ERP).

Su desarrollo es tal, que prácticamente todos los objetos que nos rodean en nuestra vida cotidiana, están contruidos total o parcialmente mediante la ayuda de la Automatización. Por este hecho, es interesante hacer llegar la Automatización a todo el mundo de forma sencilla y comprensible. Para ello utilizaremos, LEGO MINDSTORMS que permite construir, programar y controlar de modo inteligente un sistema automático, siendo un producto que está al alcance de todos, bastante económico y fácil de programar.

1.2. LEGO MINDSTORMS

LEGO Mindstorms es una plataforma de juguetes de robótica, fabricado por la empresa LEGO. Cada vez es más común utilizar esta plataforma de forma educativa, tal es así, que las ventas de LEGO Mindstorms se han disparado los últimos años en centros educativos. El origen de este aumento es debido a que LEGO Mindstorms permite la construcción de proyectos que aglutinan una gran variedad de conocimientos relacionados con la ingeniería, programación, matemáticas y física. Por esta razón son cada vez más los centros educativos que en la Enseñanza Secundaria Obligatoria (E.S.O.) y Bachillerato utilizan esta plataforma ya que, además, los estudiantes pueden experimentar con la tecnología de una manera creativa e interesante, fomentando el trabajo en equipo, toma de decisiones, la resolución de problemas o la motivación por resultados. También constituye una herramienta ideal para despertar futuras vocaciones relacionadas con el sector tecnológico.



Figura 2: Set LEGO Mindstorms.

Hasta 2016 ha habido 3 generaciones de LEGO Mindstorms: El bloque RCX, el bloque NXT y el EV3, este último el protagonista de este proyecto.



Figura 3: Bloque RCX, NXT y EV3 respectivamente.

LEGO Mindstorms se compone por un dispositivo llamado EV3 *brick* (en español, bloque EV3). Este dispositivo realiza la gestión de las entradas y salidas, por lo que es el encargado del control del sistema, en él se realiza la programación y actúa como fuente de potencia. LEGO Mindstorms también incluye una serie de sensores, desarrollados y diseñados exclusivamente para el bloque EV3.



Figura 4: Sensores y actuadores.

1.2.1. Características físicas del LEGO EV3.

Las características de hardware de los bloques de LEGO han ido evolucionando con cada modelo. A modo resumen podemos ver una comparación en la siguiente tabla, aunque posteriormente nos enfocaremos en LEGO EV3.

	RCX	NXT	EV3
Fecha de lanzamiento	1998	2006	2013

Pantalla	Segmento LCD monocromo	100x64 píxeles LCD monocromo	178x128 píxeles LCD monocromo
Procesador	Hitachi H8/300 @ 16 KB ROM	Atmel AT91SAM7S256 @ 48 MHz	TI Sitara AM1808 @ 300 MHz
Memoria principal	32 KB RAM 16 KB ROM	64 KB RAM 256 KB Flash	64 MB RAM 16MB Flash
Memoria extensible	No	No	Ranura micrsoSD
Puerto USB	No	No	Sí
WIFI	No	No	Opcional mediante un módulo Wifi externo
Bluetooth	No	Sí	Sí
Compatible con Apple	Sí	No	Sí

Tabla 1: Comparativa hardware entre RCX, NXT y EV3

En lo que se refiere al modelo EV3, presenta las siguientes características:

1.2.1.1. Características físicas del bloque EV3.

- 4 puertos de entrada rj12 (de 6 hilos) modificada, para conectar los sensores al bloque EV3.
- 4 puertos de salida rj12 modificada para conectar los motores al bloque EV3.
- Un puerto mini USB para PC, para conectar el bloque EV3 a un ordenador.
- Un puerto de host USB, para agregar un conector Wi-Fi y establecer conexiones en cadena.
- Un puerto para tarjetas Micro SD, para ampliar la memoria.
- Un altavoz integrado.
- Receptor de señales infrarrojas para recibir comandos.
- Receptor Bluetooth y wifi.

1.2.1.2. Características de hardware.

- El procesador principal es del tipo ARM de Atmel, modelo AT91SAM7S256. Este procesador es de 32 bits, con una memoria Flash de 256Kb, una memoria RAM de 64Kb y funciona con una frecuencia de 48MHz.
- Procesador secundario, es un coprocesador AVR de Atmel, modelo ATmega48. Este procesador tiene una memoria Flash de 4Kb, una memoria RAM de 512Kb y funciona con una frecuencia de 8MHz.
- Conexión con bluetooth mediante un módulo CSR BlueCoreTM 4 v2.0 con sistema EDR. Soporta Perfiles por Puerto Serie (SPP), presenta 47 Kb de memoria RAM interna, contiene 8 Mb de memoria FLASH externos y por último trabaja a 26 MHz.

- El bloque EV3 se comunica con el PC mediante USB 2.0, con una velocidad de 12 Mbits/s. Todos los puertos de salida soportan el protocolo I2C. Existen 3 puertos de salida con interfaz de 6 hilos y soporte para lectura de codificadores.

1.2.1.3. Otras características.

- En el centro del robot se encuentra un display gráfico LCD de 100x64.
- El altavoz de salida del robot tiene una resolución de 8 bits, y soporta tasas de muestreo de 2 a 16 KHz.
- El usuario puede interactuar con el robot mediante 4 botones.
- La alimentación del robot se puede hacer mediante 6 pilas alcalinas de tipo AA, bien mediante una batería de Ion Litio de 1400 mA

1.3. RobotC.



RobotC Fue desarrollado por la *Robotics Academy de la Carnegie Mellon University de Pittsbur* (Pensilvania, EEUU), y comprende un lenguaje de programación basado en el lenguaje C especializado en el campo de la educación.

Fue creado con un entorno de desarrollo para un solo uso "ETU" (Fácil de Usar). De este modo, es compatible con los dispositivos VEX IQ, VEX Corex NXT, EV3 y también con Arduino Mega 1280 y 2560.

RobotC presenta una solución multiplataforma que permite a los estudiantes aprender la programación basada en C usada en educación avanzada y aplicaciones profesionales.

Las ventajas que tiene un estudiante a la hora de operar con RobotC son:

- Usa el estándar industrial del lenguaje de programación en C.
- Moderna GUI (Interfaz gráfica de Usuario) de Windows con un interfaz visual estándar.
- Las herramientas adicionales de depuración permiten al usuario ver en tiempo real los estados del motor y los sensores.
- Más de 100 ejemplos de programas con una documentación extensa para estudiantes y aficionados que pueden empezar a aprender como programar.
- Avanzado editor de código fuente con identificación inteligente, competencia automática de código y una interfaz con pestañas que permite abrir múltiples programas.
- Video tutoriales, foros comunitarios, y planes de estudios dados por la Academia de Robótica de Carnegie Mellon.
- Habilidades de RobotC de fácil transición para usar herramientas de ingenieros profesionales.

1.4. Arduino.

Arduino es una plataforma hardware de código abierto (*open-source*) caracterizado por la facilidad hardware y software del producto. El hardware se basa en una placa con unas entradas y salidas con un entorno de programación del tipo *Processing/Wiring*. Arduino puede ser utilizado para un sinfín de aplicaciones, como desarrollar objetos interactivos y autónomos. Las placas se pueden montar a mano o adquirirse, esto hace que todas las placas y módulos de Arduino sean de muy bajo coste. Todo el entorno de desarrollo para Arduino es libre y puede descargarse gratuitamente en la web del fabricante.

1.4.1. Características físicas de Arduino.

Hay una gran variedad de placas de Arduino, nosotros utilizaremos, ARDUINO UNO R3 (*smd edition*), el aspecto que presenta es el siguiente:



Figura 5: Arduino UNO.

Las principales características de esta placa son:

Microntolador	ATmega328
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
Digital E/S Pins	14 (PIN 6 es la señal de salida PWM)
PIN de entrada analógicos	6
Intensidad DC por PIN E/S	40 mA
Intensidad DC por PIN 3.3V	50 mA
Memoria Flash	32 KB (ATmega328) de la cual 0.5 KB es usado para el bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad de reloj	16 MHz

Tabla 2: Características hardware Arduino UNO

1.5. Objetivos del proyecto.

El objetivo principal del presente proyecto es abrir el sistema de LEGO Mindstorms para conseguir que este dispositivo no solo pueda operar con sensores o actuadores distribuidos por LEGO, sino que pueda operar con cualquier otro tipo de sensor o actuador sin importar su marca, funcionamiento, tipo... utilizando una interfaz de bajo coste como puede ser una placa de Arduino.

También perseguiremos una serie de objetivos secundarios:

1. Aumentar las entradas y salidas de LEGO EV3.
2. Diseñar un sensor y un actuador electrónico de bajo coste y que LEGO EV3 sea capaz de comunicarse con él.
3. Hacer un ejemplo sencillo de un sistema de automatización utilizando LEGO EV3 como si de un PLC se tratase.
4. Realizar todo lo anterior mediante un procedimiento sencillo y metódico.

1.5.1. El proyecto.

Una vez que tenemos un conocimiento básico de LEGO Mindstorms y Arduino, estamos preparados para abordar el presente proyecto.

En líneas generales este proyecto consistirá en comunicar LEGO EV3 con Arduino, estudiar su comunicación, lograr una comunicación fluida y de fácil programación, la construcción de sensores y también realizaremos algunos ejemplos para probar su funcionamiento.

Y todo ello, de forma sencilla para que cualquier persona independientemente del nivel de formación que tenga, pueda lograr esto, con unos sencillos pasos.

Por último, para probar el presente proyecto constituiremos una serie de aplicaciones a todo lo visto, así como la construcción de un caso práctico sencillo de un sistema automatizado.

2. Estado del Arte.

2.1. Protocolo de comunicación I²C.

Este protocolo será la base de este proyecto, por lo que es de suma importancia tener unos conocimientos básicos. Realizaremos la comunicación con este protocolo ya que según la hoja del fabricante de LEGO EV3, los puertos de entrada y salida admiten este protocolo y al ser un protocolo estándar, cualquier dispositivo puede comunicarse utilizando dicho protocolo sin importar sus características físicas o software.

2.1.1. El Protocolo I²C

Un circuito inter-integrado (I²C, del inglés *Inter-Integrated Circuit*) es un bus serie de datos desarrollado en 1982 por Philips *Semiconductors* (hoy NXP *Semiconductors*). Es un estándar que facilita la comunicación entre dispositivos con cierto nivel de "inteligencia" como microcontroladores y memorias. Permite la comunicación entre dispositivos a una velocidad de entorno 100Kbits/s.

En la actualidad, muchos equipos electrónicos con circuitos integrados se comunican mediante el protocolo de comunicación I²C, como, por ejemplo, procesadores de señal, televisiones reproductoras de CD/DVD, etc.

2.1.2. Características del Protocolo I²C.

Las características de dicho protocolo son:

- Presenta una metodología de comunicación de datos en serie y síncrona.
- Presenta dos señales, más una línea de masa:
 - Señal de datos, SDA (*System Data*) es la línea de los pulsos de reloj que sincronizan el sistema.
 - Señal de reloj, SCL (*System Clock*) es la línea por la que se mueven los datos entre los dispositivos.
 - Masa, GND (*Ground*) común de la interconexión entre todos los dispositivos conectados al bus.
- Cada dispositivo conectado al bus tiene un código de dirección seleccionable mediante software. Se establece de esta manera, una relación Maestro/esclavo entre el micro y los dispositivos conectados.
- El bus permite la conexión de varios Maestros, ya que incluye un detector de colisiones.
- El protocolo de transferencia de datos y direcciones posibilita diseñar sistemas completamente definidos por software.
- Los datos y direcciones se transmiten con palabras de 8 bits.

2.1.3. Funcionamiento del Protocolo I²C.

Con el Protocolo I²C podemos conectar a un mismo bus hasta 128 dispositivos. Cada dispositivo posee una dirección única de 7 bits (por eso podemos tener hasta 128 dispositivos, ya que $2^7=128$) para poder establecer la comunicación mediante el bus. Un dispositivo que está conectado al bus puede operar tanto en modo esclavo como en modo maestro:

- Maestro: es siempre quién maneja la línea de reloj SCL, y es quién solicita información.
- Esclavo: los esclavos responden al maestro y no pueden mandar información sin la autorización del maestro.

En el mismo instante, solo puede haber un dispositivo en modo maestro, pero puede ser que haya varios dispositivos en modo maestro (multimaestro), pero esta situación no es usual. En este caso los distintos dispositivos van turnándose para actuar como maestro ya que, en un instante de tiempo determinado, solo un dispositivo puede actuar como maestro. Como he comentado anteriormente, el dispositivo en modo maestro es el único capaz de iniciar el envío de datos, y es el encargado de generar la señal de reloj.

El modo de funcionamiento del bus I²C es el siguiente:

Bajo condiciones iniciales (el bus en reposo), tanto la señal de datos (SDA) y de reloj (SCL) están en estado lógico alto. En estas condiciones, cuando un dispositivo en modo maestro cambia su estado lógico de señal de datos (SDA) a bajo, accede al bus estableciendo la secuencia de inicio.

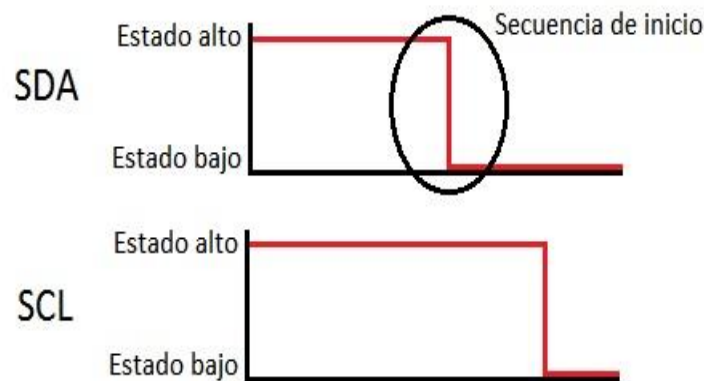


Figura 6: Envío de datos Maestro-Esclavo.

De este modo, el bus funciona como un circuito AND cableado lo que se logra utilizando una resistencia de *pull-up* a VDD y obliga a que todos los dispositivos conectados a dicho bus sean de drenador o colector abierto

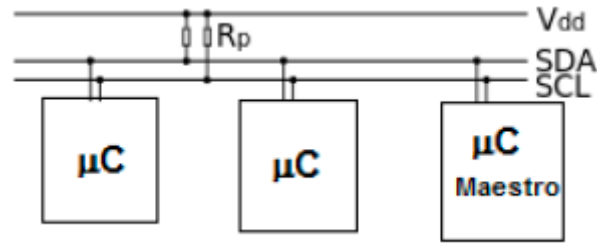


Figura 7: Esquema eléctrico para conectar a LEGO a Arduino, la resistencia de pull-up es de 82 KΩ.

En la secuencia de inicio, identificamos al dispositivo con el que queremos trabajar y la operación que queremos realizar. Para ello el primer byte, que se transmite en la condición de inicio contiene 7 bits que identifican al dispositivo (dirección) y el octavo bit (R/W) indica la operación que se va a realizar en dicho dispositivo:

- a. Escritura: nivel lógico bajo
- b. Lectura: nivel lógico alto

Una vez enviada la dirección del dispositivo con el que nos queremos comunicar, si dicho dispositivo se encuentra en el bus, responde con un bit en estado lógico bajo. Este bit se reconoce como (ACK) e indica al dispositivo en modo maestro que el dispositivo en modo esclavo reconoce la solicitud y está en condiciones de comunicarse. Una vez esto comienza el intercambio de datos entre ambos dispositivos maestro y esclavo.

Por lo tanto, tenemos 7 bits de direccionamiento, más el bit R/W, más el bit ACK, como se puede apreciar en la siguiente imagen:

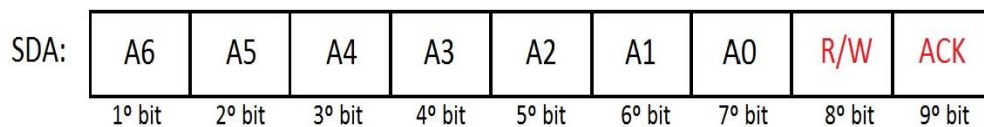


Figura 8: Caracterización de los bits en un mensaje I2C.

Si el bit R/W está en un nivel lógico bajo (escritura), el dispositivo maestro envía datos al dispositivo esclavo mientras continúe recibiendo señales de reconocimiento. El contacto concluye cuando se hayan transmitido todos los datos.

En el caso contrario, cuando el bit R/W está a un nivel lógico alto (lectura), el dispositivo maestro genera pulsos de reloj para que el dispositivo esclavo pueda enviar datos. Por tanto, cada byte recibido el dispositivo maestro genera un pulso de reconocimiento.

Si el dispositivo maestro quiere dejar libre el bus, debe generar una condición de parada, para ello la señal SDA pasa a estar, como en condiciones iniciales, a nivel lógico alto.

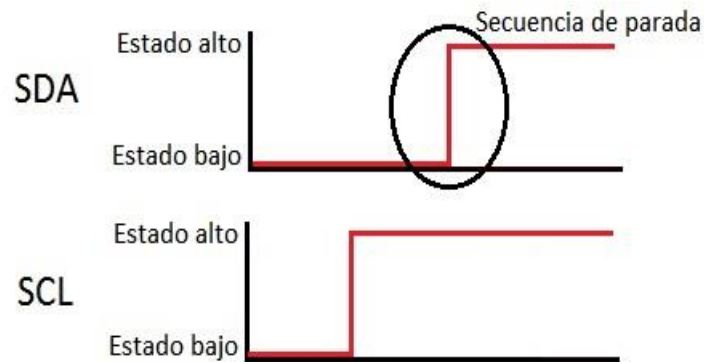


Figura 9: Envío de datos esclavo-Maestro.

Si se desea seguir transmitiendo, el dispositivo maestro puede generar otra condición de inicio. Esta nueva condición se llama "inicio reiterado" y se puede emplear para direccionar un dispositivo esclavo diferente o para alterar el estado del bit de lectura/escritura.

2.1.4. Aplicaciones del Protocolo I²C.

Son numerosas las aplicaciones en la actualidad que utilizan este protocolo, algunas de ellas pueden ser:

- Este tipo de protocolo se suele aplicar para la transmisión de datos de control y configuración como, por ejemplo:
 - Conversores de señal AD o DA con baja tasa de frecuencia de muestreo.
 - En pequeños espacios de memoria o conmutadores bidireccionales.
 - Multiplexores.
- También es utilizado como base para ACCESS.bus y monitores de interfaz VESA.
- El SMBus (del fabricante Intel) para la comunicación de componentes de la placa base, se parece mucho al bus I²C. La mayoría de los circuitos integrados soportan ambos buses.
- El protocolo I²C tuvo gran importancia en el pasado en el área de las tarjetas chip. Por ejemplo, la tarjeta sanitaria alemana hasta el 2014 era una tarjeta que utilizaba el protocolo I²C.
- Gran cantidad de instrumentación de control y automatización se rige con este protocolo.

2.2. Comunicación entre Arduino y LEGO EV3.

Para realizar el presente proyecto he elegido Arduino para la comunicación con LEGO EV3 y la instrumentación. No obstante, podríamos haber elegido otro dispositivo como por ejemplo Raspberry PI.

El motivo de elegir Arduino ha sido el gran desarrollo y la gran cantidad de información presente en internet y su bajo coste. Además, presenta una característica muy importante es que Arduino presenta una librería estándar que utiliza los pines

analógicos para las funciones SDA y SCL. Estos pines en el Arduino UNO son los pines analógicos A4 (SDA) y A5 (SCL).

La librería I²C, que presenta Arduino para gestionar este protocolo es WIRE(), hablaremos de esta librería más adelante.

En lo que respecta a LEGO EV3, según la hoja del fabricante todos los puertos son compatibles con el protocolo de comunicación I²C. Sin embargo, nuestra comunicación con Arduino siempre la debemos de realizar por el puerto 1 de LEGO ya que los otros puertos 2, 3 y 4 son más susceptibles a fallos. Algunos de estos fallos son: no establecer comunicación entre Arduino y LEGO y ocasionalmente LEGO o Arduino no recibe o envía información. No he sido capaz de encontrar una explicación a esto, por lo que el fallo, será a nivel de hardware del EV3.

2.3. Programación con LEGO EV3.

LEGO Mindstorms, se programa mediante un software que viene con el producto, esta programación está desarrollada por National Instruments LabView, su principal característica reside en que, el entorno de este software de programación es visual, el cual emula la construcción por bloques que permite hacer programas de forma relativamente rápida, sin tener muchos conocimientos de programación.

Principalmente en el software podemos distinguir 5 tipos de bloques:

- **Bloques de acción.** Son de color verde y controlan las acciones del programa como rotaciones de los motores, sonidos, luces etc.

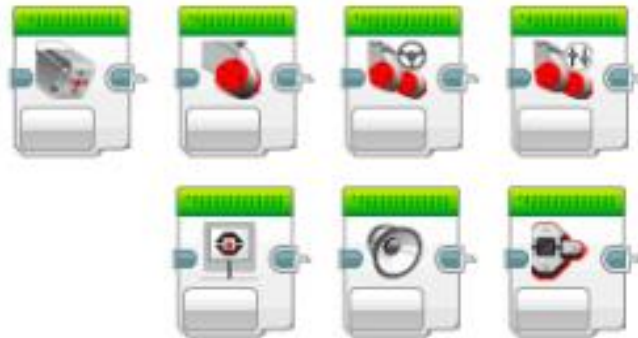


Figura 10: Bloques de acción.

- **Bloques de flujo.** Son de color naranja, controlan la transición del programa, estos permiten crear ciclos de programación como for y while.

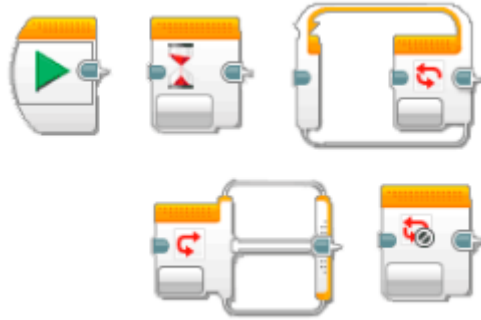


Figura 11: Bloques de flujo.

- **Bloques de sensores.** Son de color amarillo, leen datos de los sensores.



Figura 12: Bloques de sensores.

- **Bloques de operaciones de datos.** Son de color rojo y permiten leer y escribir variables, comparar valores etc.



Figura 13: Bloques de operaciones.

- **Bloques avanzados.** Estos bloques permiten, entre otras cosas administrar archivos o establecer conexiones Bluetooth.

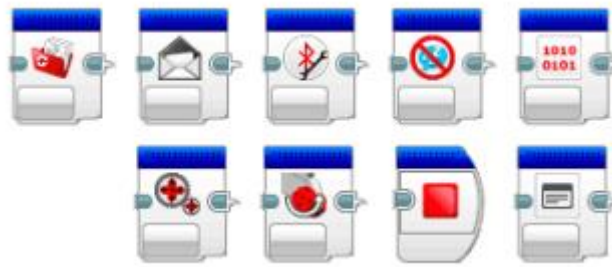


Figura 14: Bloques avanzados.

Este lenguaje permite las instrucciones secuenciales, instrucciones de ciclos e instrucciones de decisiones, éstas últimas, basadas en los datos reportados por los sensores que se pueden añadir al robot.

2.4. Programación con RobotC.

Como hemos comentado anteriormente, RobotC se basa en uso del lenguaje C. Este tipo de programación puede llegar a presentar una mayor dificultad para personas que no hayan tenido una formación previa, sin embargo, RobotC presenta una gran ventaja para proyectos que tengan cierta envergadura, ya que con RobotC podemos reducir un gran proyecto utilizando LEGO Mindstorms a unas pocas líneas de código.

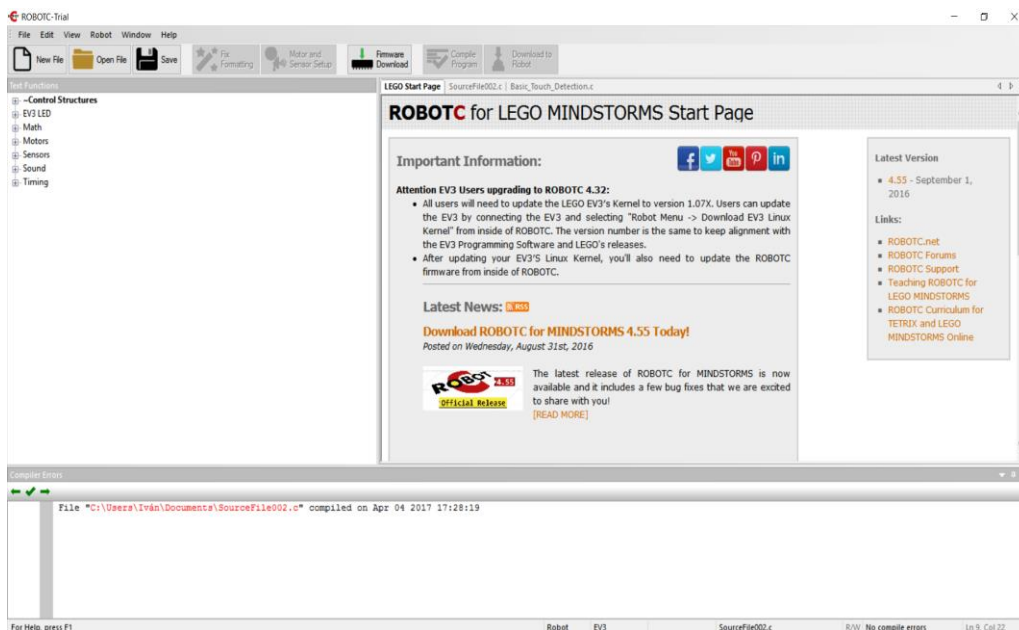


Figura 15: Interfaz de usuario del software RobotC.

Para facilitar la utilización del lenguaje C, RobotC presenta una recopilación de todas las funciones necesarias para poder operar con todos sensores, actuadores y con el propio LEGO EV3 como por ejemplo su pantalla. También presenta una serie de ejemplos en el mismo programa, también dispone de una página Web, en cuatro idiomas (ruso, chino, inglés y español), con videotutoriales, guías, proyectos y ejemplos.

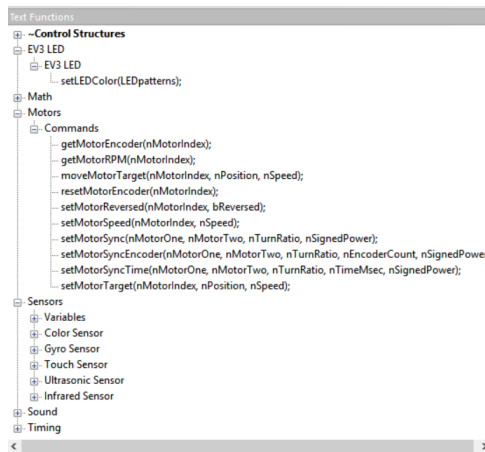


Figura 16: Ejemplos de funciones en RobotC.

Además, al igual que LEGO Mindstorms, presenta una interfaz sencilla con dos botones principales para, compilar y ejecutar el código en el bloque EV3.



Figura 17: Interfaz de usuario de RobotC.

2.5. Programación con Arduino.

2.5.1. Introducción a la programación con Arduino.

Arduino proporciona un entorno de programación sencillo y potente para programar, pero además incluye las herramientas necesarias para compilar el programa y subirlo al microcontrolador. El IDE de Arduino, nos ofrece un sistema de gestión de librerías y placas muy práctico. El IDE de Arduino es un software sencillo que carece de funciones avanzadas típicas de otros IDEs, pero suficiente para programar.

Un programa de Arduino se denomina sketch o proyecto y tiene la *extensión.ino*. La estructura básica de un sketch de Arduino es bastante simple y se compone de al menos dos partes. Estas dos partes son obligatorias y encierran bloques que contienen declaraciones o instrucciones.

Adicionalmente se puede incluir una introducción con los comentarios que describen el programa y la declaración de las variables y llamadas a librerías.

Estas partes fundamentales de Arduino son:

- **Setup()** es la parte encargada de recoger la configuración.
- **Loop()** es la que contiene el programa que se ejecuta cíclicamente.

Ambas funciones son necesarias para que el programa trabaje. De esta forma, la estructura básica de cualquier programa realizado en Arduino debe presentar la siguiente estructura.

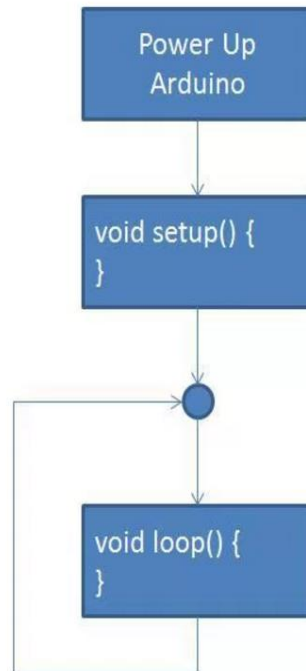


Figura 18: Estructura básica de programación Arduino.

2.5.2. Librería wire() de Arduino.

La librería Wire() es la que nos va a permitir gestionar la comunicación I²C. Como es una librería que no está muy bien explicada y su descripción suele estar en inglés, he decidido explicar las funciones más importantes de esta librería. Antes de empezar tenemos que tener presente una serie de parámetros:

- Dirección: es la dirección de 7 bits del dispositivo al que le queremos pedir los datos y el parámetro.
- Cantidad: número de bytes a pedir.
- Valor: byte que se desee enviar.
- String: cadena de tipo (char *) que se desee enviar.
- Dato: vector de datos para enviar (byte *).
- Cantidad: el número de bytes que se desea transmitir.
- Handler función que será llamada cuando el esclavo recibe datos. Esta función debería coger un único parámetro entero (el número de bytes recibidos desde un maestro) y no devolver nada.

Aunque no vamos a utilizar todas las funciones las más importantes son:

- **Wire.begin() y Wire.begin(dirección).** Esta función inicializa la librería Wire y conecta Arduino al bus. Si no se especifica la dirección, Arduino se conectará al bus como maestro, mientras que si ésta se indica lo hará como esclavo y asumiendo la dirección que se la ha proporcionado. En ambos casos, esta función no devuelve ningún valor.

- **Wire.requestFrom(*dirección*, *cantidad*)**. Solicita bytes desde otros dispositivos. Los bytes pueden ser recibidos con las funciones `available()` (que nos indica si hay datos disponibles para ser leídos en el bus) y `receive()` (que realiza la lectura de los datos). Esta función no devuelve nada.ç
- **Wire.beginTransmission(*dirección*)**. Comienza una transmisión a un dispositivo I²C esclavo con la dirección que se especifique en el parámetro “dirección”. Posteriormente, prepara los bytes a transmitir con la función `send()` y los transmite llamando a la función `endTransmission()`. Esta función no devuelve nada.
- **Wire.endTransmission()**. Finaliza una transmisión a un esclavo que fue empezada por `beginTransmission()`, y realmente lo que hace es transmitir los bytes que fueron preparados por `send()`. Esta función tampoco devuelve nada.
- **Wire.Send(*valor*), Wire.Send(*string*) o Wire.Send(*dato*, *cantidad*)**. Envía datos desde un esclavo como respuesta a una petición de un maestro, o prepara los bytes para transmitir desde un maestro a un esclavo (entre llamadas a `beginTransmission()` y `endTransmission()`).
- **Wire.available()**. Devuelve el número de bytes disponibles para recuperar con `receive()`. Debería ser llamada por un maestro después de llamar a `requestFrom()` o por un esclavo dentro de la función a ejecutar por `onReceive()`, que será lo que nosotros hagamos al recibir datos del NXT. Esta función devuelve el número de bytes disponibles para leer.
- **Wire.receive()**. Recupera un byte que fue transmitido desde un dispositivo esclavo a un maestro después de una llamada a `requestFrom()` o que fue transmitido desde un maestro a un esclavo. Devuelve el byte recibido.
- **Wire.onReceive(*handler*)**. Registra una función que será llamada cuando un dispositivo esclavo reciba una transmisión desde un maestro. Esta función no devuelve nada.
- **Wire.onRequest(*handler*)**. Registra una función que será llamada por el dispositivo esclavo cuando un maestro solicite datos. No devuelve nada.

2.5.3. Librería Servo() de Arduino.

Esta biblioteca permite usar Arduino para controlar servomotores RC. Los servos tienen engranajes integrados y un eje que puede ser controlado con precisión. Los servos estándar permiten que el eje se posicione entre 0 y 180 grados. La librería de Arduino permite hasta 12 motores.

Lo primero que tenemos que tener en cuenta a la hora de usar esta librería es llamar a dicha librería y crear una instancia tipo servo:

```
#include <Servo.h>
Servo servo1;
```

Después tenemos dos funciones principales que son las que vamos a utilizar:

- `servo1.attach(pin)`: esta función es para indicar a Arduino en que pin está conectado nuestro servo.
- `Servo1.write(posición)`: esta función sirve para indicar la posición que queremos mover el servo. La posición está en ángulos.

A continuación, podemos ver un pequeño ejemplo:

```
#include <Servo.h> // Incluir la librería Servo
Servo servo1; // Crear un objeto tipo Servo llamado servo1
int angulo = 0;

void setup()
{
  servo1.attach(9); // Conectar servo1 al pin 9
}

void loop()
{
  for(angulo = 0; angulo <= 180; angulo += 1) //incrementa angulo 1 grado
  {
    servo1.write(angulo);
    delay(25);
  }
  for(angulo = 180; angulo >=0; angulo -=1) //decrementa angulo 1 grado
  {
    servo1.write( angulo );
    delay(25);
  }
}
```

3. El proyecto.

3.1. Interfaz de comunicación entre LEGO EV3 y Arduino.

Empezaremos este proyecto analizando el conector que permite a LEGO Mindstorms comunicarse con las entradas (sensores) y salidas (motores, altavoces, etc.).

En la hoja de fabricante de LEGO, nos indica que su comunicación se realiza con conectores del tipo RJ12. El conector RJ12 se utiliza con bastante frecuencia en comunicaciones telefónicas, dispone de 6 posiciones y 6 contactos (6P6C). Estos conectores se definen por el número de posiciones "P6" y el número de contactos instalados dentro de estas posiciones "6PxC" (en nuestro caso: 6P6C). Dentro de esta familia de conectores podemos encontrar RJ10 (4P2C), RJ11 (6P4C), etc.

Del mismo modo, también podemos leer que este conector es un conector RJ12 modificado (la pestaña no la tienen en el centro si no, en un lateral), esto es debido a que este tipo de cable, puede ser conectado a una línea de teléfono fija que puede llegar a tener una tensión de 48 Vcc cuando el aparato está colgado y de hasta 104 Vca cuando está sonando el timbre de llamada. Para evitar esto, LEGO decidió modificar la posición de la pestaña a un lateral e impedir de esta forma que un niño pueda conectar el cable a una entrada telefónica.

El conector RJ12 presenta la siguiente forma:

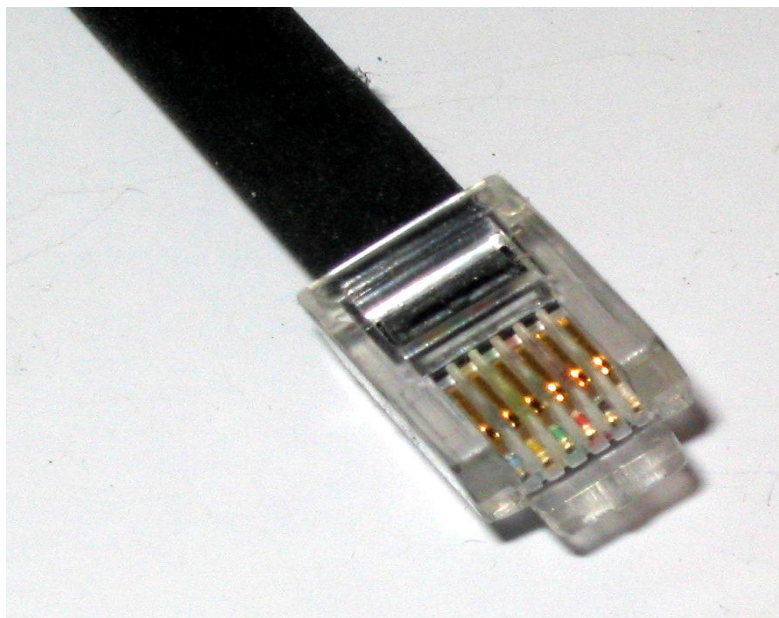


Figura 19: Conector RJ12.

Podemos apreciar los 6 colores de sus 6 pins, cuya función la podemos resumir en la siguiente tabla:







PIN	NOMBRE	COLOR	FUNCIÓN
1	ANA		Interfaz analógico. Voltaje de 9V
2	GND		Tierra
3	GND		Tierra
4	IPOWERA		Voltaje de 4,7V
5	DIGIAI0		Señal de reloj del protocolo I ² C (SCL)
6	DIGIAI1		Señal de datos del protocolo I ² C (SDA)

Tabla 3: Caracterización del cable RJ12.

Como hemos comentado anteriormente, Arduino soporta de fábrica la comunicación I²C mediante una librería estándar llamada `wire()`, que utiliza dos de los pines analógicos para las funciones SDA (Datos) y SCL (Clock).

- En el Arduino UNO, los pines I²C están en los pines analógicos A4 (SDA) y A5 (SCL).
- En el Arduino Mega y DUE, son el 20 (SDA) y en el 21(SCL).

Como nosotros vamos a utilizar Arduino UNO debemos tener en cuenta que los pines analógicos encargados de controlar el protocolo I²C son A4 (SDA) y A5 (SCL).

En esta parte del proyecto debemos comunicar LEGO EV3 y Arduino. Para ello, si hacemos una recopilación de lo que sabemos hasta el momento, sabemos que LEGO EV3 admite el protocolo I²C y que el cable de comunicación es de tipo RJ12 y dicho cable se compone por 6 cables explicados en la tabla anterior. Debemos de fijarnos en el cable amarillo y azul que son los que se encargan de operar con el protocolo I²C.

Por tanto, para comunicar ambos dispositivos debemos conectar el cable amarillo a la entrada analógica A5 del Arduino y el cable azul a la entrada analógica A4 del Arduino. Y en el caso de querer que el Arduino obtenga la alimentación de LEGO, también debemos conectar el cable rojo a la tierra y verde a la entrada Vin del Arduino.

Hay varios métodos para realizar esta comunicación, descritos a continuación:

3.1.1. Opción 1: Comprar un adaptador.

En internet podemos conseguir este adaptador, buscando con el nombre: *Breadboard Adapter for LEGO MINDSTORMS*.

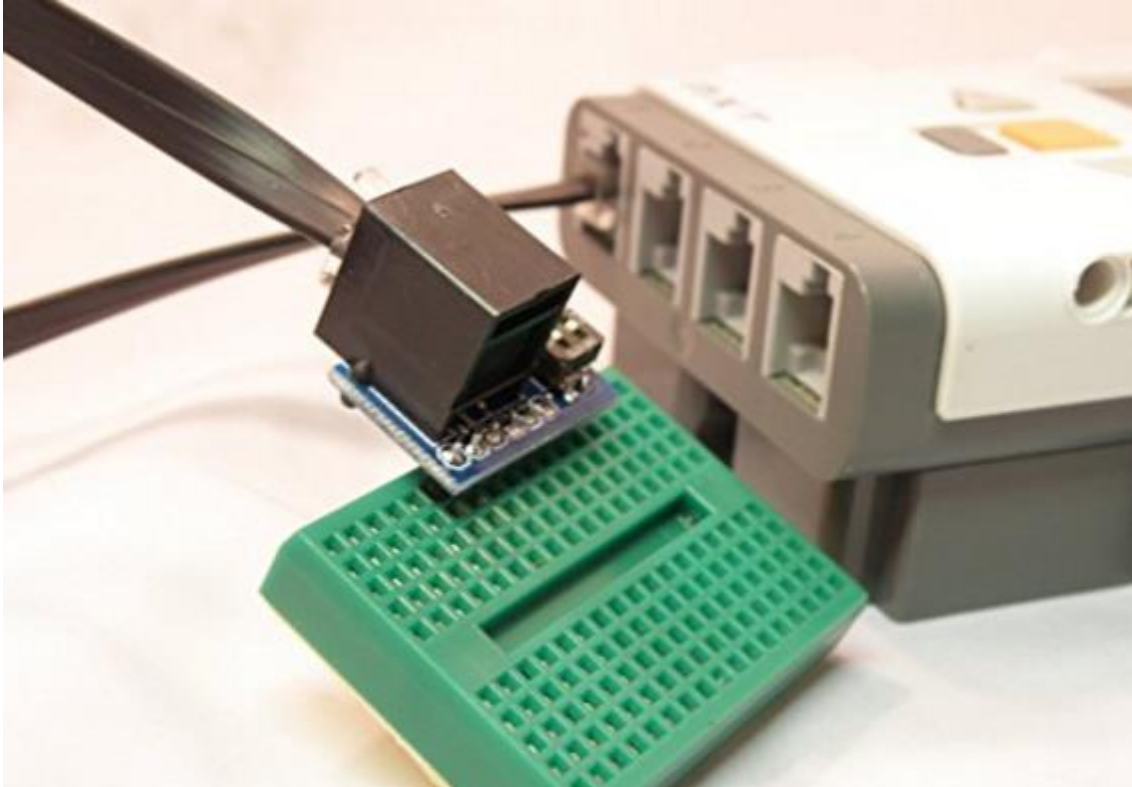


Figura 20: Adaptador LEGO-Arduino desarrollado por Dexter Industries.

La dirección específica para acceder a este conector la podemos encontrar en las referencias del presente proyecto ((Industries, Conector LEGO Arduino, 2017)).

Sin embargo, presenta algunos inconvenientes:

- El primero es su precio, el adaptador vale unos 15€ que sumados a 20€ por gastos de envío, hace un total de 35€, ya que la empresa se encuentra en Estados Unidos.
- El tiempo de llegada también presenta otro inconveniente, puede tardar de una semana a dos semanas.
- Por último, el diseño del adaptador hace que sea necesario un *breadboard* para conectarlo a Arduino lo que hace que sea incómodo.

3.1.2. Opción 2: Cortar un cable de LEGO.

La opción más fácil sería cortar por una esquina el cable de comunicación de Arduino, pelar el cable e ir separando los cables.

De los cables blanco y negro podemos prescindir de momento, los importantes son el cable verde, rojo, amarillo, y azul.

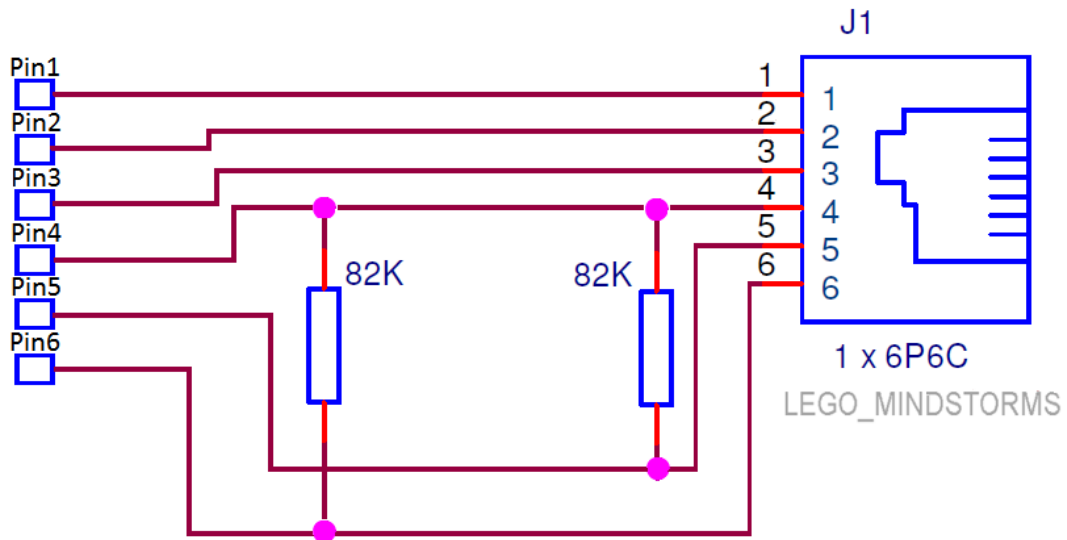


Figura 21: Diseño circuito LEGO-Arduino.

Como podemos observar, tenemos dos resistencias de 82K estas resistencias están configuradas en modo *pull-up* y son necesarias para la comunicación I²C entre LEGO y Arduino.

Como comentemos anteriormente estas resistencias de *pull-up* permiten compartir el bus de comunicación al comportarse como una compuerta AND-cableada que evita cortocircuitar el bus sin estas resistencias la comunicación no sería posible.

Para realizar los primeros experimentos de comunicación entre LEGO y Arduino y prescindir de la *breadboard* y realicé el circuito en una regleta de pines, obteniendo el siguiente resultado:



Figura 22: Conector LEGO-Arduino 1.



Figura 23: Conector LEGO-Arduino 2.

Una vez esto, podemos conectar este conector Arduino, teniendo en cuenta que:

- Cable rojo: Tierra.
- Cable verde: Vin Arduino.

- Cable amarillo: pin analógico A5 del Arduino.
- Cable verde: pin analógico A4 del Arduino.

3.1.3. Opción 3: Adaptar un cable RJ12.

Esta opción estaría pensada para reciclar cables antiguos. De este modo damos utilidad a un cable de tipo RJ12 ya que, eran utilizados para las conexiones telefónicas. A continuación, podemos ver la diferencia de ambos cables.

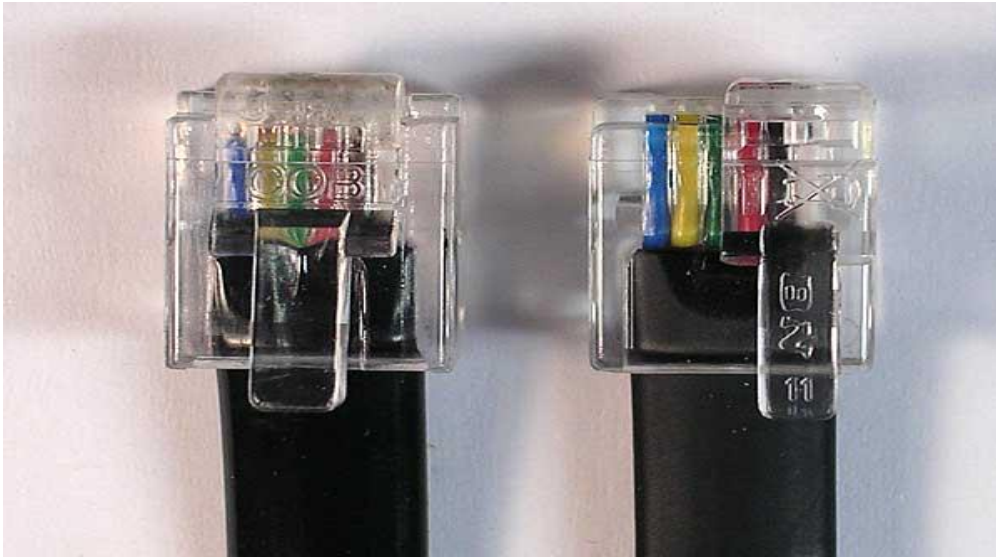


Figura 24: Cable RJ12 estándar (izquierda) Cable RJ12 LEGO (derecha).

Como podemos ver, la única diferencia es la posición de la pestaña. Para poder reciclar el cable y adaptarlo a LEGO, debemos de realizar los siguientes pasos:

1. Cortamos la pestaña, siguiendo la línea roja del cable RJ12 que queremos reciclar.

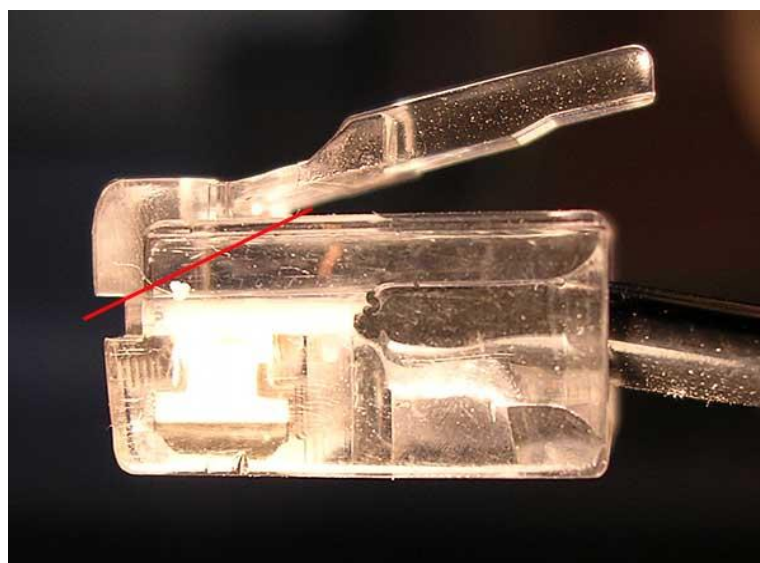


Figura 25: Dirección de corte para obtener la pestaña.

2. Una vez que hemos conseguido la pestaña, la lijamos hasta que encaje con la posición del conector de LEGO.

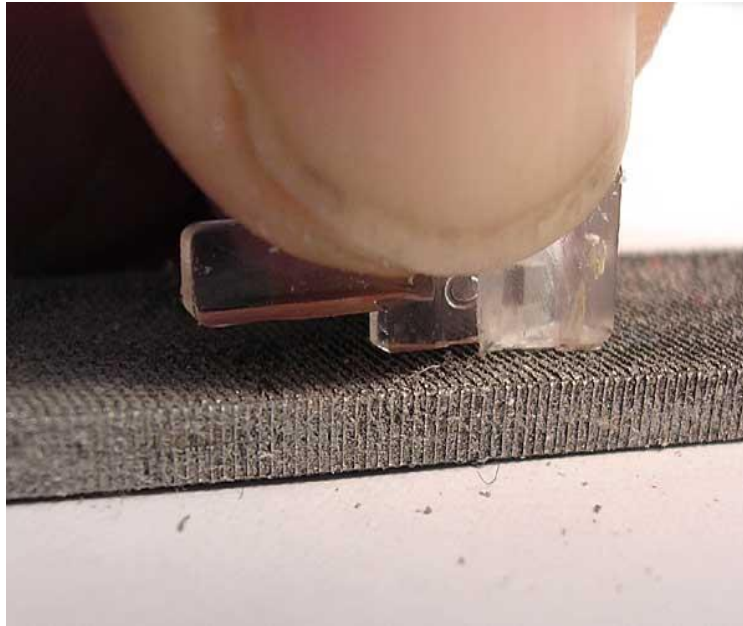


Figura 26: Lijamos la pestaña..



Figura 27: Comprobamos que encaja en el conector LEGO.

3. Finalmente pegamos la pestaña con la posición corregida. Esta es la parte más importante, se recomienda utilizar un buen pegamento o cola y una prensa para que haga presión, después dejarlo secar unas horas.



Figura 28: Pegamos la pestaña con la posición corregida.

Finalmente, ya tenemos el cable RJ12 adaptado y del mismo modo hemos ahorrado unos cuantos euros. Podemos ver la diferencia entre el cable de comunicación LEGO y el cable RJ12 adaptado en la siguiente imagen:

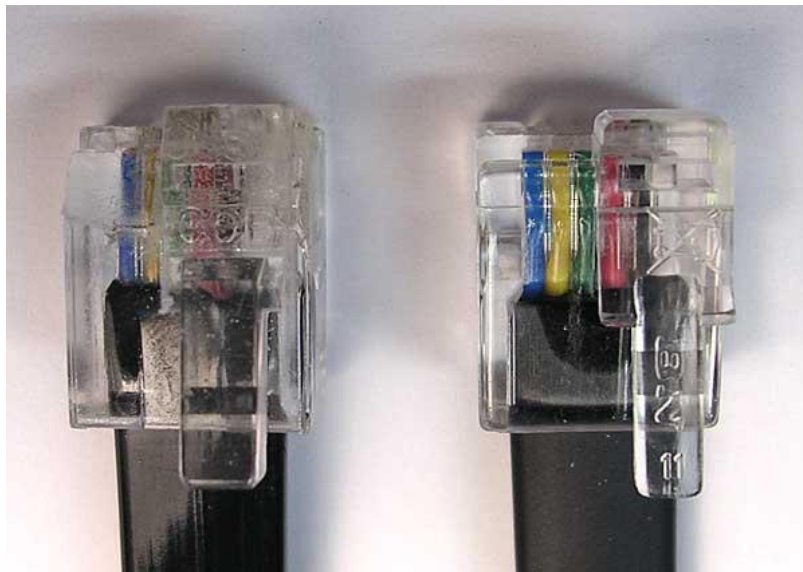


Figura 29: Diferencia entre el cable RJ12 adaptado (izquierda) y el cable LEGO (derecha).

En cuanto a la otra punta del cable, debemos de realizar la operación descrita en el punto anterior, es decir, debemos de separar los cables y conectarlos a Arduino siguiendo el esquema de la figura 21.

3.1.4. Fabricar el adaptador nosotros mismos.

También podemos diseñar nosotros mismos una sencilla placa, que permita la comunicación entre LEGO y Arduino.

El principal problema es que no podemos comprar ningún conector hembra del cable de comunicación LEGO. Pero si podemos comprar un conector hembra de un cable del tipo RJ12 ya que presentan la misma similitud. La única diferencia consiste en la posición de la pestaña, la de LEGO tiene el lateral y el cable RJ12 en posición central. Podemos ver la diferencia en la siguiente imagen:



Figura 30: Diferencia entre conectores: conector LEGO (izquierda) conector RJ12 (derecha).

Para solucionar esto podemos comprar un conector hembra de un cable RJ12 y con un soldador ir retirando el plástico que no permita la correcta conexión del cable de comunicación de LEGO, podemos verlo en la siguiente imagen:

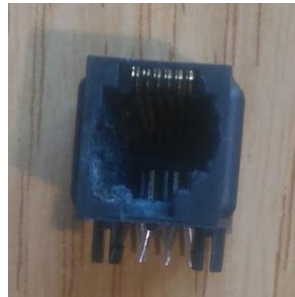


Figura 31: Conector RJ12 con plástico retirado.

Iremos retirando plástico hasta que encaje el cable de comunicación LEGO. Para comprobar que la comunicación es adecuada podemos comprobar la continuidad con un polímetro entre el cable y el conector.



Figura 32: Conexión entre el cable de LEGO y conector RJ12.

Una vez esto, diseñamos la placa siguiendo el esquema de la figura 21. La placa resultante, es la siguiente:

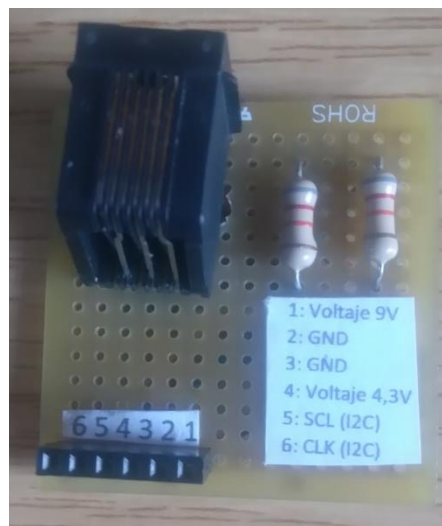


Figura 33: placa de comunicación entre LEGO EV3 y Arduino.

Finalmente conectamos esta placa a Arduino con cables realizando la misma conexión vista en la opción 1. Esta conexión sería la siguiente:

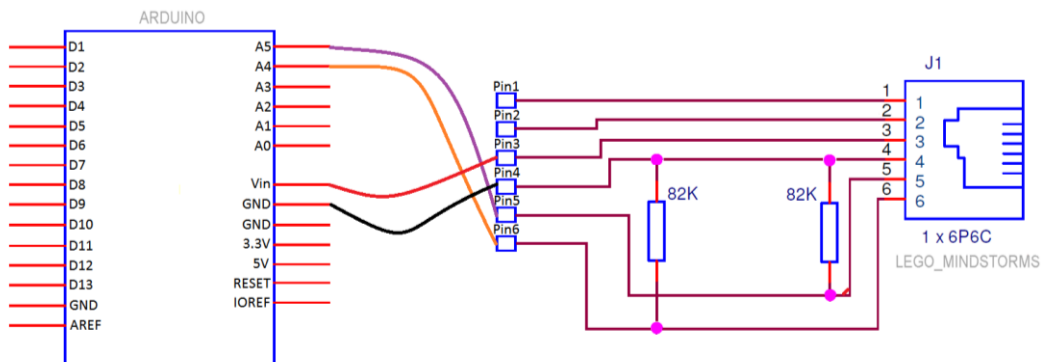


Figura 34: Comunicación LEGO Arduino.



Figura 35: Conexión entre Arduino y LEGO.

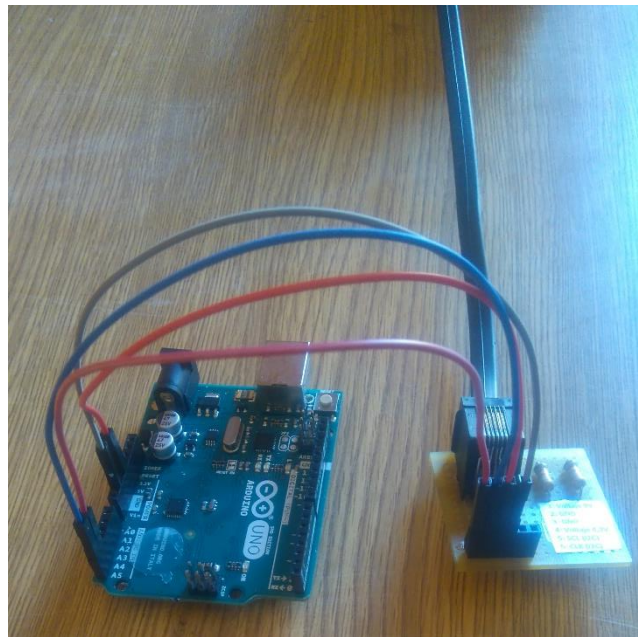


Figura 36: Conexión entre Arduino y LEGO 2.

3.2. Programación mediante LEGO.

Una vez que tenemos la interfaz física de comunicación conseguida, debemos programar ambos dispositivos para que sean capaces de comunicarse entre ellos, como hemos comentado anteriormente, esta comunicación se realizará mediante el Protocolo I²C.

Ahora, vamos a explicar cómo y de qué forma debemos programar nuestro LEGO EV3 para que sea capaz de comunicarse con Arduino, posteriormente veremos cómo debemos de programar Arduino para que sea capaz de comunicarse con LEGO.

3.2.1. Primeros pasos.

Para poder programar con el software de LEGO Mindstorms debemos de utilizar unos bloques que permiten la comunicación I²C entre LEGO y Arduino que han sido desarrollados mediante LabView EV3 por la empresa Dexter Industries. Para utilizar dichos bloques, seguiremos los siguientes pasos:

1. Descargamos e instalamos el software LEGO Mindstorms disponible en la página web del fabricante, dada en las referencias (LEGO, 2017).
2. Ahora descargamos un archivo llamado Dexter.ev3, en la siguiente página (Industries, Link de descarga bloque I2C, 2017).
3. Una vez descargado, abrimos el programa LEGO Mindstorms y seguiremos la siguiente ruta:

3.1. Pulsamos en ‘herramientas’.

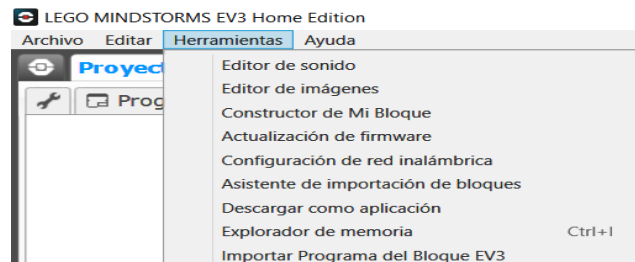


Figura 37: Importar bloques en LEGO MINDSTORMS 1.

3.2. Para luego pulsar en ‘Asistente de importación de bloques’.

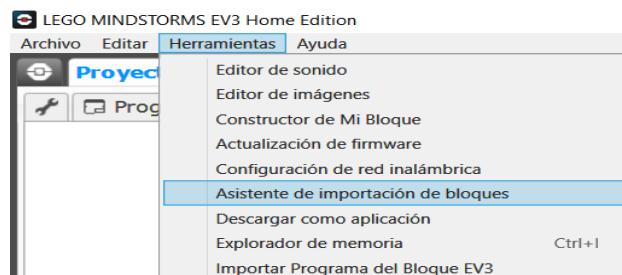


Figura 38:: Importar bloques en LEGO MINDSTORMS 2.

3.3. Finalmente seleccionamos el fichero Dexter.ev3 y seleccionamos en ‘importar’.

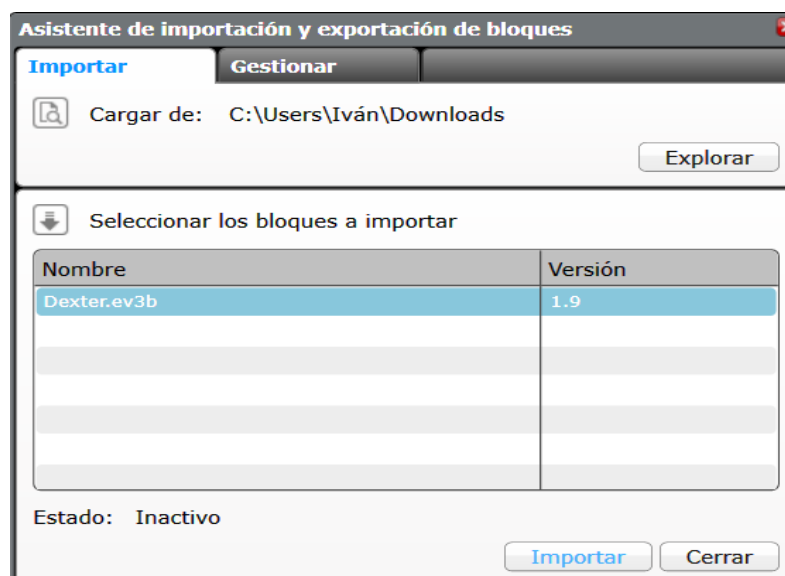


Figura 39: Ilustración 29: Importar bloques en LEGO MINDSTORMS 3.

3.4. El bloque que acabamos de importar, se llama EV3_I2C y lo podemos seleccionar en la pestaña amarilla de bloques de sensores, su imagen es la siguiente:



Figura 40: Bloque I2C desarrollado por Dexter Industries.

3.2.2. EL bloque EV3_I2C.

Como he comentado anteriormente este bloque está desarrollado por una empresa privada mediante Labview y tiene siete posibilidades de programación, aunque solo vamos a utilizar dos.

Estas siete posibilidades son:

- *Read 1 Byte*: Lee un byte desde Arduino.
- *Write 1 byte*: Escribe un byte en el Arduino
- *Write 8 bytes*: Escribe 8 bytes en un mensaje y lo manda a Arduino.
- *Read 8 bytes*: Lee 8 bytes desde el Arduino.
- *Read 8 bytes ASCII*: Lee 8 bytes como caracteres desde Arduino. Este bloque no funciona bien, aún está en desarrollo.
- *Analog Read*: lee el valor desde el pin analógico del Arduino.
- *Digital Write*: Escribe un estado digital en un pin del Arduino.

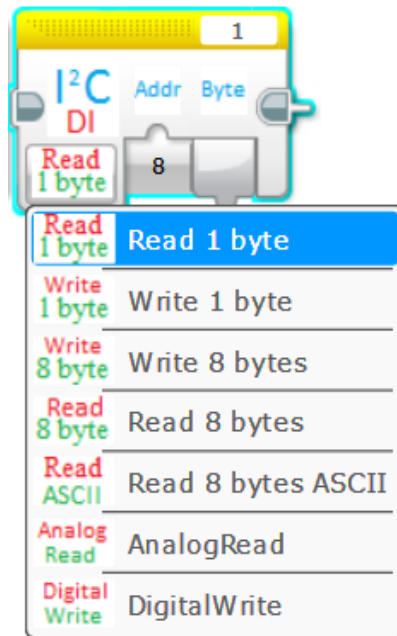


Figura 41: Bloque I2C.

Como podemos ver tenemos varias posibilidades. No obstante, vamos a realizar el presente proyecto utilizando solo dos bloques, para lograr un proceso metódico y de fácil utilización.

Concretamente lo que vamos hacer es “codificar” todos los parámetros de nuestras instrucciones en los bytes de estos dos bloques que vamos a utilizar:

- **Write 8 bytes**. Este bloque será utilizado para enviar datos del LEGO EV3 al Arduino. Cada byte contendrá la siguiente información:

Write_8_Bytes (Dirección_Arduino, Código_Dispositivo, PIN_numero, valor_especial, opcional_byte1, opcional_byte2, 0, 0 ,0)

- Dirección Arduino: es la dirección que identifica al esclavo, si por ejemplo, queremos que nuestro bloque EV3 se comunice con tres Arduinos está dirección deberá ser diferente para estos tres. Esta dirección normalmente estará en hexadecimal.
- Código dispositivo: aquí indicamos que tipo de dispositivo estamos configurando:
 - 1: LED.
 - 2: Servo_motor.
 - 3: DC_motor.
 - 4: Sensor.
- PIN número: este parámetro lo podemos utilizar de dos formas. La primera es elegir directamente a que PIN va estar conectado nuestro dispositivo. También lo podemos considerar para identificar el dispositivo que vamos a utilizar. Si estamos construyendo un proyecto y no tenemos claro a que pin de nuestro Arduino va a estar conectado cada actuador y sensor, este byte identificara cada uno. Por ejemplo, motor1, motor2, motor3, LED1, LED2, etc.
- Valor especial: Este valor varía del positivo. Por ejemplo, para un Servo Motor será el ángulo, para un DC motor la velocidad, para un LED su estado... De la misma forma serán los bytes opcionales.

Este bloque presenta el siguiente aspecto:

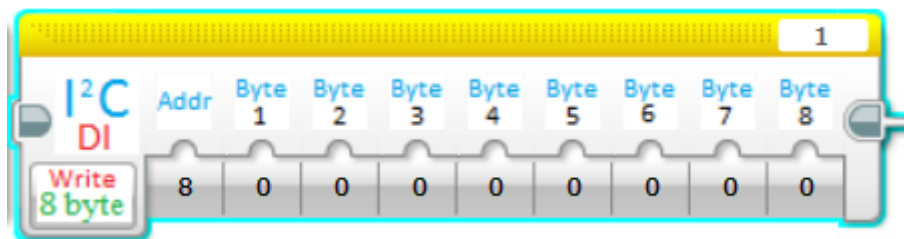


Figura 42: Bloque I2C Write 8 bytes.

- **Read 1 byte.** Este bloque será utilizado para enviar datos del Arduino al LEGO. Cada byte contendrá la siguiente información:

Read_1_Byte (Dirección_Arduino, Salida)

- Dirección Arduino: es la dirección del esclavo, en este caso Arduino, siempre debemos colocarla en todos los bloques que diseñemos. Esta dirección es 0x04 en hexadecimal, para este esclavo.
- Salida: en este byte enviaremos la salida del sensor (el valor) que estemos utilizando. Por ejemplo, la salida de un interruptor (1 activo, 0 apagado), un sensor de infrarrojo, un LDR...

Este bloque presenta el siguiente aspecto:

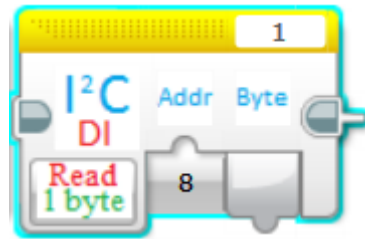


Figura 43: Bloque I2C Read 1 byte.

Después de cada uno de los dos bloques anteriores, es muy importante poner un tiempo suficiente para que a LEGO pueda tanto recibir como enviar datos, con 0,05 segundo será suficiente. El bloque que debemos de insertar es el siguiente:

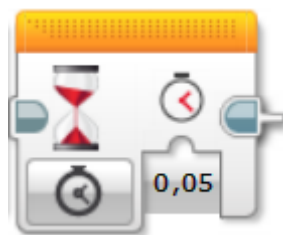


Figura 44: Bloque de tiempo.

Cabe destacar que, solo podemos enviar y recibir valores entre 0 y 128. Si nuestro sensor supera el valor 128 los datos enviados serán erróneos. Por lo tanto, si queremos que LEGO sea capaz de operar con sensores analógicos, tenemos que tener esto presente para ajustar la escala en Arduino.

3.2.3. Construcción de bloques Arduino.

En este punto, vamos a explicar cómo construir bloques con software LEGO Mindstorms para que sean compatibles con Arduino. En lo que se refiere a términos de programación, estos bloques equivaldrían a funciones.

Cabe destacar, que estas funciones o bloques de función no se pueden exportar a otro proyecto. Es un inconveniente que presenta LEGO Mindstorms, para ello lo que podemos hacer es crear un proyecto base solo para crear todos los bloques de función que vayamos a utilizar y en futuro utilizar este proyecto base para construir todos los proyectos que queramos.

Vamos a realizar un ejemplo de un bloque que encienda un LED conectado a Arduino. Para empezar, debemos de seleccionar el bloque EV3_I2C y elegir la opción "Write 8 bytes", ya que vamos a mandar información de LEGO EV3 a Arduino (el estado del LED). Y a continuación poner un bloque de tiempo. Debe de quedar de la siguiente forma:

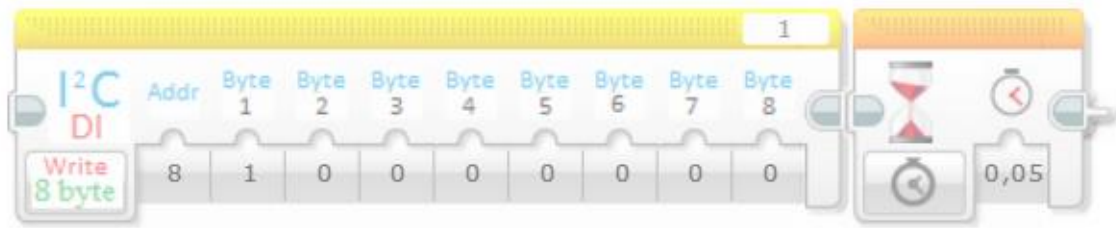


Figura 45: Construcción de un bloque LED.

Podemos apreciar que ambos bloques aparecen translucidos, esto es porque no hemos conectado ningún bloque de inicio. Para poder crear un nuevo bloque personalizado debe de ser así (sin bloque de inicio).

Una vez esto, seleccionamos ambos bloques y accedemos a “Herramientas” y luego en “Constructor de Mi bloque”

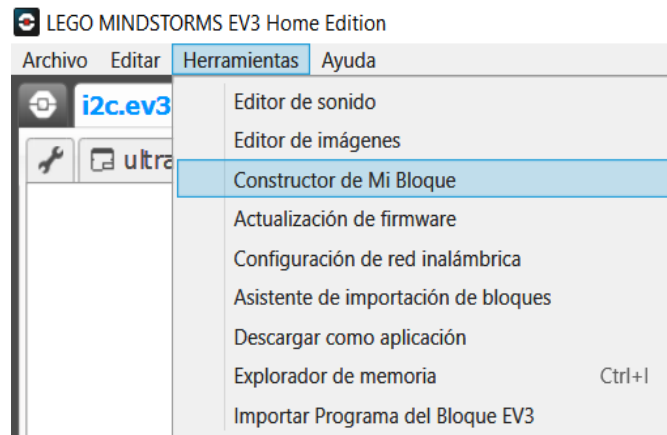


Figura 46: Ruta para crear un bloque personalizado.

Nos aparecerá la siguiente ventana:

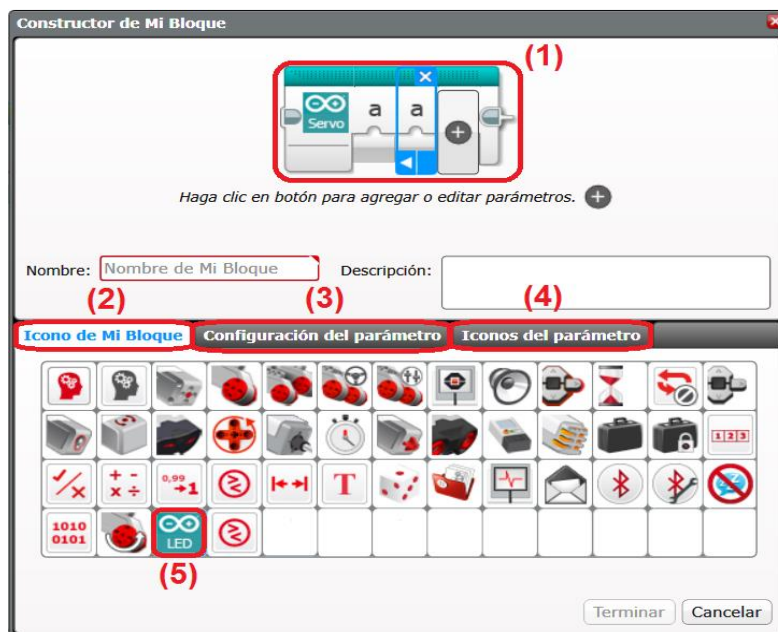


Figura 47: Constructor de mi bloque.

Vamos a explicar las partes de esta importante ventana:


- 1) Esta es la imagen final que tendrá nuestro bloque, para añadir nuevas funciones debemos pulsar en .
- 2) Icono de Mi bloque: aquí seleccionamos la imagen que aparecerá en el bloque.
- 3) Configuración del parámetro: En este apartado configuramos las entradas y salidas que tendrá nuestro bloque.



Figura 48: Configuración del parámetro.

Tenemos varias posibilidades:

- Tipo de parámetro: parámetro de entrada: el valor es introducido mediante programación. Parámetros de salida su valor no es introducido mediante programación
 - Tipo de dato: puede ser: numérico, lógico, texto, secuencia lógica y secuencia numérica.
 - Valor por defecto: establecer el valor en condiciones iniciales.
- 4) Iconos del parámetro: a cada entrada o salida que hemos añadido en el punto anterior le podemos asignar un pequeño dibujo para una programación más visual.
 - 5) Si queremos añadir una imagen específica para nuestro bloque, debemos de guardarla en la siguiente dirección:

C:\Program Files\LEGO Software\LEGO MINDSTORMS EV3 Home Edition\Resources\MyBlocks\images

Podemos coger cualquier imagen ya creada para respetar los pixeles necesarios y editarla a nuestro gusto con un programa como Paint. Para que esto sea posible es importante señalar que son tres imágenes las cuales, debemos de modificar:



Figura 49: Imágenes para los bloques en LEGO MINDSTORMS

El número del nuevo dispositivo que diseñemos (subrayado en azul) no debe coincidir con cualquier otro. Del mismo modo el nombre tampoco debe coincidir con cualquier otro (subrayado en naranja). Lo demás no debe de ser alterado o cambiado ya que, si no, LEGO Mindstorms no lo reconocerá (subrayado en rojo). Es importante señalar que para que las imágenes de los bloques (LED, Interruptor, Sensor...) sean visibles en el software de LEGO Mindstorms debemos de pegar la carpeta "Images" en la siguiente ruta:

C:\Program Files (x86)\LEGO Software\LEGO MINDSTORMS EV3 Home Edition\Resources\MyBlocks\images

Dicha carpeta la podemos descargar en el enlace disponible en Referencias en un archivo comprimido.

La extensión de las imágenes debe ser .png. Cada imagen tiene una distinta función:

Diagram: es la imagen del bloque que aparece en la programación.



Figura 50: Icono programación.

Palette: es la imagen del bloque que aparece en la paleta de bloques.



Figura 51: Paleta de bloques.

Palette_MouseOver: es la imagen que aparece en la paleta de bloques cuando pasamos el ratón por encima.

Retornando a al diseño de nuestro LED, la configuración sería la siguiente:

1. Dirección [Entrada].
2. PIN del LED conectado a Arduino [Entrada].
3. Estado del LED: encendido (true) o apagado (false) [Entrada].

Una vez configurado, nos creará el siguiente bloque:



Figura 52: Bloque de programación LED.

Y nos aparecerá la siguiente pantalla:

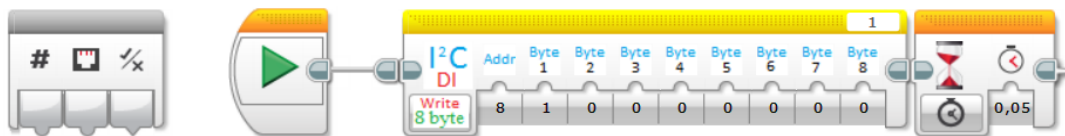


Figura 53: Diseño del bloque LED 1.

Ahora simplemente debemos unir, cada entrada de nuestro bloque de función (bloque gris) con cada byte explicado anteriormente, una vez realizado esto quedará de la siguiente forma:

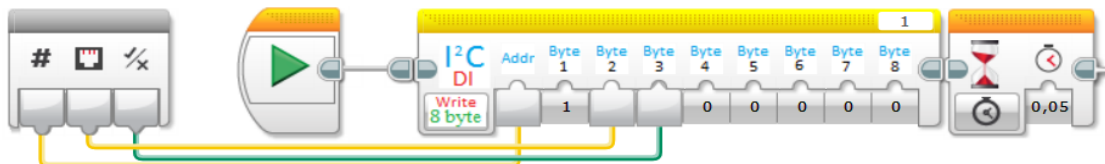


Figura 54: Diseño del bloque LED 2

Es importante escribir en el byte 1, el tipo de dispositivo que tenemos (1, 2 ,3 o 4), en este caso un 1 (explicado anteriormente).

Una vez realizado esto, ya podemos acceder a nuestro bloque personalizado en la pestaña azul “Mis bloques”



Figura 55: Bloque Arduino LED 1.

Al usar dicho bloque tendremos las tres entradas anteriores que debemos de completar antes de ponerlo en funcionamiento:



Figura 56: Bloque Arduino LED 2.

La primera casilla, como hemos comentado antes, es la dirección del esclavo. La segunda entrada es el PIN que está conectado el LED en el Arduino, por último, el estado del LED que queremos asignarle apagado (false) o encendido (true).

3.2.4. Programación de los bloques de Arduino con LEGO MINDSTORMS.

A continuación, podremos ver una serie de ejemplos, de construcción de bloques de función. Toda la conexión con Arduino de estos sensores y actuadores la podemos encontrar en el Anexo 1.

3.2.4.1. LED.

- Programación del bloque: Este bloque ya lo hemos explicado anteriormente.

Los bytes de este bloque presentan la siguiente información:

- ❖ *Write_8_bytes* (Dirección, Código dispositivo, PIN número, Valor especial,0,0,0,0)

Donde el valor especial es el estado del LED.

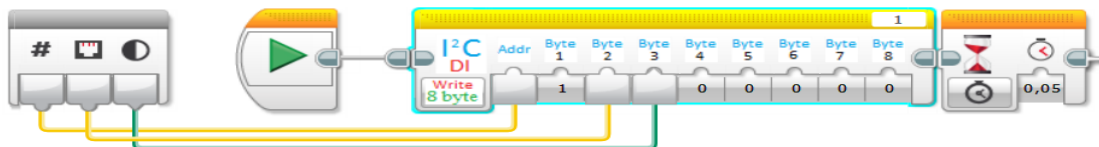


Figura 57: Programación bloque LED.

- Ejemplo de programación:

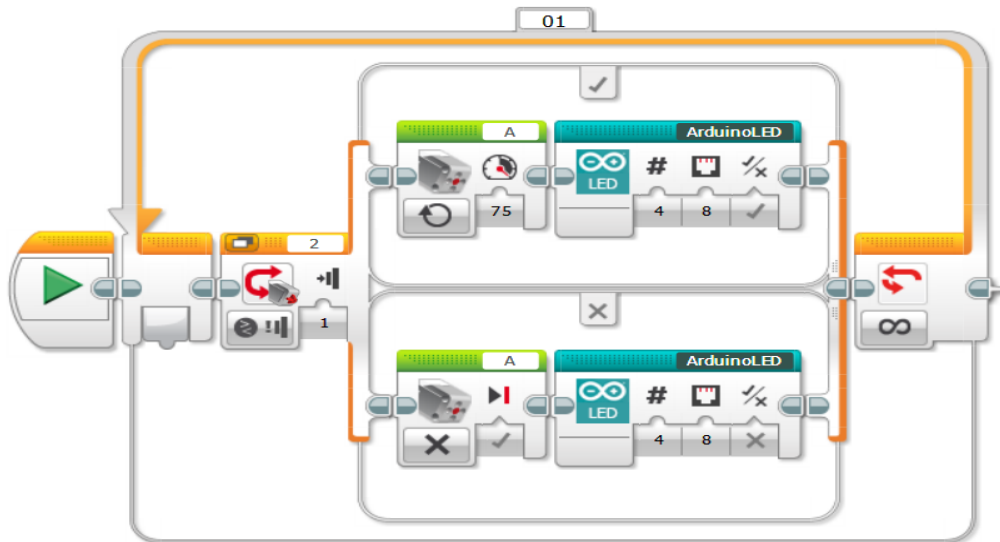


Figura 58: Ejemplo bloque LED.

Este es un ejemplo sencillo en el cual podemos comprobar la comunicación de LEGO a Arduino. Si presionamos el botón táctil, se encenderá el LED conectado a Arduino (en el PIN digital 8) y al mismo tiempo el motor. Podemos comprobar que no hay ningún problema de incompatibilidad entre los bloques de Arduino y los originales de LEGO.

3.2.4.2. Sensor

- Programación del bloque:

Los bytes de este bloque presentan la siguiente información:

- ❖ *Write_8_bytes* (Dirección, Código Dispositivo, PIN Número, Valor especial, 0,0,0,0)

Valor Especial es el tipo de sensor, analógico o digital.

- ❖ *Read_1_byte* (Dirección Arduino, Salida)

Este caso, es parecido al bloque del LED, pero en vez de tener la entrada Estado del LED hemos añadido la entrada Analógico/Digital. Con esta entrada configuraremos el bloque para indicar que se trata de un sensor analógico (*true*) o un sensor digital (*false*). También hemos añadido un nuevo bloque *Read 1 byte* el cual enviará a LEGO el valor del sensor. Ambos bloques *Write 8 byte* y *Read 1 byte* deben de tener la misma dirección (siempre la dirección es 4).

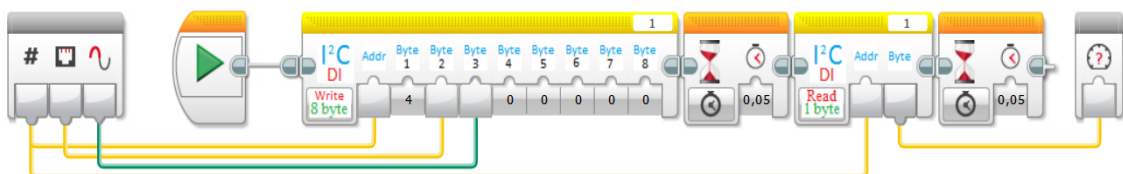


Figura 59: Programación bloque sensor.

Como hemos comentado anteriormente, este bloque lo podemos utilizar para sensores analógicos como LDR, fotorresistencias, NTC, PTC, LDR, etc. o también lo podemos utilizar para sensores digitales como puede ser un interruptor. Como los interruptores son comúnmente utilizados, he creado un bloque cuya programación es idéntica a la anterior, pero con el nombre de interruptor. Podemos observar que como el interruptor es un sensor digital, la tercera casilla es true.



Figura 60: Bloque interruptor.

- Ejemplos de programación:

En primer lugar, tenemos un ejemplo de un interruptor conectado a Arduino. La salida del sensor está conectado a un interruptor o IF. Como es un sensor digital si el interruptor está activado Arduino envía a LEGO un 1 y se activa el motor y el LED. Este es un buen ejemplo, para comprobar que LEGO y Arduino envían información, por una parte, Arduino le envía el estado del interruptor a LEGO y este, envía a Arduino la activación del LED.

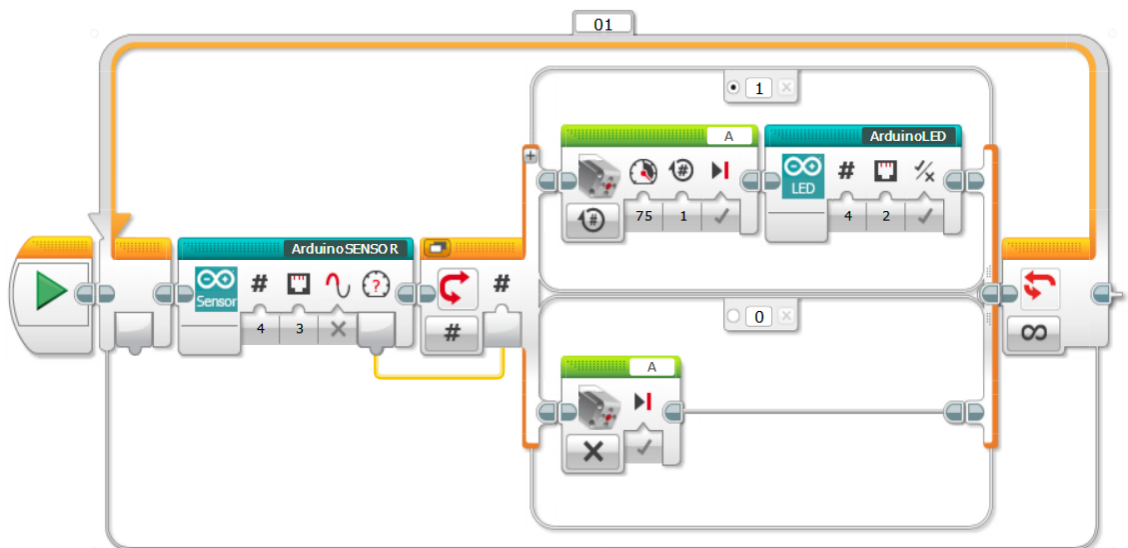


Figura 61: Ejemplo bloque sensor 1.

El siguiente ejemplo, es un sensor analógico como el sensor de temperatura. Este sencillo ejemplo muestra la temperatura captada por la pantalla del LEGO EV3, y además si la temperatura excede 25°C encenderá un LED.

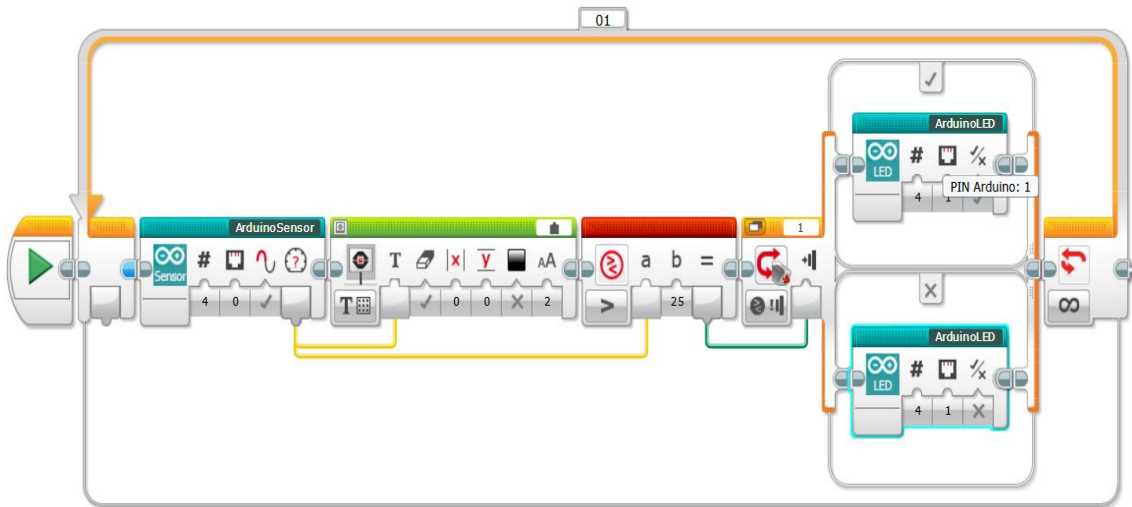


Figura 62: Ejemplo bloque sensor 2.

3.2.4.3. Sensor infrarrojo.

- Programación del bloque:

Los bytes de este bloque presentan la siguiente información:

- ❖ *Write_8_bytes* (Dirección, Código Dispositivo, PIN Número, Valor especial,0,0,0,0)

Este bloque representa al diodo emisor infrarrojo, tiene las mismas características que el bloque LED.

- ❖ *Write_8_bytes* (Dirección, Código Dispositivo, PIN Número, Valor especial,0,0,0,0)

Este bloque representa al receptor infrarrojo, tiene las mismas características que el bloque del sensor.

- ❖ *Read_1_byte* (Dirección Arduino, Salida)

Por último, este bloque envía el valor de la salida del receptor infrarrojo a LEGO EV3.

En este ejemplo hemos unido los dos anteriores y hemos creado en su conjunto un sensor detector de presencia. Del mismo modo hemos utilizado un diodo emisor de infrarrojo y un diodo receptor de infrarrojo. Para el diodo emisor lo hemos tratado como si fuera un LED y para el receptor como un sensor analógico y se trata básicamente de unir ambos en un solo bloque. La programación en LEGO MINDSTORMS la podemos ver en la siguiente imagen:

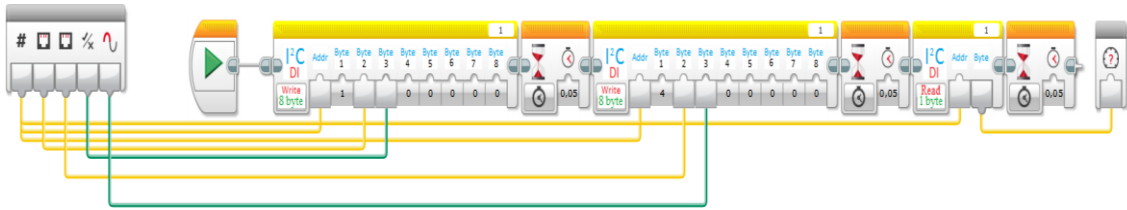


Figura 63: Programación sensor de infrarrojo.

Como podemos ver se trata de una combinación del bloque del sensor y del LED.

- Ejemplos de programación:

Con el ejemplo siguiente consiguiremos encender un LED en función de la incidencia del diodo emisor infrarrojo en el diodo receptor. De esta forma si el valor del diodo receptor está por debajo del valor 5 el LED se encenderá.

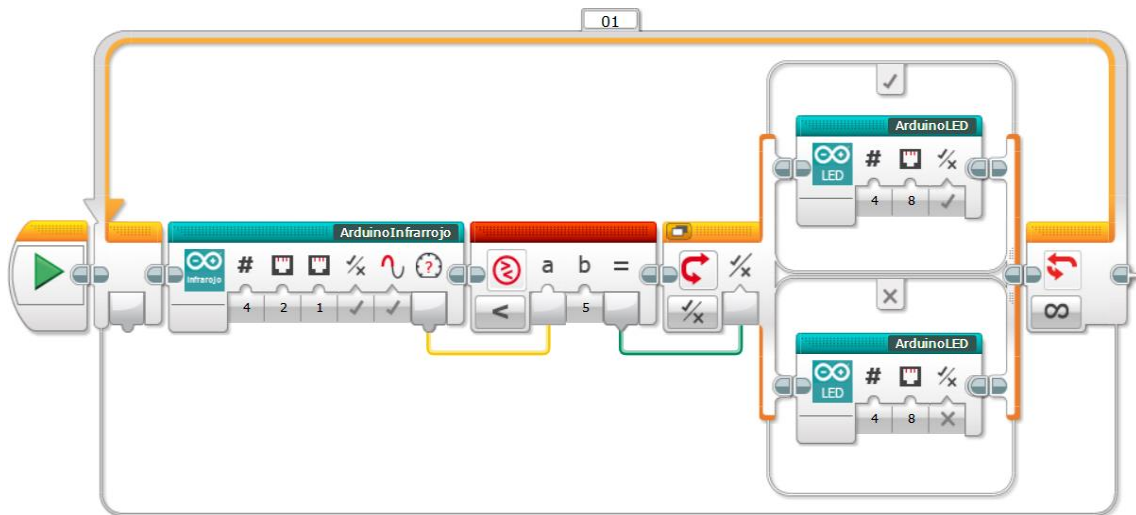


Figura 64: Ejemplo programación bloque infrarrojo.

3.2.4.4. Servo Motor.

- Programación del bloque:

Los bytes de este bloque presentan la siguiente información:

- ❖ **Write_8_bytes** (Dirección, PIN motor, ángulo, 0, 0,0,0,0,0)

Como he comentado anteriormente, solo podemos intercambiar datos en una escala de 0 a 124, por lo tanto, no podemos representar un rango de [0,180]. Para solucionar esto, enviaremos la mitad del ángulo insertado, por lo que ahora la escala será de [0,90]. Esto hará que perdamos un poco de precisión en los ángulos impares. Por ejemplo, si queremos un ángulo de 90°, en el software LEGO MINDSTORMS introduciremos eso 90° pero enviaremos 45° y en el Arduino volveremos a multiplicar este valor por dos. Si en vez de 90° fuera 19° en realidad el servo se moverá 18° por lo que en valores impares podremos tener un grado de imprecisión. Finalmente, la programación en LEGO MINDSTORMS es la siguiente:

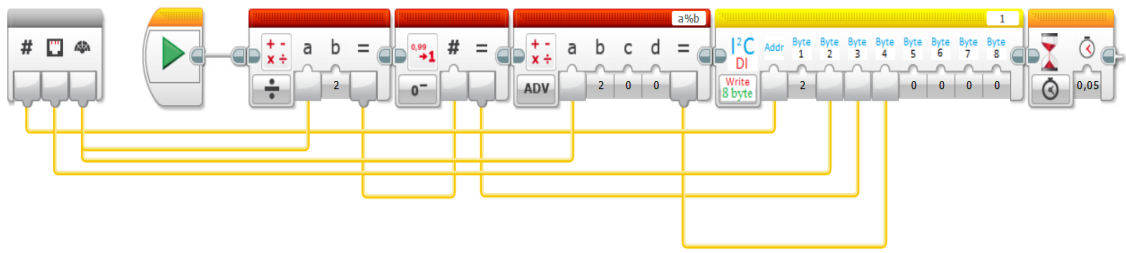


Figura 65: Programación Servomotor

- Ejemplos de programación:

La programación de un servo es muy sencilla, en el siguiente ejemplo usaremos el sensor infrarrojo y si en dicho sensor incide una luz superior a 5 el servo girará 90º si es inferior a 5 girará 180º:

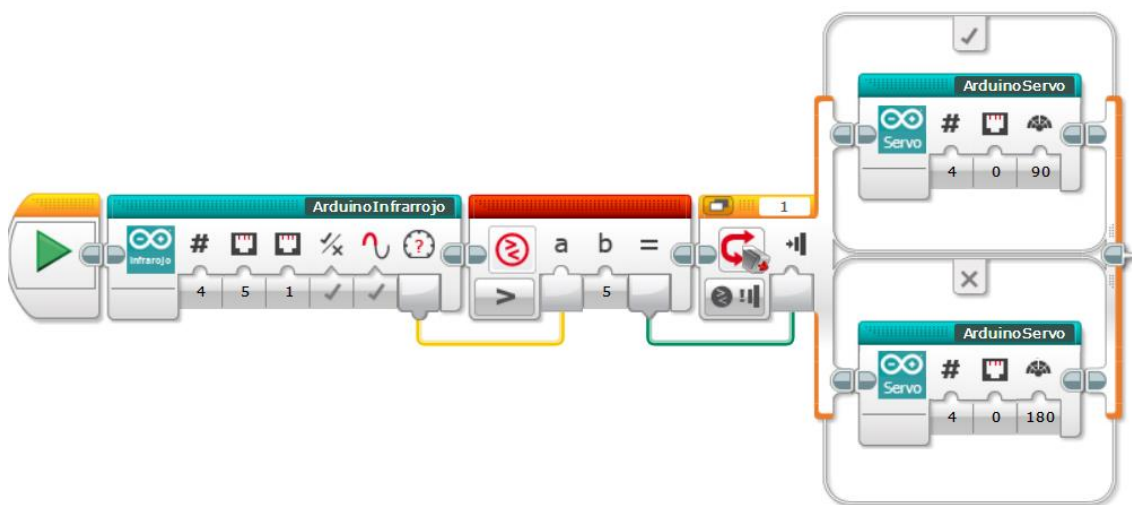


Figura 66: Ejemplo Servomotor.

3.2.4.5. Motor DC.

- Programación del bloque:

Los bytes de este bloque presentan la siguiente información:

- ❖ *Write_8_bytes* (Dirección, PIN Número 1, PIN número 2, PIN Número 3, Valor especial,0,0,0,0)

Donde:

- PIN Número 1: Es el pin en el cual hemos conectado uno de los bornes del motor.
- PIN Número 2: Es el pin en el cual hemos conectado el otro borne del motor.
- PIN Número 3: Es el pin status del motor, es el que enciende o apaga el motor y es el que nos permite regular la velocidad. Este pin siempre debe estar conectado a una salida digital PWM del Arduino.

- Valor especial: Este valor será la velocidad del motor y podemos asignarle en un rango de 0 a 100.

La programación en LEGO MINDSTORMS la podemos ver en la siguiente imagen:

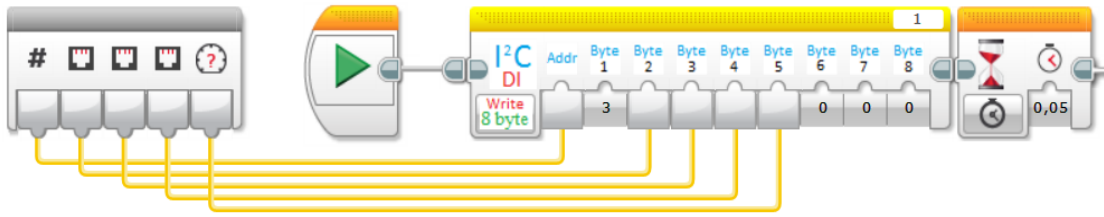


Figura 67: Programación Motor CC.

Este bloque no solo permite regular la velocidad de los motores conectados sino también el sentido de giro. El PIN Número 1 será el positivo siempre, por lo tanto, para invertir el sentido debemos de intercambiar los pines, en siguiente punto veremos un ejemplo de ello.

Por otro lado, me gustaría comentar un posible fallo que podemos tener a la hora de programar motores. Como sabemos, la comunicación se efectúa mediante mensajes que posteriormente son decodificados para obtener la función a realizar del dispositivo. Si solo mandamos un mensaje de activación del motor (como en la figura 66) el motor seguirá funcionando, incluso cuando no haya comunicación física entre LEGO y Arduino, ya que lo último que obtuvo Arduino fue dicho mensaje con la activación del motor correspondiente.

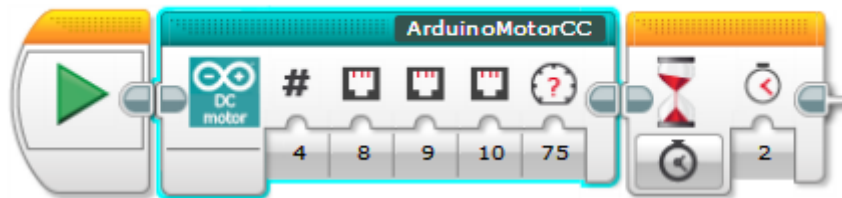


Figura 68: Error a la hora de programar motores 1.

Esto lo debemos corregir insertando un nuevo bloque, pero con su velocidad cero, como la imagen 67. De esta manera, Arduino recibe un mensaje con su activación y otro con su desactivación cuando lo requiera. Siempre hay que insertar un bloque para actualizar la nueva velocidad o el nuevo sentido de giro del motor.



Figura 69: Error a la hora de programar motores 2.

- Ejemplos de programación:

Seguiremos el esquema explicado anteriormente a la hora de conectar el motor a Arduino. El primer ejemplo nos permite incrementar la velocidad del motor progresivamente:



Figura 70: Ejemplo de programación de motor 1.

A continuación, pondremos otro ejemplo en el cual, si pulsamos el botón táctil de LEGO el motor girará en un sentido y si no lo presionamos el motor girará en otro sentido. Podemos observar que el único cambio es la posición de los pines:

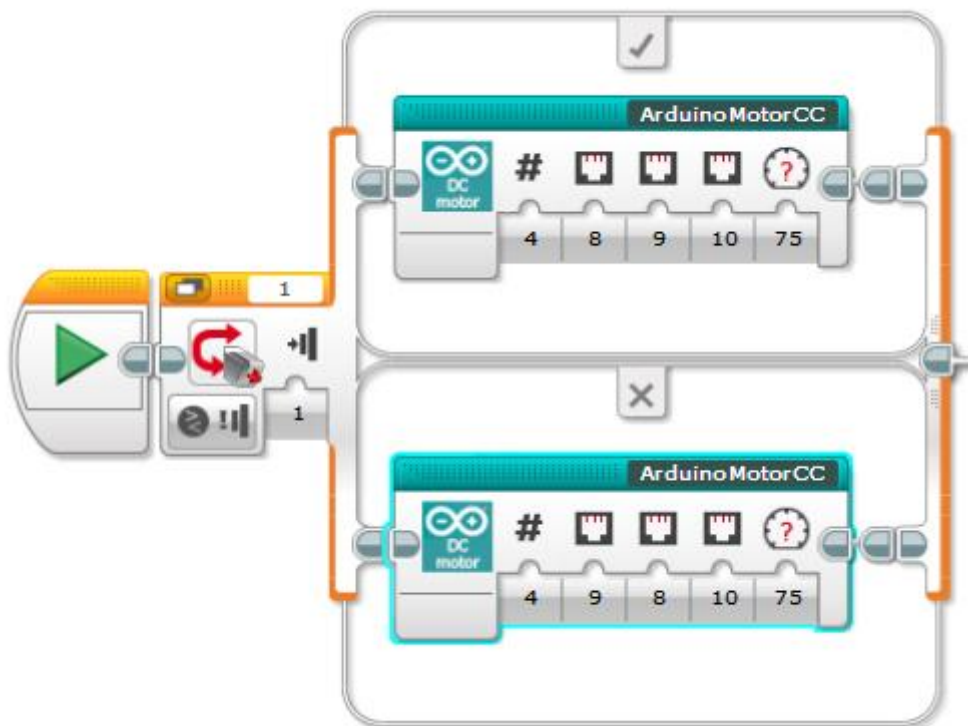


Figura 71: Ejemplo de programación de motor 2.

Finalmente, en el anexo 2, podemos encontrar una tabla resumiendo todos los bloques vistos hasta ahora. Por otro lado, en referencias podemos descargar un archivo comprimido con un proyecto de LEGO Mindstorms hecho por mí con todos estos bloques ya hechos. (Link de descarga del archivo comprimido, s.f.).

3.3. Programación mediante RobotC.

3.3.1. Primeros pasos.

Primero debemos descargarnos el software de RobotC disponible en la página oficial del autor (Link de descarga RobotC, s.f.).

Una vez descargado e instalado debemos configurar RobotC para que sea capaz de comunicarse con nuestro bloque EV3, para ello abrimos el programa “*Robot Virtual Worlds - LEGO 4.X*” que instalemos anteriormente.

Una vez abierto, debemos seleccionar que tipo de bloque LEGO vamos a utilizar, el programa por defecto utiliza el bloque NXT, sin embargo, nosotros utilizaremos el bloque EV3, para cambiar esto seguimos la siguiente ruta: Robot>>Platform Type>>LEGO Mindstorms>>LEGO Mindstorms EV3

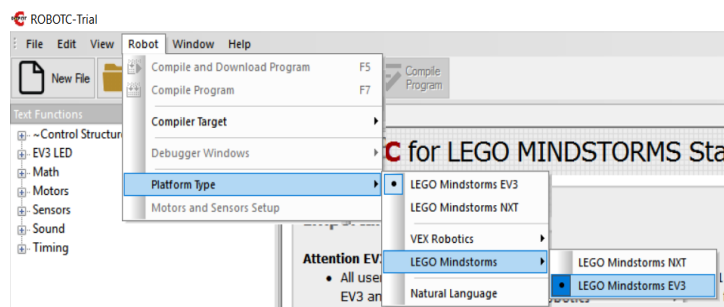


Figura 72: Elección del bloque EV3 en RobotC

Una vez hecho esto, debemos de actualizar el firmware del bloque EV3, para ello solo debemos de conectar mediante USB nuestro bloque EV3 al ordenador y acceder a la siguiente ruta:

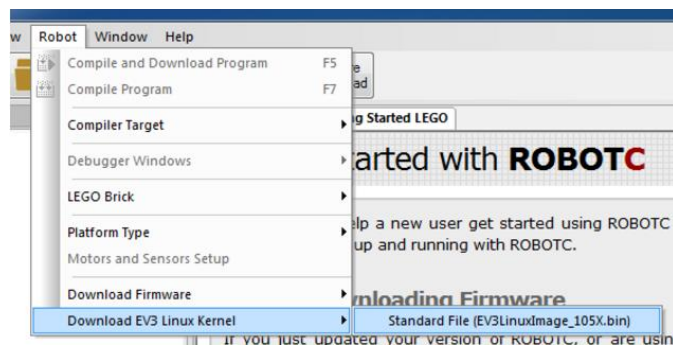


Figura 73: Actualización firmware RobotC

Este firmware que hemos instalado permite compatibilizar la programación RobotC y LabVIEW con nuestro bloque EV3, puede tardar unos cinco minutos.

Finalmente, instalamos la máquina virtual (VM) ROBOTC, para poder programar nuestro bloque EV3 con RobotC. Para ello seguimos la siguiente ruta: Robot>>Platform Type>>Download Firmware>>Standard File (librobortc.so)

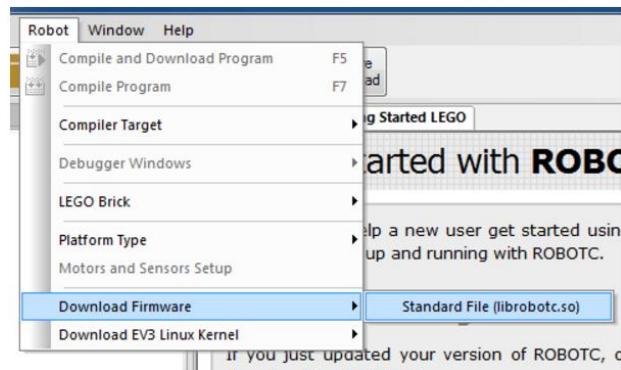


Figura 74: Instalación de la máquina virtual de RobotC.

3.3.2. Programación de RobotC.

Dividiremos el código, en tres partes: Comunicación I²C, Dispositivos y Programa y explicaremos cada parte por separado:

1. Comunicación I²C.

En esta primera parte del código establecemos la comunicación I²C con Arduino, esta comunicación estará establecida en la función "i2c_msg".

```
// Establecemos el puerto que vamos a establecer la
// comunicación con Arduino
#pragma config(Sensor, S1, TIR, sensorI2CCustom)

#define ARDUINO_PUERTO S1

//*****
//***** Comunicación I2C *****
//*****

// La dirección del esclavo (Arduino) es 0x04 en binario es
// 0100
// Como la comunicación entre RobotC y LEGO utiliza 7 bytes eb
// vez de 8 bytes desplazamos un byte: 1000 => 0x08
#define ARDUINO_ADDRESS 0x08 // Arduino: 0x04

ubyte mensajeI2C[22];
char respuestaI2C[20];

int msg_i2c(byte dir_arduino, int tam_mensaje, int
tam_respuesta, ubyte byte0, ubyte byte1, ubyte byte2, ubyte
byte3 ,ubyte byte4)
{
    memset(respuestaI2C, 0, sizeof(respuestaI2C));
    tam_mensaje = tam_mensaje+3;

    mensajeI2C[0] = tam_mensaje;
    mensajeI2C[1] = dir_arduino;

    mensajeI2C[2] = byte0;
    mensajeI2C[3] = byte1;
    mensajeI2C[4] = byte2;
    mensajeI2C[5] = byte3;
    mensajeI2C[6] = byte4;
}
```

```

    sendI2CMsg(S1, &mensajeI2C[0], tam_respuesta);
    wait1Msec(20);

    readI2CReply(ARDUINO_PUERTO, &respuestaI2C[0],
tam_respuesta);

    int x = respuestaI2C[0];

    wait1Msec(35);
    return x;
}

```

En primer lugar, declaramos las matrices “mensajeI2C” y “respuestaI2C” las cuales nos servirán como base a la hora de utilizar el Protocolo I²C.

Como podemos observar, el Protocolo I²C de RobotC varía con respecto al Protocolo I²C del software LEGO MINDSTORMS, esto es debido a que hay variantes de dicho Protocolo.

La codificación de 8 bits, es utilizada por RobotC y el ladrillo NXT y su estructura es la siguiente:

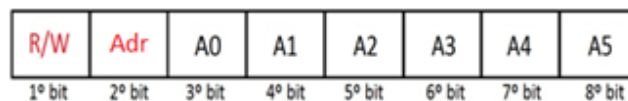


Figura 75: Protocolo comunicación I²C RobotC.

Esta codificación la utilizan algunos dispositivos para operar con el esclavo, en donde el bit 1 es uno si lee del esclavo, y es cero si escribe en el esclavo. La diferencia entre ambas variantes es que desplazamos un byte.

La auténtica codificación de 7 bits, es utilizada entre Arduino y el software de LEGO y es la que vimos anteriormente y la forma usual de operar con el Protocolo I²C.

El parámetro "tam_respuesta" representa el número de bytes que solicita del esclavo como respuesta a un mensaje enviado. Si sólo tenemos que enviar un mensaje al esclavo y no necesitamos una respuesta, debemos restablecerlo a 0.

El parámetro "tam_mensaje" representa el número de bytes de "datos" que desea enviar al esclavo. Como podemos observar, dentro de la función "i2c_msg", sumamos 3 bits a tam_mensaje. Estos 3 bytes son el encabezado del mensaje:

- dirección_esclavo.
- tam_mensajeI2C.
- tam_respuestaI2C.

La comunicación entre el maestro (LEGO EV3) y el esclavo (Arduino) es “enviar” y “solicitar” de forma predeterminada por lo que no hay instrucciones de “leer” y “escribir”, como hacíamos en la programación de LEGO MINDSTORM.

Programando mediante RobotC los datos captados por los sensores no estarían en un rango de [0,124], sino de [0,255].

Los bytes de datos son 5 bytes y podemos codificar lo que necesitemos en ellos. Después de varias pruebas, no he podido encontrar a una explicación de porqué el último byte debe de tener un valor máximo de 99, por lo tanto, si lo utilizamos debemos de tenerlo en cuenta, ya que a valores superiores de 99 el valor codificado enviado es erróneo.

2. Dispositivos.

En esta parte del código escribimos los dispositivos que vamos a tener en nuestro proyecto (sensores y actuadores). A la hora de realizar la programación de los sensores y actuadores, la realizaremos mediante funciones, una función por cada actuador o sensor que vaya a participar en nuestro proyecto.

Dicha función tendrá unas entradas, y dichas funciones estarán compuestas por un mensaje que contendrá la función que realizará nuestro sensor o actuador. Este mensaje lo codificaremos, con el Protocolo I²C pero esta vez de 7 bytes.

Como hicimos anteriormente en la programación de LEGO, vamos a utilizar como ejemplo, el control de un LED. Su programación estará compuesta por una función cuyas entradas son las variables necesarias para controlar dicho LED, en este caso son:

- Dirección del esclavo.
- Numero pin que está conectado el LED en el Arduino.
- Estado: uno encendido, cero apagado.

El objetivo de esta parte del código es declarar los dispositivos que vamos a utilizar, en la tercera parte del código desarrollamos nuestro programa e indicaremos exactamente el número del esclavo, el pin de conexión, así como su estado. Por lo tanto, la función presentaría la forma:

```
void set_LED(byte direccion, int num_pin , bool estado)
{
}
```

Finalmente, dentro de esta función enviaremos una matriz de parámetros con la estructura del Protocolo I²C de 7 bytes:

```
void set_LED(byte direccion, int num_pin , bool estado)
{
    msg_i2c(direccion, 2, 0, 1, num_pin, estado, 0, 0);
}
```

La información que presenta cada byte es:

- Byte 1: dirección del esclavo.

- Byte 2: corresponde al “tam_mensaje”. Es dos porque estamos enviando dos datos el número del pin donde el LED está conectado y el estado de dicho LED.
- Byte 3: corresponde al “tam_respuesta”. Es cero ya que no necesitamos ninguna respuesta del LED, solo le enviamos un estado.
- Byte 4: este byte también lo utilizábamos en la programación LEGO y representa el código para identificar qué tipo de dispositivo: 1. LED 2: Motor 3: Servomotor y 4: Sensor. Por lo tanto, tenemos un 1 que corresponde a un LED.
- Byte 5: número de conexión del LED al Arduino.
- Byte 6: estado del LED.
- Byte 7: no utilizado para este dispositivo.
- Byte 8: no utilizado para este dispositivo.

Ya tenemos este dispositivo caracterizado y listo para usar en la tercera parte del código.

A continuación, explicaré algunos ejemplos vistos anteriormente utilizando la programación del software de LEGO, podemos observar que la base es la misma:

3.3.2.1. LED. Este dispositivo ya lo hemos explicado anteriormente:

```
void set_LED(byte direccion, int num_pin , bool estado)
{
    msg_i2c(direccion, 2, 0, 1, num_pin, estado, 0, 0);
}
```

3.3.2.2. Sensor. Su programación sería la siguiente:

```
int leer_sensor(byte direccion, int num_pin, bool analog_digital)
{
    int valor = msg_i2c(direccion, 2, 1, 4, num_pin, analog_digital
, 0, 0);
    return valor;
}
```

Podemos observar que, en la función ahora tenemos otra variable que es la variable analógica/digital, la misma que utilizábamos con la programación con LEGO MINDSTORMS.

La información que presenta cada byte es:

- Byte 1: dirección del esclavo.
- Byte 2: corresponde al “tam_mensaje”. Es dos porque estamos enviando dos datos el número del pin donde el sensor está conectado y la variable que nos caracteriza el sensor como analógico o digital.
- Byte 3: corresponde al “tam_respuesta”. En este caso es uno ya que la función devuelve un valor, en este caso el valor del sensor que conectemos.
- Byte 4: este byte también lo utilizábamos en la programación LEGO y representa el código para identificar qué tipo de dispositivo: 1. LED 2: Motor 3: Servomotor y 4: Sensor. Por lo tanto, tenemos un 4 que corresponde a un LED.
- Byte 5: número de conexión del sensor al Arduino.

- Byte 6: valor de la variable análogo/digital: uno sensores analógicos, cero sensores digitales.
- Byte 7: no utilizado para este dispositivo.
- Byte 8: no utilizado para este dispositivo.

3.3.2.3. Servomotor. Su programación sería la siguiente:

```
void set_servo_motor(byte direccion, int num_pin, int angulo)
{
    msg_i2c(direccion, 2, 0, 2, num_pin, angulo, 0, 0);
}
```

Este dispositivo sigue la misma estructura que el caso del LED, solo que, en vez de controlar su estado, controlamos el ángulo del servomotor.

3.3.2.4. Motor CC. Su programación sería la siguiente:

```
void set_motor_cc(byte direccion, int num_pin1, int num_pin2, int
pin_enable,int velocidad)
{
    msg_i2c(direccion, 3, 0, 4, num_pin1, num_pin2, pin_enable,
velocidad);
}
```

De igual forma este dispositivo sigue la misma base que el LED, en este caso tenemos 4 entradas por lo que ponemos un 4 en el byte 4, y un 3 en byte 2 al ser un motor DC.

Por lo tanto, esta segunda parte del código quedaría de la siguiente forma:

```
//*****
//***** Dispositivos *****
//*****

void set_LED(byte direccion, int num_pin , bool estado)
{
    msg_i2c(direccion, 2, 0, 1, num_pin, estado, 0, 0);
}

//_____
void set_servo_motor(byte direccion, int num_pin, int angulo)
{
    msg_i2c(direccion, 2, 0, 2, num_pin, angulo, 0, 0);
}

//_____

void set_motor_cc(byte direccion, int num_pin1, int num_pin2, int
pin_enable,int velocidad)
{
    msg_i2c(direccion, 3, 0, 4, num_pin1, num_pin2, pin_enable,
velocidad);
}

//_____

int leer_sensor(byte direccion, int num_pin, bool analog_digital)
```

```

{
    int valor = msg_i2c(direccion, 2, 1, 4, num_pin, analog_digital
, 0, 0);
    return valor;
}

```

Este último “leer_sensor”, es la función que nos permite leer los valores del sensor, como podemos observar el tercer pin es un 1, indicando que envía información a LEGO.

3. Programa.

Una vez que hemos establecido el Protocolo I²C y tenemos definidos todos los dispositivos que vamos a utilizar en nuestro proyecto, podemos a empezar a escribir el código que controlará nuestro proyecto.

A continuación, podemos ver cuatro sencillos ejemplos:

En el siguiente ejemplo, encendemos un LED conectado a Arduino cuando pulsamos un sensor táctil de LEGO.

```

//*****
//***** Programa *****
//*****

task main()
{
    const tSensors bumper = (tSensors) S2;
    while(true)
    {
        if (SensorValue(bumper) == 1)
        {
            set_LED(DIRECCION_ARDUINO, 3, true); //LED
conectado a Arduino en el pin 3
        }
        if (SensorValue(bumper) == 0)
        {
            set_LED(DIRECCION_ARDUINO, 3, false);
        }
    }
}
}

```

El siguiente ejemplo encendemos un motor y un LED conectado a Arduino, en función de un interruptor también conectado a Arduino.

```

//*****
//***** Programa *****
//*****

task main()
{

```

```

while(true)
{
    interruptor = leer_sensor( DIRECCION_ARDUINO, 0, false)
// configuramos un sensor conectado al pin A0 y digital (false)
    if (interruptor == 1)
    {
        set_LED(DIRECCION_ARDUINO, 3, true); //LED
conectado a Arduino en el pin 3
        set_motor_cc(DIRECCION_ARDUINO, 8,9,10,100);//
Moror dc conectado en los pines 8 (input 1), 9 (input 2), 10 (enable)
con velocidad 100
    }
    if (SensorValue(bumper) == 0)
    {
        set_LED(DIRECCION_ARDUINO, 3, false);
        set_motor_cc(DIRECCION_ARDUINO, 8,9,10,0);
    }
}
}

```

En el último ejemplo, si el sensor infrarrojo conectado a Arduino obtiene un valor superior a 50, se activa un servomotor conectado a Arduino girando 45°, si el sensor infrarrojo sigue obteniendo un valor de 50, girará otros 45° hasta que el valor obtenido por el sensor sea inferior a 50. En este caso se activarán dos motores de LEGO mientras el valor del sensor sea inferior a 50.

```

//*****
//***** Programa *****
//*****

task main()
{
    infrarrojo = leer_sensor( DIRECCION_ARDUINO, 0, true);
    motor[puerto_motor]= potencia;
    while(true)
    {

        if(infrarrojo < 50)
        {
            motor[motorA]=60;
            motor[motorB]=60;
        }

        while (infrarrojo >= 50)
        {
            set_servo_motor(DIRECCION_ARDUINO,11,45);//
Servomotor conectado al pin 11, y gira un angulo de 45
            motor[motorA]=0;
            motor[motorB]=0;
        }
    }
}

```

3.4. Otros lenguajes de programación.

En el presente proyecto he elegido el lenguaje de programación típico de LEGO MINDSTORMS, así como un lenguaje de programación más técnico como RobotC para proyectos más complicados a la hora de programar nuestro EV3.

No obstante, hay diversas opciones a la hora de programar nuestro bloque EV3, en la página dada en las referencias (Tabla comparativa lenguajes de programación para EV3, s.f.) podemos acceder a una tabla que muestra una comparativa de todos los lenguajes de programación que pueden ser usados para programar nuestro LEGO EV3. Podemos encontrar información relacionada como que lenguaje soporta bluetooth, Wi-Fi, el Protocolo I²C, Windows, Mac, etc.

Dicha tabla es interesante ya que a la hora de abarcar un proyecto puede darnos una idea de que lenguaje de programación debemos usar.

Por ejemplo, a la hora de abordar el presente proyecto, intenté llevarlo a cabo mediante LeJos un firmware que se instala en el bloque EV3 mediante una tarjeta SD, se programa mediante JAVA y podemos programar cómodamente nuestro bloque EV3 para grandes proyectos. No obstante, desistí en esta idea a la hora de comunicarlo con LEGO ya que el proceso era largo y tedioso y requería un nivel de programación elevado.

3.5. Programación de Arduino.

3.5.1. Consideraciones previas.

En este punto procederemos a explicar la programación de Arduino. Lo único que debemos de tener en cuenta a la hora de programar Arduino son los sensores analógicos cuyo valor de lectura no esté comprendido entre 0 y 124 si utilizamos la programación mediante LEGO Mindstorms o de 0 a 255 si utilizamos la programación mediante RobotC. Vamos a suponer que usamos la programación mediante LEGO Mindstorms, como comenté anteriormente, en un byte solo podemos enviar un valor de 0 a 124 por tanto, debemos hacer el cambio de escala en el Arduino.

Un ejemplo típico es el sensor de temperatura LM35, este sensor produce una tensión de 10.0 mV/°C, para conectarse al Arduino tenemos que asegurar que la tensión de salida del circuito electrónico que usa el LM35 este ajustada a el rango de entrada del circuito de medida (esto es, entrada al Arduino) que como sabemos es de 0 a 5 Vcc. Además de esa calibración de ganancia y offset del circuito electrónico se debe realizar una calibración por software del circuito de medida (el Arduino) para ajustar señal medida a la escala o unidades de ingeniería correspondientes.

Por ejemplo, si el sensor va a medir temperaturas entre 0 y 50°C, el circuito electrónico debe ajustarse para que con cero grados tenga una medición de cero voltios y con cincuenta grados produzca una tensión de cinco voltios. En el Arduino se programa para que la medición del convertidor analógico a digital coincida con la lectura real en unidades de ingeniería, por ejemplo, utilizando la instrucción “*map*” de Arduino:

```
sensorValue = map(sensorValue, sensorMin, sensorMax, unidadesMin, unidadesMax);
```

```
/* Mapea un valor analógico entre Cero y 50 */  
void setup() {} // ____  
void loop()  
{  
int val = analogRead(0);
```

```
val_UI = map(val, 0, 1023, 0, 50);  
}
```

Podemos ver otro ejemplo en el que el sensor va a medir temperaturas entre -50 °C y 50°C, el circuito electrónico debe ajustarse para que con menos cincuenta grados tenga una medición de cero voltios y con cincuenta grados produzca una tensión de cinco voltios. En el Arduino se programa para que la medición del convertidor analógico a digital coincida con la lectura real en unidades de ingeniería, utilizando el mismo programa del ejemplo anterior, tendríamos lo siguiente:

```
/* Mapea un valor analógico entre -50 y +50 */  
void setup()  
{  
  int val = analogRead(0);  
  val_UI = map(val, 0, 1023, -50, 50); }  
}
```

Estos cambios de escala son comúnmente utilizados en instrumentación electrónica.

3.5.2. Código de Arduino.

Como comenté anteriormente, el maestro envía mensajes y el esclavo los procesa y responde si es necesario. Cuando se recibe un mensaje, el esclavo decodifica sus bytes para saber qué acciones le está comandando el maestro. Para ello, hemos almacenado todos los posibles valores en una matriz (matriz de instrucciones) y luego procesamos cada instrucción.

Cuando se recibe un mensaje I²C, el esclavo llama a la función que gestiona el evento (onReceive). Entonces, si el mensaje requiere una respuesta, llama a otra función que se encarga de responder al maestro (onRequest).

A continuación, vamos a explicar el código de Arduino, finalmente podremos ver el código completo. No obstante, en referencias podemos encontrar un enlace con un archivo comprimido con este código, y todo lo referente a este proyecto.

El código de Arduino es independiente de la programación utilizada (LEGO Mindstorms o RobotC). En primer lugar, declaramos las librerías correspondientes ya comentadas anteriormente en el presente proyecto. También declaramos la matriz de instrucciones, en esta matriz almacenaremos todos los datos enviados de LEGO a Arduino

```
#include<Wire.h>  
#include <Servo.h>  
  
int instruction[5] = {5,0,0,0,0};
```

A continuación, explicaremos cómo funciona la matriz de instrucción. Como he comentado anteriormente en dicha matriz almacenaremos los datos enviados de LEGO a Arduino, la matriz es consecutiva por lo que el sin contar el primer byte de la dirección:

- instruction[0] = byte 1
- instruction[1] = byte 2
- instruction[2] = byte 3
- instruction[3] = byte 4
- instruction[4] = byte 5
- instruction[5] = byte 6
- instruction[6] = byte 7
- instruction[7] = byte 8

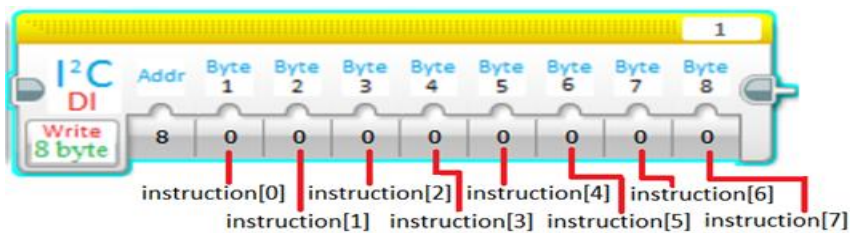


Figura 76: Matriz de instrucciones Arduino mediante programación LEGO.

De este modo la instrucción[0] podrá tener el valor de 1,2,3,4 dependiendo del tipo de dispositivo que tengamos conectado (LED, Motor DC, Servomotor o sensor), las demás dependerán del mismo.

En cuanto a RobotC, el uso de la matriz de instrucciones también lo utilizemos para poder comunicarnos con Arduino. En la programación de Arduino creamos otra matriz de instrucciones para guardar los datos enviados y recibidos, de la siguiente forma:

- instruction[0] = tam_mensaje
- instruction[1] = ard_direccion
- instruction[2] = byte 0
- instruction[3] = byte 1
- instruction[4] = byte 2
- instruction[5] = byte 3
- instruction[6] = byte 4

En segundo lugar, declaráramos todos los sensores con los que vamos a trabajar con sus respectivos cambios de escala (sí es necesario) para que el valor captado por estos sensores este esté comprendido en la escala [0,124] (LEGO) o [0,255] RobotC. Esto será lo único que tenemos que modificar en el código de Arduino. La forma más ordenada es declarar funciones para cada sensor que vayamos a necesitar:

```
float LM35()
{
  int temp_pin = A0;
  if (instruction[1] != 0) temp_pin += instruction[1];
  int val = analogRead(temp_pin);
  val_UI = map(val, 0, 124, 0, 50); // Programando mediante LEGO [124,0]
  Serial.println("temperatura");
}
```

```

    return i;
}

float infrarrojo()
{
    int temp_pin = A0;
    if (instruction[1] != 0) temp_pin += instruction[1];
    int i = analogRead(temp_pin);
    val_UI = map(val, 0, 255, 0, 50); // Programando mediante RobotC [255,0]
    Serial.println("infrarrojo");
    return i;
}

```

Ahora declaramos el objeto Servo, ya hemos hablado de él anteriormente. También declaramos la variable de tipo "int val_sensor". Esta variable será el valor del sensor y es la que enviaremos a LEGO cuando el sensor recoge información. En la función "void setup ()" llamamos a las funciones encargadas de enviar y recibir información entre Arduino y LEGO. Por último, establecemos la dirección del esclavo.

```

Servo temp_servo;
int val_sensor = 0;

void setup()
{
    Wire.begin(0x04); // Dirección del esclavo
    Wire.onRequest(requestEvent); // Enviar información a NXT/EV3
    Wire.onReceive(receiveI2C); // Recibir información de NXT/EV3

    Serial.begin(9600);
}

```

En primer lugar, declaramos dos variables read_byte y byte_count utilizadas para saber cuándo estamos recibiendo datos de LEGO. Posteriormente declaramos la función encargada de recibir datos de LEGO, básicamente si la conexión entre Arduino y LEGO está disponible lee los bytes entrantes continuamente y en caso de que hubiera bytes los almacena, para posteriormente leerlos.

```

byte read_byte = 0x00;
int byte_count = 0;

//Cuando recibimos un dato del NXT/EV3
void receiveI2C(int bytesIn)
{
    read_byte = bytesIn;
    byte_count = 0;
    while(1 < Wire.available())

```



```

{
  read_byte = Wire.read(); //El byte entrante será el registro que utilizaremos

  instruction[byte_count] = read_byte;

  byte_count++;
}
int x = Wire.read(); //leemos byte entrante

```

A partir de aquí vamos a dividir el código según el byte 2 (aún estamos dentro de la función recibir datos), el cual nos daba información sobre el código del dispositivo (LED=1, Servo=2, Motor=3, Sensor=4). Dicho byte está almacenado en la matriz de instrucciones.

El código del control de un LED es bastante sencillo y lo podemos ver a continuación:

```

if( instruction[0] == 1 ) // si el byte 2 es igual a 1 es porque lo hemos configurado
como LED
{
  Serial.println(" LED ");

  Serial.print("PIN: ");
  Serial.println(instruction[1]);
  pinMode(instruction[1], OUTPUT);

  Serial.print("ESTADO: ");
  if(instruction[2] == 0)
  {
    Serial.print("Apagado");
    digitalWrite(instruction[1], LOW);
  }
  else
  {
    Serial.println("Encendido");
    digitalWrite(instruction[1], HIGH);
  }
}
}

```

Del mismo modo, el código del servomotor:

```

else if( instruction[0] == 2 )
{
  Serial.println(" Servo Motor ");

  Serial.print("PIN: ");
  Serial.println(instruction[1]);
  temp_servo.attach(instruction[1]);
}

```

```

    instruction[2] = instruction[2]*2 + instruction[3]; // quitar esta línea si
programamos mediante RobotC

    Serial.print("Angulo: ");
    Serial.println(instruction[2]);
    temp_servo.write(instruction[2]);
}

```

En el código del motor multiplicamos la velocidad por 2,55, ya que la escala en Arduino es de [0,255] y en LEGO Mindstorms la escala es de [0,100] (en el caso de RobotC, no habría que poner esto, ya que la escala coincide con la escala de Arduino), de este modo si insertamos en LEGO Mindstorms una velocidad de 100 Arduino la multiplicará por 2,55 cuyo resultado será 255 y se corresponde con el máximo valor que podemos introducir a un motor insertado a Arduino.

```

else if( instruction[0] == 3 )
{
    Serial.println(" Motor DC ");

    Serial.print("PIN_1: ");
    Serial.print(instruction[1]);
    pinMode(instruction[1], OUTPUT);
    digitalWrite(instruction[1], HIGH);

    Serial.print("PIN_2: ");
    Serial.print(instruction[2]);
    pinMode(instruction[2], OUTPUT);
    digitalWrite(instruction[2], LOW);

    Serial.print("PIN_3: ");
    Serial.print(instruction[3]);
    pinMode(instruction[3], OUTPUT);

    Serial.print(" Velocidad: ");
    analogWrite(instruction[3], instruction[4]*2.55);
    Serial.print(instruction[4]*2.55);

}

```

Por último, el sensor, en su código podemos observar que el byte 3 nos indica si el sensor es analógico o digital en caso de ser analógico llama a la función infrarrojo o LM35 y almacena su valor en val_senior. Si es digital declara el pin establecido en el software LEGO Mindstorms y almacena su estado en val_sensor.

```

else if( instruction[0] == 4 )

```



```

int instruction[5] = {5,0,0,0,0};

///
/// Este código hace referencia al byte 2 que insertemos en el
programa LEGO MINDSTORMS
///
/// instruction[0] = 1 (LED), 2 (servo motor} ,3 (DC motor), 4
(sensor)
///
/// instruction [0] = 1 ==> instruction [1] puerto (PIN digital)
/// instruction [2] 0 (LED apagado) o 1 (LED
encendido)
///
/// instruction [0] = 2 ==> instruction [1] puerto (numero
identificación Servo Motor)
/// instruction [2] ángulo
///
/// instruction [0] = 3 ==> instruction [1] número PIN1
/// instruction [2] velocidad1 (0-99)
/// instruction [3] numero PIN2
/// instruction [4] velocidad2 (0-99)
///
/// instruction [0] = 4 ==> instruction [1] puerto
/// instruction [2] 0 (PIN analógico) o 1
(PIN digital)
///
///
///
///
/// Aquí declaramos todos los sensores analógicos que tengamos.
///

float LM35()
{
  int temp_pin = A0;
  if (instruction[1] !=0 ) temp_pin += instruction[1];

  float t=analogRead(temp_pin);
  t=t*5.0/1024;
  t*=100;
  Serial.println("Temperature");
  return t;
}

float infrarrojo()
{
  int temp_pin = A0;
  if (instruction[1] !=0 ) temp_pin += instruction[1];
  float i=analogRead(temp_pin);
  float j;
  j = map(i, 0, 150, 0, 100);

  Serial.println("infrarrojo");
  return j;
}
///

Servo temp_servo;
int val_sensor = 0;

```

```

void setup()
{
  Wire.begin(0x04); // Dirección del esclavo
  Wire.onRequest(requestEvent); // Enviar información a NXT/EV3
  Wire.onReceive(receiveI2C); // Recibir información de NXT/EV3

  Serial.begin(9600);
}

void loop()
{
  delay(500);
}

///

---



byte read_byte = 0x00;
int byte_count = 0;

//Cuando recibimos un dato del NXT/EV3
void receiveI2C(int bytesIn)
{
  read_byte = bytesIn;
  byte_count = 0;
  while(1 < Wire.available())
  {
    read_byte = Wire.read(); //El byte entrante será el registro que
    utilizaremos

    instruction[byte_count] = read_byte;

    byte_count++;
  }
  int x = Wire.read(); //leemos byte entrante

  if( instruction[0] == 1 ) // si el byte 2 es igual a 1 es porque
  lo hemos configurado como LED
  {
    Serial.println(" LED ");

    Serial.print("PIN: ");
    Serial.println(instruction[1]);
    pinMode(instruction[1], OUTPUT);

    Serial.print("ESTADO: ");
    if(instruction[2] == 0)
    {
      Serial.print("Apagado");
      digitalWrite(instruction[1], LOW);
    }
    else
    {
      Serial.println("Encendido");
      digitalWrite(instruction[1], HIGH);
    }
  }
}

```

```

else if( instruction[0] == 2 )
{
  Serial.println(" Servo Motor ");

  Serial.print("PIN: ");
  Serial.println(instruction[1]);
  temp_servo.attach(instruction[1]);

  instruction[2] = instruction[2]*2 + instruction[3];

  Serial.print("Angulo: ");
  Serial.println(instruction[2]);
  temp_servo.write(instruction[2]);

}
else if( instruction[0] == 3 )
{
  Serial.println(" Motor DC ");

  Serial.print("PIN_1: ");
  Serial.print(instruction[1]);
  pinMode(instruction[1], OUTPUT);
  digitalWrite(instruction[1], HIGH);

  Serial.print("PIN_2: ");
  Serial.print(instruction[2]);
  pinMode(instruction[2], OUTPUT);
  digitalWrite(instruction[2], LOW);

  Serial.print("PIN_3: ");
  Serial.print(instruction[3]);
  pinMode(instruction[3], OUTPUT);

  Serial.print(" Velocidad: ");
  analogWrite(instruction[3], instruction[4]*2.55);
  Serial.print(instruction[4]*2.55);

}
else if( instruction[0] == 4 )
{
  Serial.println(" Sensor ");

  Serial.print(": ");
  Serial.print(instruction[1]);

  if ( instruction[2] == true )
  {
    Serial.println("Sensor analogico");
    val_sensor = infrarrojo();
  }
  else
  {
    Serial.println("Sesnor digital");
    pinMode(instruction[1], INPUT);
    val_sensor = digitalRead(instruction[1]);
  }

}

}

} // Final

```

```
//
void requestEvent()
{
  if (instruction[0] == 4)
  {
    Wire.write(val_sensor); // Envía el da to a LEGO
    Serial.print("Value: ");
    Serial.println(val_sensor);
  }
}
}
```

3.5.3. Uso del serial en Arduino.

Todos los datos recibidos y enviados a LEGO serán registrados por Arduino. Para saber si Arduino y LEGO se están comunicando correctamente es buena táctica, abrir el serial y ver los datos que se están enviando. Si hacemos esto, es importante conectar a Arduino solo los cables que se encargan del Protocolo I²C es decir, amarillo (A5) y azul (A4). Ya que si también conectamos los cables verde y rojo encargados de alimentar al Arduino y por otra parte lo alimentamos por el ordenador podemos romper nuestro Arduino.

```
COM6 (Arduino/Genuino Uno)
: 0Sensor analogico
infrarrojo
Value: 11
Value: 11
Motor DC
PIN_1: 9PIN_2: 8PIN_3: 10 Velocidad: 255.00 Motor DC
PIN_1: 9PIN_2: 8PIN_3: 10 Velocidad: 0.00 Sensor
: 3Sesnor digital
Value: 1
Value: 1
Sensor
: 0Sensor analogico
infrarrojo
Value: 11
Value: 11
Motor DC
PIN_1: 9PIN_2: 8PIN_3: 10 Velocidad: 255.00 Motor DC
PIN_1: 9PIN_2: 8PIN_3: 10 Velocidad: 0.00 Sensor
```

Figura 77: Ejemplo Comunicación serial.

4. Aplicaciones.

4.1. Aumentar las entradas y salidas de LEGO EV3 con sensores y actuadores de LEGO Mindstorms y Arduino.

LEGO Mindstorms dispone de un número limitado de sensores y actuadores. Con el presente proyecto hemos aprendido a compatibilizar todo tipo de sensores electrónicos y actuadores a LEGO EV3. Esto también es aplicable para los sensores ya distribuidos por LEGO, permitiendo aumentar las entradas y salidas de nuestro bloque EV3.

La dificultad que presenta comunicar los sensores y actuadores propios de LEGO, es conocer la comunicación que estos realizan con el propio LEGO, ya que nosotros en vez de conectar estos dispositivos a LEGO los conectaremos al Arduino. Como hemos comentado anteriormente, estos dispositivos se comunican mediante un cable del tipo RJ12 compuesto a su vez por 6 cables. Por lo que para lograr la comunicación de estos dispositivos debemos de conocer la función que realizará cada cable.

Para lograr esto debemos de ir a la página web del fabricante y descargar la documentación relacionada con el hardware de dichos sensores.

La comunicación que realiza LEGO EV3 es diferente a como se comunica el LEGO NXT con sus sensores y actuadores. LEGO NXT realiza una comunicación analógica estándar como cualquier sensor analógico, respecto a la comunicación del bloque EV3 hablaremos de ella más adelante.

Como he comentado anteriormente si accedemos a la página web del fabricante podemos encontrar la documentación que nos proporcionará la información necesaria para caracterizar los sensores y actuadores del bloque NXT para poder conectarlos a Arduino. Aunque sean del bloque NXT, nuestro bloque EV3 podrá operar con ellos sin problemas.

4.1.1. Sensor táctil NXT.

El esquema del circuito que compone el sensor táctil es el siguiente:

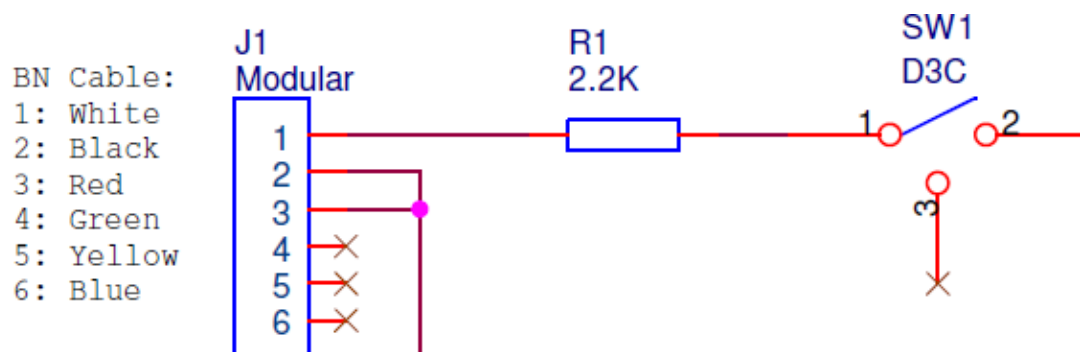


Figura 78: Esquema electrónico Sensor táctil.

La caracterización de los cables es la siguiente:

PIN	NOMBRE	COLOR	FUNCIÓN
1	ANA		Vcc (+5V)
2	GND		Input 2
3	GND		Input 3
4	IPOWERA		-
5	DIGIAI0		-
6	DIGIAI1		-

Tabla 4: Caracterización Sensor táctil.

Como podemos observar, las entradas “Input 2” y “Input 3” permitirán la activación o la desactivación del dispositivo que tengamos conectado, mientras que “Input 1” estaría conectado a Vcc. Por lo tanto, la conexión del sensor táctil del bloque NXT lo podemos realizar como un interruptor normal, explicado en el Anexo 1 del presente proyecto.

4.1.2. Sensor de sonido NXT.

El esquema principal del circuito que compone el sensor de sonido es el siguiente:

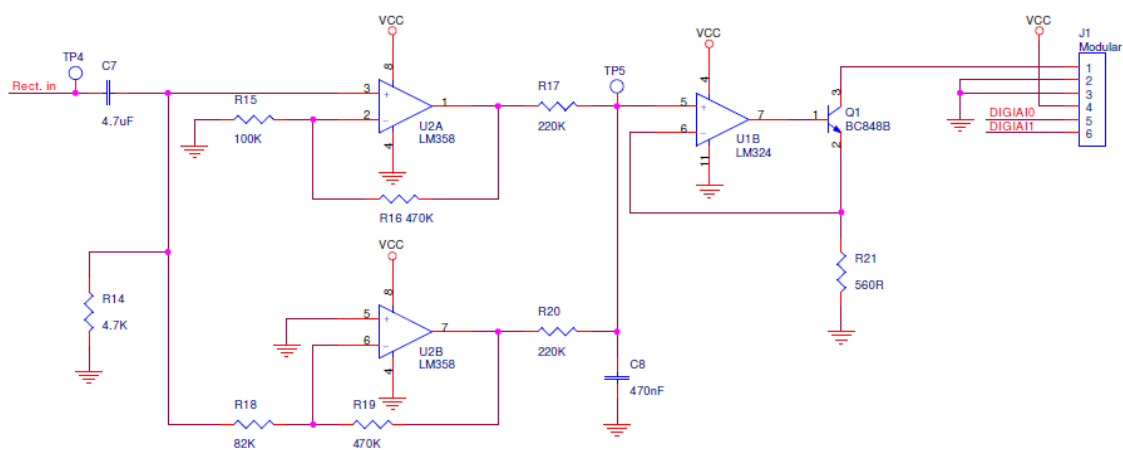


Figura 79: Esquema electrónico Sensor de sonido.

La caracterización de los cables es la siguiente:

PIN	NOMBRE	COLOR	FUNCIÓN
1	ANA		Output 1
2	GND		GND
3	GND		GND
4	IPOWERA		Vcc (+5V)
5	DIGIAI0		-
6	DIGIAI1		-

Tabla 5: Tabla caracterización Sensor de sonido.

Mediante la salida 1 podemos leer la intensidad del sonido, si aumentamos la intensidad del volumen aumentará el voltaje. Sin embargo, a la hora de conectarlo al Arduino, el campo de detección estará limitado y solo podremos diferenciar entre una intensidad sonora elevada y baja. Podemos variar este umbral sonoro conectando la salida 1 a un potenciómetro de un mega ohmio, el voltaje variaría entre 0 y 5V y nos permitirá ajustar

la sensibilidad del sensor, pero como he comentado anteriormente, solo nos permitirá diferenciar dos estados.

Si queremos obtener una salida numérica real proporcional a la intensidad sonora debemos utilizar un convertidor analógico digital.

4.1.3. Sensor de luminosidad NXT.

El esquema principal del circuito que compone el sensor de luminosidad es el siguiente:

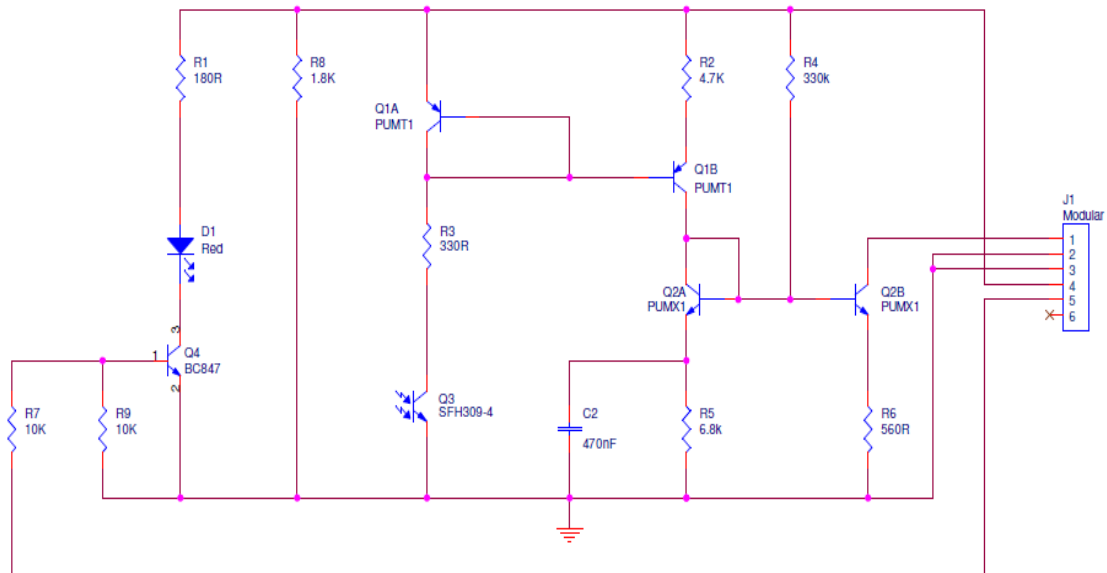


Figura 80: Esquema electrónico Sensor de luminosidad.

La caracterización de los cables es la siguiente:

PIN	NOMBRE	COLOR	FUNCIÓN
1	ANA		Output 1 analógico
2	GND		GND
3	GND		GND
4	IPOWERA		Vcc (+5V)
5	DIGIAIO		Output 2 digital
6	DIGIAI1		-

Tabla 6: Caracterización del Sensor de luminosidad.

El sensor de luz funciona de igual forma que el sensor de sonido, si aumentamos la intensidad luminosa, aumentará el voltaje. Para el sensor de luz podemos operar de dos formas, mediante la salida analógica utilizando la salida "Output 1", la cual nos dará un número proporcional a la intensidad luminosa sin necesidad de utilizar un convertidor analógico digital, o podemos utilizar la salida digital "Output 1" de la misma forma que el sensor de sonido.

4.1.4. Sensor ultrasónico NXT.

El esquema principal del circuito que compone el sensor ultrasónico es el siguiente:

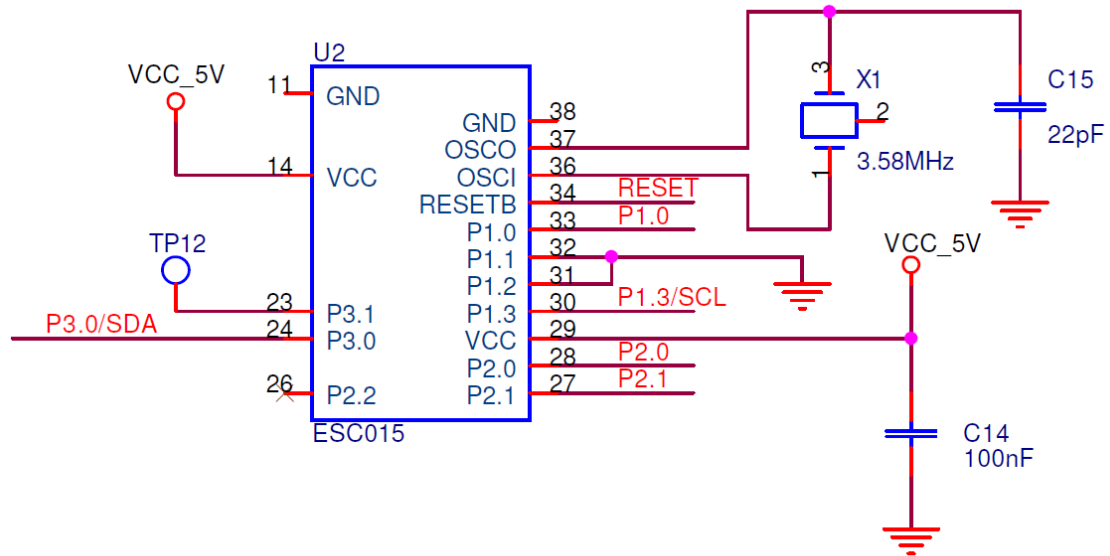


Figura 81: Esquema electrónico principal del Sensor ultrasónico.

La caracterización de los cables es la siguiente:

PIN	NOMBRE	COLOR	FUNCIÓN
1	ANA		Vcc (+9V)
2	GND		GND
3	GND		GND
4	IPOWERA		Vcc (+5V)
5	DIGIAIO		SCL
6	DIGIAI1		SDA

Tabla 7: Caracterización del Sensor ultrasónico.

El sensor ultrasónico, en principio no lo podemos utilizar debido a que este sensor se comunica con el maestro (cualquiera que sea, LEGO o Arduino) mediante I²C. La razón es que Arduino solo presenta dos pines los cuales gestionan la comunicación I2C (A4 y A5) y estos pines son los que utilizamos para la comunicación entre LEGO y Arduino, por lo que, en el caso de querer conectar este sensor, necesitaríamos otros dos pines más de Arduino, encargados de gestionar el Protocolo I2C.

4.1.5. Sensores y actuadores de LEGO EV3.

Por el mismo motivo que el sensor ultrasónico del bloque NXT, todos los sensores del bloque EV3, llevan un microcontrolador cuyo objetivo principal es gestionar la comunicación entre el bloque EV3 y estos sensores. Esta comunicación no se puede hacer de forma analógica como en el caso del sensor de luz o en el sensor de sonido del bloque NXT y requiere un conocimiento más profundo del microcontrolador que utilizan estos sensores, del Protocolo I2C y de programación. Sin contar que necesitaríamos más pines de Arduino encargados de gestionar el Protocolo I2C.

4.1.6. Motores.

Los actuadores de LEGO son dos motores, uno grande y uno mediano. Ambos motores son de 9 V, pero podemos alimentarlos hasta 12 V, por lo tanto, necesitaremos una

alimentación externa y el uso del puente en H. Mediante este procedimiento podemos conectar cualquier motor de LEGO tanto del bloque EV3 como del bloque NXT.

Para poder operar con los motores LEGO necesitaremos prestar atención a los cables negro y blanco del cable de comunicación RJ12 de LEGO. Estos dos cables son los encargados de alimentar y controlar el motor. El método para realizar esta conexión lo podemos encontrar en el Anexo I.

Por lo tanto, los cables para operar con el motor son:

PIN	NOMBRE	COLOR	FUNCIÓN
1	ANA		Input 1
2	GND		Input 2
3	GND		-
4	IPOWERA		-
5	DIGIAI0		-
6	DIGIAI1		-

Tabla 8: Caracterización de los motores de LEGO.

Ambos motores quedan caracterizados en las tablas siguientes:

	Torque	Velocidad de rotación	Intensidad	Fuerza mecánica	Potencia	Rendimiento
4.5V	6.64 N.cm	24 rpm	0.35 A	0.17 W	1.57 W	10%
6 V	6.64 N.cm	72 rpm	0.35 A	0.50 W	2.10 W	24%
7.5 V	6.64 N.cm	120 rpm	0.35 A	0.83 W	2.62 W	32%
9 V	6.64 N.cm	165 rpm	0.35 A	1.15 W	3.33 W	34%
10.5 V	6.64 N.cm	207 rpm	0.35 A	1.44 W	3.88 W	37 %
12 V	6.64 N.cm	249 rpm	0.35 A	1.73 W	4.44 W	39 %

	Torque	Velocidad de rotación	Intensidad	Fuerza mecánica	Potencia	Rendimiento
4.5V	17.3 N.cm	24 rpm	0.69 A	0.43 W	3.10 W	10%
6 V	17.3 N.cm	51 rpm	0.69 A	0.92 W	4.14 W	24%

7.5 V	17.3 N.cm	78 rpm	0.69 A	1.41 W	5.17 W	32%
9 V	17.3 N.cm	105 rpm	0.69 A	1.90 W	6.21 W	34%
10.5 V	17.3 N.cm	132 rpm	0.69 A	2.39 W	7.54 W	37 %
12 V	17.3 N.cm	153 rpm	0.69 A	2.77 W	8.28 W	39 %

4.2. Maqueta utilizando sensores y actuadores de bajo coste.

Uno de los objetivos principales de este proyecto era construir un sensor y un actuador electrónico y que LEGO Mindstorms sea capaz de trabajar con él. En este punto demostraremos que hemos alcanzado este objetivo, para ello hemos construido una maqueta cuya función es clasificar los vasos atendiendo al color de los mismos.

Para ello nos hemos servido de un diodo emisor de infrarrojo que actuará como un sensor y un motor de 4,5V reciclado que actuará como un actuador.

Como los vasos presentan diferentes colores, la luz captada por el sensor infrarrojo será distinta. De esta forma el sensor infrarrojo presenta una escala de 0 si no recibe ninguna luz infrarroja hasta 100 que sería la máxima luz infrarroja que puede captar. Haciendo varias pruebas calibrando el sensor, nos damos cuenta de que:

- El estado de reposo del sensor es de 50.
- Si el vaso es transparente el sensor no presenta ningún cambio en su lectura.
- Si el vaso es negro, el sensor disminuye en valores inferiores a 20.
- Si el vaso es de color blanco, el sensor disminuye hasta valores de 30.

Por lo tanto, tenemos un rango para poder clasificar los vasos atendiendo al valor del sensor. De esta forma, si el valor del sensor es menor que 20 el vaso es negro, si el vaso es de color blanco el valor está entre 50 y 30, y por último si no obtenemos ningún cambio el vaso es transparente.

El motor girará hacia la izquierda o derecha, dependiendo de las lecturas del sensor, y actuará sobre una rueda dentada que desplazará a una pala construida con piezas LEGO que retirará el vaso con pequeño desplazamiento.

La maqueta diseñada en el departamento de automática presenta el siguiente aspecto

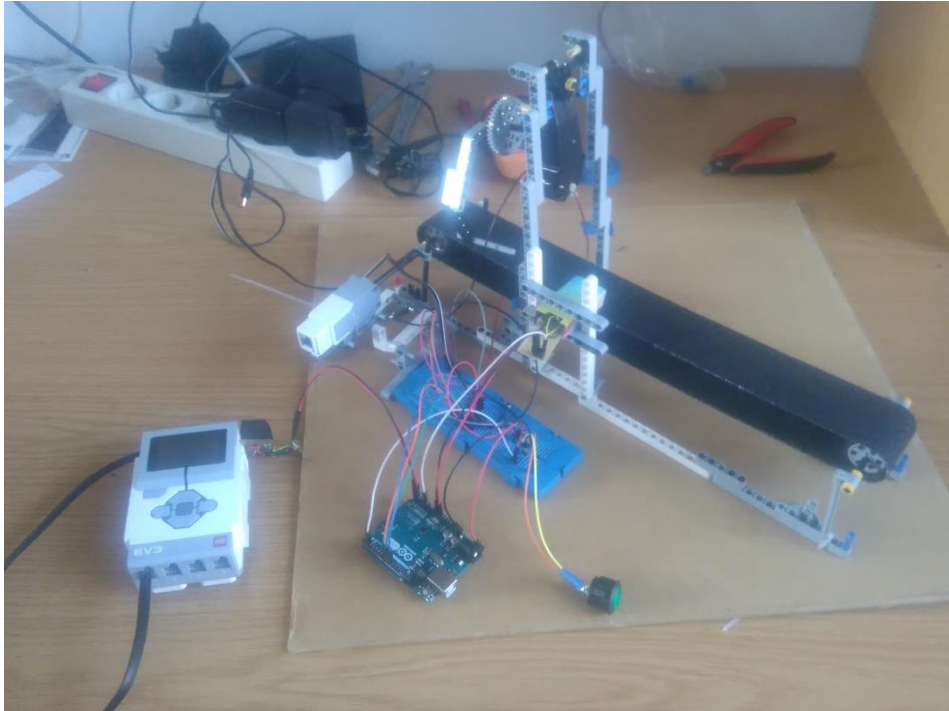
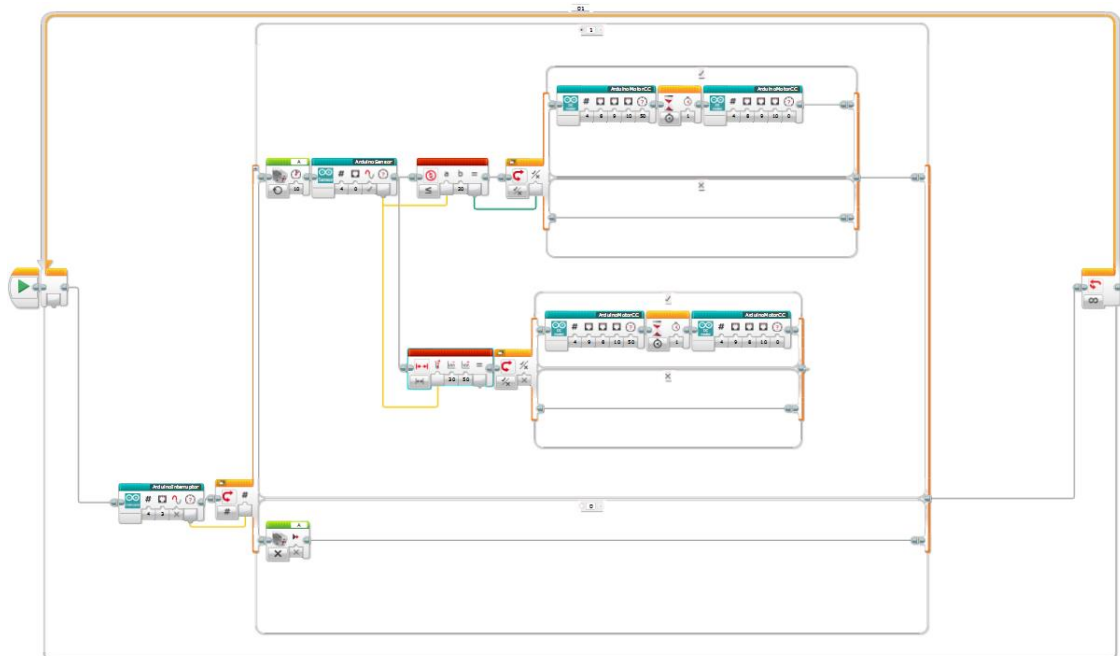


Figura 82: Sistema de automatización basado en sensores y actuadores de bajo coste.

Para conectar el motor hemos utilizado el encapsulado L293D explicado en el ANEXO I.

La programación de LEGO la podemos ver en la siguiente imagen, al igual que el equivalente a la programación RobotC.



En el enlace que podemos encontrar en referencias podemos descargar, el programa para una mejor visualización.

A continuación, mostraremos el código equivalente con RobotC. Recordemos que el código de RobotC se divide entre partes: Comunicación I2C, Dispositivos y programa. Solo pondremos la parte referida al programa, ya que las otras dos partes no varían.

```
//*****
//***** Programa *****
//*****
task main()
{
    interruptor = leer_sensor( DIRECCION_ARDUINO, 1, false);
//interruptor conectado a Arduino.
    infrarrojo = leer_sensor( DIRECCION_ARDUINO, 0, true);
//infrarrojo conectado a Arduino.
    motor[puerto_motor]= potencia;//motor de LEGO EV3 mediano.
    while(true)
    {

        if(interruptor == 1)// si pulsamos en el interruptor
        {
            motor[motorA]=10; //se enciende el motor a una velocidad
reducida para que el sensor pueda percibir los colores.

            if(infrarrojo>5 && infrarrojo<10) //si la respuesta del
sensor esta entre 5 y 10 es un defecto tipo 1.
            {
                set_motor_dc(DIRECCION_ARDUINO, 8,9,10,100); //Pin
1:8, Pin 2:9 giro a derechas.
            }
            else if(infrarrojo>10) //si la respuesta del sensor eesta
por encima de 10 es un defecto tipo 2.
            {
                set_motor_dc(DIRECCION_ARDUINO, 9,8,10,100); //Pin
1:9, Pin 2:8 giro a izquierdas.
            }

        }

    }
}
```

5. Conclusión.

Este proyecto ha sido un gran reto para mí, sobre todo al principio, que al desconocer el Protocolo I²C y utilizar la opción de LeJos, comentado anteriormente, hizo que tuviera que aprender programación en JAVA y el concepto de dicho protocolo.

Tras varias semanas sin conseguir avanzar significativamente decidí cambiar de lenguaje de programación y utilizar el propio de LEGO Mindstorms. Al plantearme esta opción se abrió un abanico de posibilidades y pude lograr que la comunicación sea satisfactoria. Un aspecto importante que tuve con esta programación son los bloques de tiempo (de 0,05 segundos) necesarios para que LEGO le dé tiempo a recibir y enviar datos a Arduino, ya que si no insertamos estos bloques la comunicación no funcionaría.

También decidí utilizar un lenguaje de programación más técnico como RobotC, para proyectos más complejos, al realizar la programación mediante esta plataforma, tuvo la complejidad de que RobotC utilizaba una variante del Protocolo I²C de 7 bytes ya que en un principio desconocía que pudiera haber variantes de dicho protocolo lo que me llevo a investigar y comprender mejor este protocolo para poder utilizarlo. La comprensión y realización del código en RobotC fue crucial los conceptos adquiridos en la asignatura de Informática Aplicada impartida en mi titulación.

Respecto a la comunicación cableada entre LEGO y Arduino, el primer experimento que realice fue sin las resistencias *pull-up* por lo que no conseguí que ambos dispositivos se comunicasen. Posteriormente, tras investigar pude comprender el concepto de las llamadas resistencias *pull-up* y que función presentan en dicha comunicación, por lo que, al realizar esta comunicación con dichas resistencias, la comunicación se produjo satisfactoriamente.

Gracias a la cantidad de información presente en internet sobre Arduino, su programación no tuvo mayor complicación, sin embargo, había que tener en cuenta que al realizar la programación con LEGO Mindstorms el rango de valores que recogía el sensor era de [0,124] de tal modo que, si el valor del sensor sobrepasaba dicho rango, los valores que recibidos por LEGO no se correspondían con los del sensor. Esto lo pude solucionar con la función "map" de Arduino, la cual escala los valores recogidos por el sensor en la escala que nosotros deseemos, en este caso de [0,124].

Una de las grandes ventajas que presenta la comunicación realizada entre ambos dispositivos es que cualquier sensor o actuador compatible con Arduino puede ser operado por LEGO, lo que implica que LEGO podrá operar una gran cantidad de sensores y actuadores y no solo los que están distribuidos por LEGO. Para los sensores NXT de LEGO tuvimos que caracterizar los cables para saber que función tenían a la hora de conectarlos a Arduino, una vez hecho esto, los sensores podían comunicarse con Arduino sin problemas.

Respecto a los sensores del bloque EV3, la comunicación es más compleja y toda esta comunicación es gestionada por un microcontrolador, entonces, para que Arduino pueda comunicarse con ellos es necesario saber cómo se comunican estos

microcontroladores con LEGO, todo esto conlleva un estudio profundo y no es el objetivo de este proyecto, no obstante, investigué y deje información complementaria para posibles proyectos futuros que podemos ver en el capítulo siguiente.

Si hacemos una búsqueda por internet numerosos proyectos utilizan dos o más bloques de LEGO para realizar proyectos complejos. Con este método hemos conseguido aumentar las entradas y salidas, por lo que con un solo bloque de LEGO podemos controlar un gran número de sensores y actuadores tanto propios como sensores o actuadores electrónicos de bajo coste, ya que además podemos alimentar estos sensores y actuadores por separado para que el bloque de LEGO no sea afectado. Los sensores y actuadores conectados a Arduino y a LEGO son perfectamente compatibles entre ellos y podemos combinarlos como queramos.

Respecto a la conexión de motores con Arduino presenté el problema inicial que Arduino no presenta en los pines de entradas y salidas la intensidad suficiente para poder operar con motores, ya que no están diseñados para alimentar grandes cargas. Esto me llevó a investigar y analizar qué forma era la más apropiada para conectar motores a Arduino, debido a que no solo nos bastaba hacer funcionar el motor sino controlar el sentido de giro y la velocidad del motor. De esta manera llegué a la conclusión que la forma más apropiada era utilizando el encapsulado L293D que es un driver para motores que se fundamenta en un puente H. Por lo tanto, gracias a lo aprendido anteriormente en estos años de universidad en materias como fundamentos de electrónica e instrumentación pude afrontar este problema satisfactoriamente.

Finalmente, uno de los objetivos principales del presente proyecto era diseñar un actuador y un sensor y que LEGO pueda operar con ellos, para demostrar esto, decidí elaborar un sistema de control automatizado con sensores de electrónicos de bajo coste que permita mostrar el funcionamiento y la comunicación entre LEGO y Arduino.

Por lo tanto hemos conseguido un proceso metódico de fácil implantación, comprensión y programación, por lo que el presente proyecto podría aplicarse a la enseñanza, debido a que los alumnos no solo asimilarían conceptos de robótica, programación, instrumentación... que proporciona la plataforma LEGO Mindstorms, sino que además introducimos Arduino un controlador que está muy presente actualmente el cual, resulta básico para cualquier ingeniero y lograría aumentar los conocimientos de los alumnos relacionados con la electrónica y de esta forma ayudar a despertar vocaciones relacionadas con los estudios de ciencias e ingeniería.

6. Trabajos futuros.

La plataforma de LEGO Mindstorms puede dar mucho más de sí ya que presenta un gran abanico de posibilidades y más aún cuando la juntamos con Arduino.

Me gustaría proponer una serie de ampliaciones interesantes para futuros proyectos relacionados con la plataforma LEGO Mindstorms y Arduino.

6.1. Comunicación inalámbrica entre Arduino y LEGO.

Sería interesante lograr una comunicación inalámbrica entre Arduino y LEGO ya sea mediante Wi-Fi o bluetooth ya que nos ahorraría el uso del diseño de una placa o fabricar cables que permitan la comunicación entre ambos dispositivos.

A continuación, podemos acceder una serie de páginas web que podemos encontrar información sobre este tema:

- (Conectar Arduino a LEGO mediante bluetooth, URL 1., 2017)
- (Conectar Arduino a LEGO mediante bluetooth con LeJos, URL 2., 2017)
- (Conectar Arduino a LEGO mediante bluetooth, URL 3, 2017)
- (Conectar Arduino a LEGO mediante bluetooth URL 4, 2017)

6.2. Comunicación de sensores EV3 con Arduino.

Como comentemos anteriormente, tenemos el problema a la hora de conectar sensores EV3 con Arduino, ya que este solo presenta dos pines (A4 y A5) encargados de operar con el protocolo I²C y estos pines ya los estamos utilizando para la comunicación con LEGO.

No obstante, he investigado por internet y existe la posibilidad de emular pines de Arduino para que sean capaces de operar con el Protocolo I²C. Podemos acceder a más información en las siguientes páginas web:

- (Emular pines de Arduino para utilizar el protocolo I2C, 2017)
- (Emular pines de Arduino para utilizar el protocolo I2c (2), s.f.)

Esto en parte, nos solucionaría parte del problema, pero faltaría como programar la comunicación, para que Arduino sea capaz de comunicarse con el sensor. Para ello Arduino debe comunicarse con el microcontrolador del sensor encargado de gestionar la comunicación.

En el caso del sensor ultrasónico del NXT podemos encontrar una pequeña guía de esta comunicación (mediante Protocolo I²C) proporcionada por LEGO, y que podemos descargarla dentro del archivo comprimido de los dispositivos del bloque NXT, junto todos los esquemas de los circuitos de los sensores NXT y EV3:

- (Enlace de descarga de los esquemas electrónicos de los sensores y actuadores del bloque NXT., 2017)
- (Enlace de descarga de los esquemas electrónicos de los sensores y actuadores del bloque EV3, 2017)

Con respecto a los dispositivos del bloque EV3, la comunicación es más compleja y LEGO proporciona una guía aparte de comunicación del bloque EV3.

➤ (Guía de comunicación del bloque EV3, s.f.)

Esta guía se recogen funciones de comunicación como enviar datos o recibir datos. También incluye ejemplos de programación con sensores y actuadores.

7. Referencias

- Industries, D. (2017, 3 24). *Conector LEGO Arduino*. Retrieved from Amazon: https://www.amazon.com/Breadboard-Adapter-for-LEGO-MINDSTORMS/dp/B00A0P5FY6/ref=as_li_ss_tl?ie=UTF8&linkCode=sl1&tag=de xteindus-20&linkId=a520dd335dc128dc45affded10098935
- Industries, D. (24 de 3 de 2017). *Link de descarga bloque I2C*. Obtenido de Github: https://github.com/DexterInd/EV3_Dexter_Industries_Sensors
- LEGO. (26 de 03 de 2017). *Descargar software LEGO MINDSTORMS*. Obtenido de LEGO: <https://www.lego.com/es-es/mindstorms/downloads/download-software>
- Link de descarga del archivo comprimido*. (s.f.). Obtenido de Iván. https://www.dropbox.com/sh/p5o8bmjkgusqh7i/AABvg6Jj_t3ZvAYuPfQXt3Oua?dl=0
- Link de descarga RobotC*. (s.f.). Obtenido de RobotC: <http://www.robotc.net/download/lego/>
- Tabla comparativa lenguajes de programación para EV3*. (s.f.). Obtenido de Teamhassenplug: <http://www.teamhassenplug.org/NXT/NXTSoftware.html>

8. Bibliografía

- Aprende a programar LEGO.* (s.f.). Obtenido de LEGO MINDSTORMS:
<https://www.lego.com/es-es/mindstorms/learn-to-program>
- Conectar Arduino a LEGO mediante bluetooth con LeJos, URL 2.* (4 de 2017). Obtenido de miguelduarte: <http://miguelduarte.pt/2013/07/23/bluetooth-communication-between-arduino-and-lego-nxt-using-lejos/>
- Conectar Arduino a LEGO mediante bluetooth URL 4.* (4 de 2017). Obtenido de makerzone: <http://makerzone.mathworks.com/resources/connect-to-lego-mindstorms-nxt-via-bluetooth/>
- Conectar Arduino a LEGO mediante bluetooth, URL 1.* (4 de 2017). Obtenido de Arduino foro: <https://forum.arduino.cc/index.php?topic=189266.0>
- Conectar Arduino a LEGO mediante bluetooth, URL 3.* (4 de 2017). Obtenido de stackoverflow: <http://stackoverflow.com/questions/22550069/control-lego-nxt-using-arduino-bluetooth>
- Conectar motores LEGO a Arduino.* (s.f.). Obtenido de robotc: http://www.robotc.net/wikiarchive/Tutorials/Arduino_Projects/Mobile_Robotics/Lego/Connecting_A_Lego_Motor
- Conectar sensor luminoso NCT a Arduino.* (s.f.). Obtenido de robotc: http://www.robotc.net/wikiarchive/Tutorials/Arduino_Projects/Mobile_Robotics/Lego/Connecting_an_Active_Light_Sensor
- Conexión de dispositivos básicos a Arduino.* (s.f.). Obtenido de luisllamas: <https://www.luisllamas.es/encender-un-led-con-arduino/>
- Conexión de motores CC a Arduino.* (s.f.). Obtenido de aprendiendoarduino: <https://aprendiendoarduino.wordpress.com/tag/motor-dc/>
- Controlar un motor CC.* (s.f.). Obtenido de iescamp: <http://www.iescamp.es/miarduino/2016/02/21/2-motores-de-cc-velocidad-y-cambio-de-sentido/>
- Curso de programación EV3.* (s.f.). Obtenido de Educagratis: <http://educagratis.cl/moodle/course/view.php?id=619>
- Curso de programación de Arduino.* (s.f.). Obtenido de Luisllamas: <https://www.luisllamas.es/tutoriales-de-arduino/>
- Diferencia entre los diferentes tipos de cables RJ.* (s.f.). Obtenido de Alvaro-blogg: <http://alvaro-blogg.blogspot.com.es/2012/01/que-diferencia-existe-entre-el-conector.html>
- Ejemplo de Protocolo I2C con Arduino.* (s.f.). Obtenido de Prometec: <http://www.prometec.net/bus-i2c/>

Ejemplo I2C con Arduino. (s.f.). Obtenido de Arduino: <https://geekytheory.com/tutorial-arduino-conectar-lcd-16x2-por-protocolo-i2c>

Ejemplos básicos programación RobotC. (s.f.). Obtenido de ugciscrobotica: <https://ugciscrobotica.wordpress.com/2014/12/02/ejemplo-de-manejo-de-sensor-tactil-en-robotc/>

Ejemplos de programación RobotC. (s.f.). Obtenido de electricbricks: <http://blog.electricbricks.com/2010/03/tutorial-robotc-lego-mindstorms-nxt-2/>

Emular pines de Arduino para utilizar el protocolo I2c (2). (s.f.). Obtenido de todobot: <http://todobot.com/blog/2010/09/25/softi2cmaster-add-i2c-to-any-arduino-pins/>

Emular pines de Arduino para utilizar el protocolo I2C. (4 de 2017). Obtenido de jeelabs: <http://jeelabs.org/2010/03/15/software-and-hardware-i2c/>

Emular pines de Arduino para utilizar el protocolo I2C. (4 de 2017). Obtenido de todobot: <http://todobot.com/blog/2010/09/25/softi2cmaster-add-i2c-to-any-arduino-pins/>

Enlace de descarga de los esquemas electrónicos de los sensores y actuadores del bloque EV3. (4 de 2017). Obtenido de LEGO Mindstorms: <https://lc-www-live-s.legocdn.com/r/www/r/mindstorms/-/media/franchises/mindstorms%202014/downloads/firmware%20and%20software/advanced/lego%20mindstorms%20ev3%20hardware%20developer%20kit.zip?l.r2=1016700452>

Enlace de descarga de los esquemas electrónicos de los sensores y actuadores del bloque EV3. (4 de 2017). Obtenido de LEGO Mindstorms: <https://lc-www-live-s.legocdn.com/r/www/r/mindstorms/-/media/franchises/mindstorms%202014/downloads/firmware%20and%20software/advanced/lego%20mindstorms%20ev3%20hardware%20developer%20kit.zip?l.r2=1016700452>

Enlace de descarga de los esquemas electrónicos de los sensores y actuadores del bloque NXT. (4 de 2017). Obtenido de LEGO Mindstorms: https://lc-www-live-s.legocdn.com/r/www/r/mindstorms/-/media/franchises/mindstorms%202014/downloads/firmware%20and%20software/nxt%20software/hdk_download1.zip?l.r2=-1260971408

Explicación del Protocolo I2C y el Protocolo I2C en Arduino. (s.f.). Obtenido de Diarioelectronicohoy: <http://www.diarioelectronicohoy.com/blog/introduccion-al-i2c-bus>

Guia de comunicación del bloque EV3. (s.f.). Obtenido de LEGO Mindstorms: <https://lc-www-live-s.legocdn.com/r/www/r/mindstorms/-/media/franchises/mindstorms%202014/downloads/firmware%20and%20software/>

ware/advanced/lego%20mindstorms%20ev3%20communication%20developer%20kit.pdf?l.r2=1239680513

Historia LEGO MINDSTORMS. (n.d.). Retrieved from LEGO MINDSTORM:
<https://www.lego.com/es-es/mindstorms/history>

Industries, D. (2017, 3 24). *Conector LEGO Arduino.* Retrieved from Amazon:
https://www.amazon.com/Breadboard-Adapter-for-LEGO-MINDSTORMS/dp/B00A0P5FY6/ref=as_li_ss_tl?ie=UTF8&linkCode=sl1&tag=de-xteindus-20&linkId=a520dd335dc128dc45affded10098935

Industries, D. (24 de 3 de 2017). *Link de descarga bloque I2C.* Obtenido de Github:
https://github.com/DexterInd/EV3_Dexter_Industries_Sensors

Información acerca del hardware de LEGO EV3. (s.f.). Obtenido de LEGO MINDSTORMS:
<https://www.lego.com/es-es/mindstorms/downloads>

Información complementario del hardware Arduino UNO. (n.d.). Retrieved from Arduino: <https://www.arduino.cc/en/Main/ArduinoBoardUno>

Información del Protocolo I2C. (s.f.). Obtenido de Comunidadelectronicos:
<http://www.comunidadelectronicos.com/articulos/i2c.htm>

Información detallada Protocolo I2C. (s.f.). Obtenido de Sparkfun:
<https://learn.sparkfun.com/tutorials/i2c>

Información general sobre Arduino. (s.f.). Obtenido de Arduino: <http://arduino.cl/que-es-arduino/>

Información relacionada con el conector RJ12. (s.f.). Obtenido de Wikipedia:
<https://es.wikipedia.org/wiki/RJ-12>

Información sobre conectar sensores de LEGO a Arduino. (s.f.). Obtenido de instructables:
<http://www.instructables.com/id/How-to-use-LEGO-NXT-sensors-and-motors-with-a-non-/>

Información sobre el sensor ultrasónico de LEGO. (s.f.). Obtenido de legoengineering:
<http://www.legoengineering.com/tag/ultrasonic-sensor/>

Información sobre el set LEGO. (s.f.). Obtenido de LEGO MINDSTORMS:
<https://www.lego.com/es-es/mindstorms/products/mindstorms-ev3-31313>

Información sobre la función loop. (s.f.). Obtenido de Arduino:
<http://arduino.cc/en/Reference/Loop>

Información sobre la función Setup. (s.f.). Obtenido de Arduino:
<http://arduino.cc/en/Reference/Setup>

Información técnica sobre sensores de LEGO. (s.f.). Obtenido de philohome:
<http://www.philohome.com/motors/motorcomp.htm>

Iniciación a la programación EV3. (s.f.). Obtenido de Electricbricks:
<http://blog.electricbricks.com/2013/11/iniciacion-a-la-programacion-con-ev3/>

LEGO. (26 de 03 de 2017). *Descargar software LEGO MINDSTORMS.* Obtenido de LEGO:
<https://www.lego.com/es-es/mindstorms/downloads/download-software>

Librería Wire(). (n.d.). Retrieved from Arduino:
<https://www.arduino.cc/en/Reference/Wire>

Link de descarga del archivo comprimido. (s.f.). Obtenido de Iván.

Link de descarga RobotC. (s.f.). Obtenido de RobotC:
<http://www.robotc.net/download/lego/>

Motores CC. (s.f.). Obtenido de prometec: <http://www.prometec.net/motorcc/>

Programación básica Arduino. (s.f.). Obtenido de dfists:
http://dfists.ua.es/~jpomares/arduino/page_01.htm

Resistencia LDR con Arduino. (s.f.). Obtenido de Foro de Arduino:
<https://forum.arduino.cc/index.php?topic=108941.0>

RobotC. (s.f.). Obtenido de robotc: <http://www.robotc.net/>

RobotC foro oficial con gran cantidad de proyectos. (s.f.). Obtenido de robotc:
<http://www.robotc.net/blog/>

Sensor fotoelectrónico Arduino. (s.f.). Obtenido de forum.arduino:
<https://forum.arduino.cc/index.php?topic=223583.0>

Sensor infrarrojo de presencia Arduino. (s.f.). Obtenido de hardwarehackingmx:
<https://hardwarehackingmx.wordpress.com/2014/01/15/leccion-20-arduino-sensor-infrarrojo-basico/>

Servomotores con Arduino. (s.f.). Obtenido de aprendiendoarduino:
<https://aprendiendoarduino.wordpress.com/tag/servo/>

Tabla comparativa lenguajes de programación para EV3. (s.f.). Obtenido de Teamhassenplug: <http://www.teamhassenplug.org/NXT/NXTSoftware.html>

9. Anexos.

9.1. Anexo I.

En el presente proyecto se han diseñado y desarrollado una serie de sensores y actuadores para LEGO EV3, pero para que funcionen de forma adecuada es necesario saber cómo conectar estos sensores y actuadores a Arduino.

Arduino presenta una serie de entradas digitales y analógicas configurables como entradas o salidas mediante su programación. Hay otra serie de patillas, para alimentar nuestro circuito a 5V o 3.3V y por lo general tres o dos tomas a tierra.

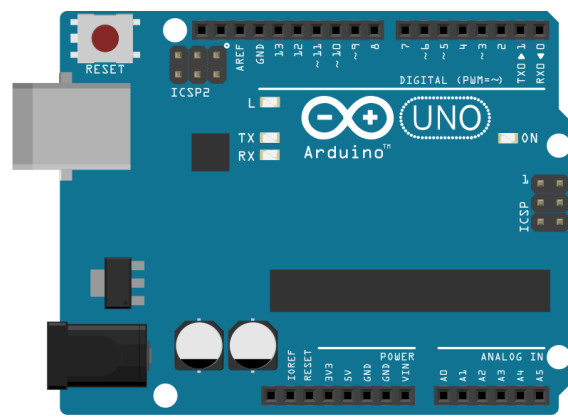


Ilustración 1: Arduino UNO

A continuación, podremos ver cómo se conectan varios dispositivos a Arduino, estos esquemas son los que he utilizado a la hora de desarrollar el presente proyecto:

- LED. En primer lugar, empezaremos como se conecta un LED a Arduino, puede ser sencillo, pero seguro, que todos hemos fundido algún LED alguna vez.

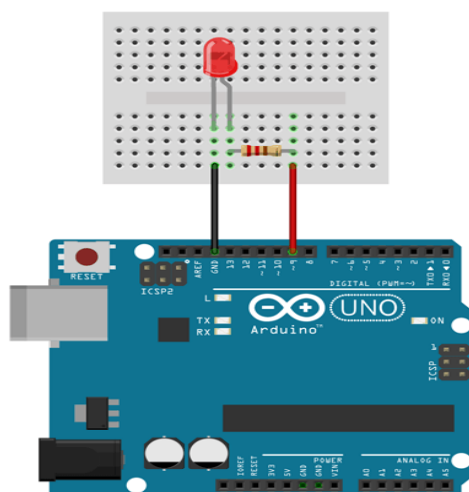


Figura 83: Conexión de un LED a Arduino.

- Sensor infrarrojo. Para realizar la conexión en el Arduino del sensor infrarrojo podemos seguir el siguiente esquema:

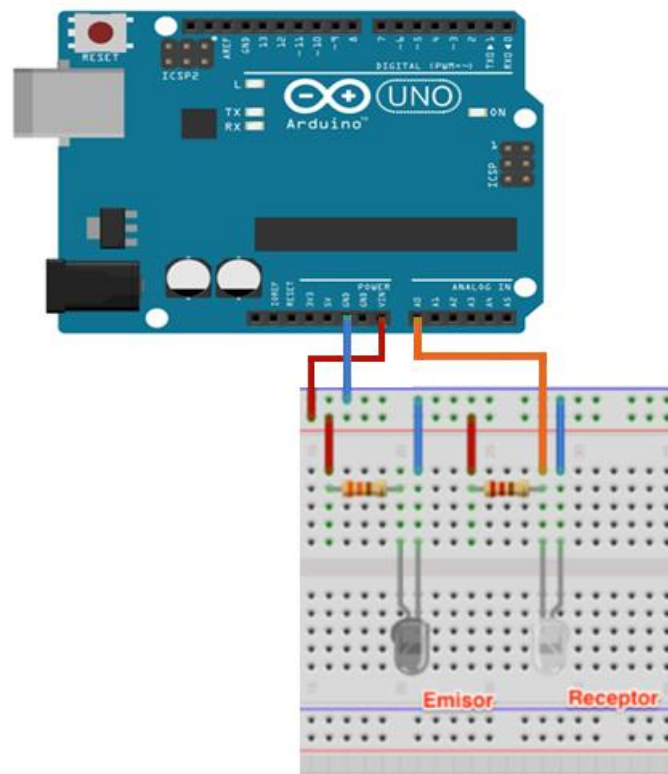


Ilustración 2: Esquema de conexión sensor Infrarrojo barrera

- Sensor de temperatura: concretamente utilicé el LM35

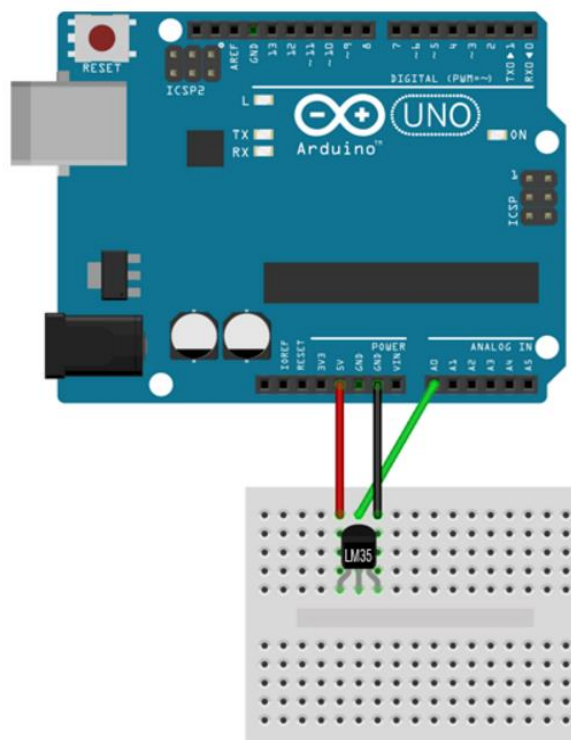


Figura 84: Conexión sensor de temperatura a Arduino

- Servomotor, la conexión del servomotor como podemos apreciar es sencilla:

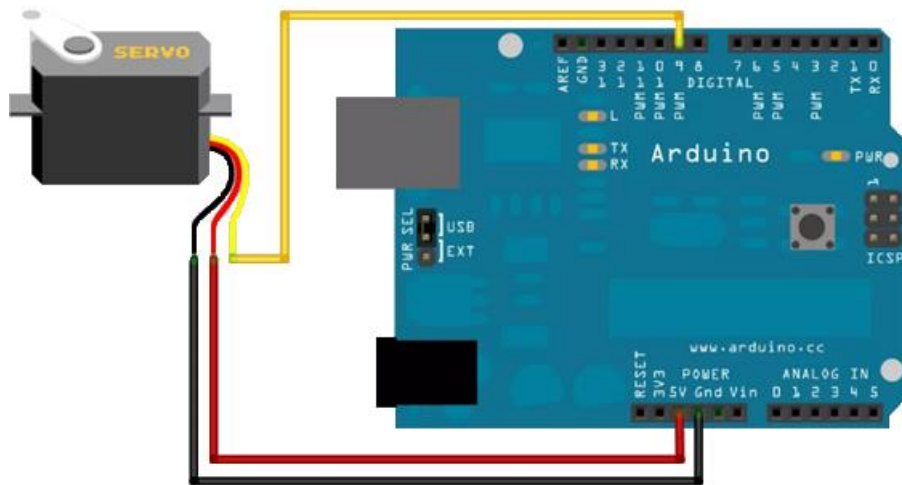


Figura 85: Conexión servomotor a Arduino.

- Motor CC: Este es el componente que más dificultad puede presentar a la hora de conectar a Arduino. Esto es debido a que los pines de Arduino, aunque puedan suministrar un voltaje acorde con el motor (como motores de 4 o 5 V) no presentan la intensidad suficiente. La intensidad disponible por una salida de Arduino es de 100-150 mA, esto es debido a que estas salidas no fueron diseñadas para alimentar grandes cargas. Para mover cualquier motor necesitaremos un mínimo de una intensidad de unos 300mA.

Para poder obtener más de 150mA debemos usar un **punte en H** que permite obtener hasta 600mA. En primer lugar, el puente en H permite cambiar el sentido de giro del motor, su esquema es el siguiente:

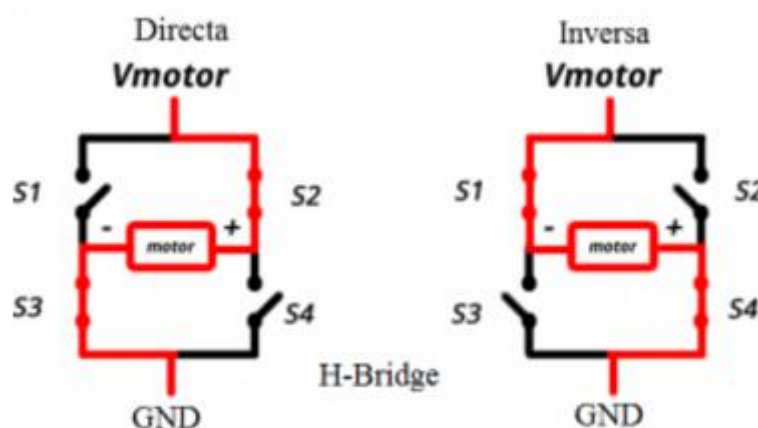


Figura 86: Puente en H.

Cuando usamos los interruptores según la imagen izquierda el motor gira en un sentido. Pero si los colocamos en la posición de la derecha girará en sentido contrario, porque hemos invertido la polaridad de la tensión en las entradas del motor, y por tanto el sentido de giro, sin necesidad de invertir la polaridad de la tensión.

Sin embargo, no solo necesitamos cambiar el giro del motor sino también regular su velocidad y alimentarlo externamente, ya que Arduino no presenta la suficiente intensidad para ello.

Para lograr todo esto debemos de utilizar un driver para motores, aunque hay varios modelos yo he utilizado el L293D. Este driver se compone de un puente H y permite el control de dos motores simultáneamente. El patillaje es el siguiente:

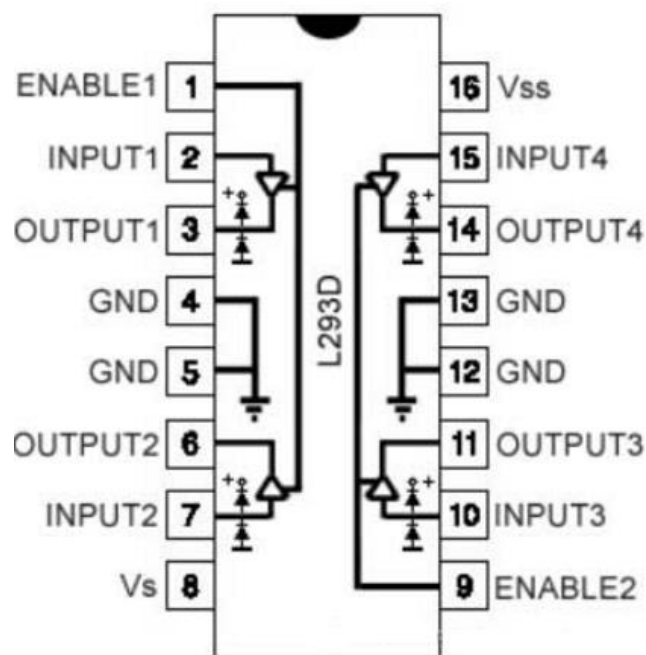


Figura 87: L293D

Veamos los diferentes pines del **L293D**:

- El pin 16, Vss, son los 5V con los que alimentamos el chip y el pin 8, Vs, es la tensión con la que alimentamos el motor.
- Los pines del 1 al 7 controlan el primer motor y los pines 9 a 15 controlan el segundo motor.
- El pin 1, Enable1, Activa el uso del motor 1. Con un valor HIGH, el motor puede girar dependiendo del valor de I1 e I2. Si es LOW se para independientemente de los valores del resto de pines
- Los pines 2 y 7 son los pines de control para el motor 1, e irán conectados a nuestros Arduino para controlar el sentido de giro.
- Los pines 3 y 6 son la salida a la que se conecta el motor 1, cuya polaridad se invierte en función los valores de 2 y 7.

- En el diagrama de arriba veis que hay pines equivalentes para el motor 2 y cuales son.
- Los pines 4, 5, 12 y 13 van a GND.

Podemos resumir todas las conexiones en la siguiente tabla:

Podemos hacer una tabla para mostrar la lógica que sigue el giro del motor en función de los tres pines:

Enable	Control pin 2	Control pin 7	Motor status
LOW	-	-	Motor parado
HIGH	HIGH	LOW	Girar adelante
HIGH	LOW	HIGH	Motor parado
HIGH	LOW	LOW	Motor parado

Por tanto, tenemos que activar el pin ENABLE para que el motor gire y después usamos los pines Input1 e Input2 con valores opuestos para hacer girar el motor en una dirección o en la contraria.

Una vez que comprendemos el uso del integrado L293D, cuando conectamos un motor a Arduino nos podemos encontrar con dos situaciones:

- El motor es inferior a 5V: Si el motor es inferior a 5V el esquema que debemos de seguir el siguiente:

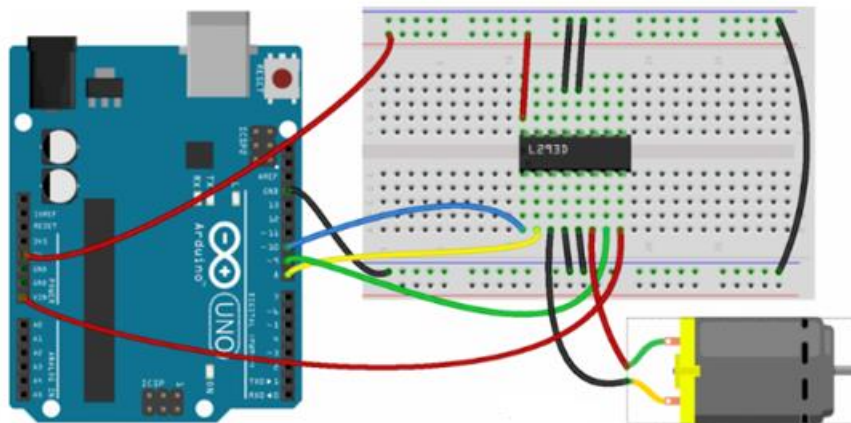


Figura 88: Conexión del encapsulado L293D para motores inferiores a 5V

Las conexiones que debemos de realizar las podemos resumir en la siguiente tabla:

Pin L293D	PIN Arduino	Descripción
1	10	Enable
2	9	Input 1
3	-	Motor +

4,5,12,13	GND	GND
6	-	Motor -
7	8	Input 2
8	Vin	Alimentación del motor
16	5V	Alimentación del L293D

- El motor es superior a 5V, para alimentar estos motores necesitamos alimentación externa ya que el Arduino no puede suministrar más de 5V. El esquema que debemos de seguir es igual que el anterior, pero conectando un segundo motor y poner tanto la alimentación el integrado L293D como la alimentación del motor a una fuente externa de entre 6 y 9V:

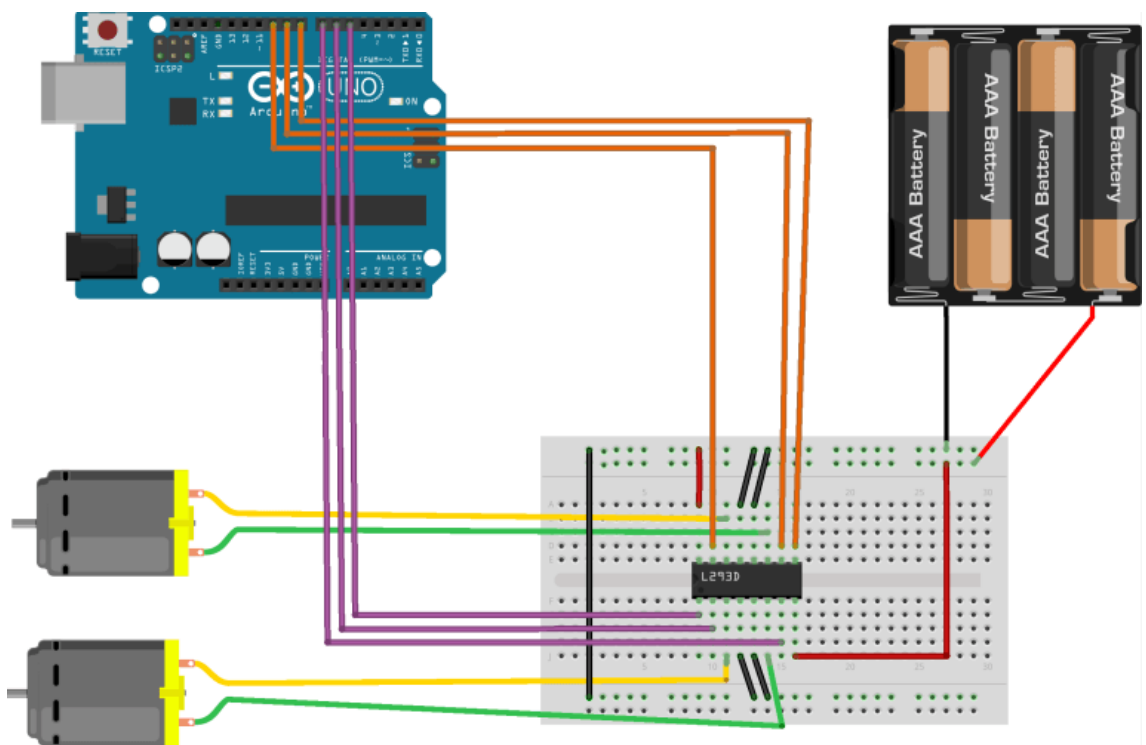


Figura 89: Conexión del encapsulado L293D para motores superiores a 5V

El esquema que hemos seguido es el siguiente:

Pin L293D	PIN Arduino	Descripción
1	10	Enable motor 1
2	9	Input 1 motor 1
3	-	Motor +
4,5,12,13	GND	GND
6	-	Motor -
7	8	Input 2 motor 1
8	Vin	Alimentación del motor
9	6V-9V	Alimentación del L293D
10	5	Input 1 motor 2

11	-	Motor +
14	-	Motor -
15	6	Input 2 motor 2
16	7	Enable motor 2

A continuación, podemos ver uno de cómo controlar el sentido de giro del motor con Arduino:

```
#define e1 10 // Enable Pin for motor 1
#define i1 8 // Control pin 1 for motor 1
#define i2 9 // Control pin 2 for motor 1

void setup()
{
  for (int i = 8 ; i<11 ; i++)
    pinMode( i, OUTPUT);
}
void loop(){
  digitalWrite(e1, HIGH); // Activamos Motor1
  digitalWrite(i1, HIGH); // Arrancamos
  digitalWrite(i2, LOW);
  delay(3000);

  digitalWrite(e1, LOW); // Paramos Motor 1
  delay(1000);
  digitalWrite(e1, HIGH); // Activamos Motor1
  digitalWrite(i1, LOW); //cambio de dirección
  digitalWrite(i2, HIGH);
  delay(3000);

  digitalWrite(e1, LOW); // Paramos Motor 1
  delay(1000);
}
```

El código consiste en mover el motor hacia un sentido durante 3 segundos pararlo 1 segundo y moverlo hacia el otro sentido durante otros 3 segundos

Por último, podemos ver un último ejemplo de cómo controlar la velocidad de un motor conectado a Arduino

```
#define e1 10 // Enable Pin for motor 1
#define i1 8 // Control pin 1 for motor 1
#define i2 9 // Control pin 2 for motor 1

void setup()
```

```

{
  for (int i = 8 ; i<11 ; i++)
    pinMode( i, OUTPUT);
}

void loop(){
  analogWrite(e1,analogRead(A1)/4);
  // Activamos Motor1
  digitalWrite(i1, HIGH); // Arrancamos
  digitalWrite(i2, LOW);
  delay(3000);

  digitalWrite(e1, LOW); // Paramos Motor 1
  delay(1000);

  analogWrite(e1,analogRead(A1)/4);
  // Activamos Motor1
  digitalWrite(i1, LOW); // cambio de dirección
  digitalWrite(i2, HIGH);
  delay(3000);





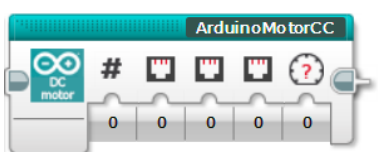
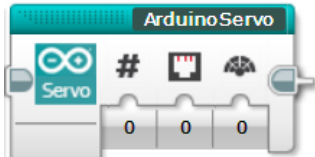
  digitalWrite(e1, LOW); // Paramos Motor 1
  delay(1000);
}

```

Este código es igual que el anterior, pero al inicio de cada cambio de sentido podremos variar su velocidad mediante un potenciómetro conectado en el pin analógico A1.

9.2. Anexo II.

A continuación, mostraremos una tabla resumen de todos los bloques de LEGO MINDSTORMS que hemos desarrollado, resumiendo sus entradas y salidas.

Icono	Nombre	Entradas	Salidas
	Bloque LED	<ul style="list-style-type: none"> ▪ Dirección ▪ Pin de conexión ▪ Estado 	Ninguna
	Bloque Sensor	<ul style="list-style-type: none"> ▪ Dirección ▪ Pin de conexión ▪ Analógico/digital 	Valor del sensor
	Bloque infrarrojo	<ul style="list-style-type: none"> ▪ Dirección ▪ Pin de conexión Emisor ▪ Pin de conexión Receptor ▪ Estado emisor ▪ Analógico/digital 	Valor del sensor
	Bloque interruptor	<ul style="list-style-type: none"> ▪ Dirección ▪ Pin de conexión ▪ Analógico/digital 	Valor del interruptor (1,0)
	Bloque Motor DC	<ul style="list-style-type: none"> ▪ Dirección ▪ Pin de conexión motor (+) ▪ Pin de conexión motor (-) ▪ Pin de conexión Enable 	Valor del sensor
	Bloque servo motor	<ul style="list-style-type: none"> ▪ Dirección ▪ Pin de conexión ▪ Angulo girado 	Ninguna