Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRONICA DE COMUNICACIONES

Trabajo Fin de Grado

DESARROLLO DE EJERCICIOS PRÁCTICOS BASADOS EN ARDUINO PARA EL APRENDIZAJE DE SISTEMAS DIGITALES

Autor: Julia Terrazas Negro

Tutor/es: Juan Jesús García Domínguez

2021

ÍNDICE

ÍNDICE DE ILUSTRACIONES

1 RESUMEN

Este proyecto tiene un objetivo puramente didáctico, cuyo fin es que alumnos fuera del ámbito de la ingeniería se interesen por el mundo de los sistemas digitales.

Para ello se llevará a cabo el desarrollo de un sencillo robot que se pueda controlar con una aplicación móvil desde un smartphone.

En primer lugar se explicará de forma teórica la plataforma en la que se va a desarrollar la aplicación, así como la base de sistemas electrónicos digitales que sean necesarios.

Después se llevará todo lo aprendido a la práctica mediante una serie de experimentos que constituirán el robot.

2 ABSTRACT

This project has a purely educational objective with the purpose of creating an interest into the world of digital systems in students out of the field of engineering.

For making this possible, the construction of a robot that can be managed with a mobile application from a smartphone will be carried out.

First, the platform on which the application will be developed, as well as the basis for the necessary digital electronic systems, will be explained theoretically.

Then everything learned will be put into practice through a series of experiments that will constitute the robot.

3 ACRÓNIMOS Y ABREVIATURAS

- CPU: (Central Processing Unit), unidad central de procesamiento.
- ROM: (Read Only Memory), memoria de solo lectura.
- RAM: (Random Access Memory), memoria de acceso aleatorio.
- LED: (Light Emitting Diode), diodo emisor de luz.
- ADC: (Analog to Digital Converter), conversor de analógico a digital.
- 3D: 3 dimensiones.
- QR: (Quick Response), respuesta rápida.
- UART: Transmisor-Receptor Asíncrono Universal.

4 RESUMEN EXTENDIDO

Este proyecto tiene una función completamente didáctica cuyo objetivo es crear el contenido de una asignatura relacionada con los sistemas electrónicos digitales en un grado fuera del ámbito de la ingeniería.

Como este tipo de asignaturas no suelen ser las más atractivas para los alumnos a los que va focalizado este proyecto, se ha planteado enfocarla de una manera más visual con la intención de hacerla más atractiva, interesante y sobre todo que les motive a aprender sobre sistemas electrónicos digitales.

Para que eso sea posible, se deben introducir poco a poco las bases de todos los conocimientos teóricos, pero de manera concisa y eficaz para dedicar el mayor tiempo posible a su aplicación en los experimentos que se van a realizar. Estos incrementarán gradualmente su complejidad hasta obtener la soltura necesaria para crear un robot sencillo y funcional controlado con un teléfono móvil, que les permita ver reflejados todos los conceptos aprendidos en algo útil y real.

Para llevarlo a cabo se ha planteado la siguiente organización conceptual:

- Primero se explicará de manera teórica los siguientes fundamentos de sistemas electrónicos digitales que se van a necesitar a lo largo de la asignatura, a saber: Arduino, tanto a nivel de software como de hardware; como iniciarse a la programación, tanto los tipos de variables, como los tipos de operadores y las estructuras de control junto con ejemplos claros para su mejor compresión y cómo realizar funciones y por último que es un microcontrolador y cómo funciona.

- En segundo lugar se explicará el funcionamiento de la plataforma Mitt App Inventor 2, la cual se va a usar para desarrollar la aplicación móvil. Para ello primero se aprenderá sobre la parte gráfica, es decir, los objetos con los que interactúa cada usuario al utilizar la aplicación, por ejemplo, botones, imágenes, cuadros de texto, etc. Posteriormente se aprenderá la programación de estos, para poder lograr que interactúen de la manera que se requiera, y completar así el funcionamiento íntegro de la aplicación. La programación de esta aplicación se realiza en bloques, para facilitar su desarrollo ya que, como se ha explicado previamente, este proyecto está enfocado para estudiantes que no han programado antes o que lo han hecho, pero no tienen soltura con ello.

Para este proyecto se ha seleccionado el uso de Arduino, ya que para crear cualquier robot es imprescindible la utilización de un microcontrolador, y Arduino es uno de los más sencillos de manejar. De esta manera, se entra en contacto con los microcontroladores y su programación, pero de la forma más sencilla e intuitiva posible.

Por último, se realizarán una serie de experimentos en los cuales se aplicarán los conceptos que hayan sido aprendidos anteriormente. Estos irán aumentando po-

co a poco su complejidad hasta conseguir construir el robot, demostrando así el aprendizaje de toda la materia. Los experimentos serán un total de cuatro y consistirán en lo siguiente:

1. **Encender un LED:** Se desarrolla una aplicación para el teléfono móvil con tres botones, uno superior para conectarnos con un módulo de Bluetooth a Arduino, para que cuando en la aplicación se pulsen los dos botones restantes de ON y OFF, el LED conectado a la placa de Arduino se encienda o se apague.

   De esta manera se entra en contacto práctico con: la programación del módulo de Bluetooth en Arduino, el cual se utiliza en todos los experimentos, ya que este será el método que se empleará para comunicar el smartphone con la placa de Arduino, la cual gestionará las ordenes que se envíen a través de la aplicación móvil; y por último se inicia en la utilización de la plataforma de Mitt App Inventor 2 creando esta aplicación tan sencilla que solo necesita la programación de tres botones.

2. **Sensores:** En este experimento se desarrolla una aplicación móvil a través de la cual se visualiza la temperatura y la humedad ambiente, así como la distancia a la que se encuentra un objeto del sensor. Para ello, se dispone del sensor de temperatura y humedad, y del sensor de ultrasonido, los cuales mostrarán visualmente los datos que recopilen en pantalla del smartphone.

   Con ello se explicará la manera de hacer visualizaciones en la plataforma de desarrollo de la aplicación, se avanzará un poco en la comunicación por Bluetooth, y se aprenderá a leer sensores y a su vez, a distinguir los datos que proporciona cada uno de ellos.

3. **Mover el Robot:** En este experimento es necesario utilizar el chasis del robot que viene adjunto en el primer anexo. Este chasis ha sido diseñado para que cualquier alumno pueda descargar el archivo e imprimirlo con una impresora 3D, solo será necesario adquirir los motores con las ruedas, y una rueda loca. También se desarrolla una aplicación que constará de 5 botones, el de conectarse con el módulo de Bluetooth, y otros cuatro para poder mover el robot adelante, atrás, izquierda y derecha; y un quinto para activar la comunicación Bluetooth con Arduino.

   Con ello se consigue avanzar en el aprendizaje de la programación y de la electrónica, puesto que se precisa realizar el montaje del chasis del robot con todas sus partes y programar las mismas, además de como a gestionar los diferentes tipos de datos. Y, por último, se interiorizará el manejo en la plataforma de desarrollo de la aplicación.

4. **Robot completo:** Se unificarán los dos experimentos anteriores, de manera que en la aplicación, además de poder visualizar la temperatura y humedad ambiente y la distancia del robot a cualquier objeto, se podrá mover a través de los cuatro botones mencionados anteriormente.

En este último experimento, y como proyecto final, se incorporan todos los conceptos que se han aprendido a lo largo de todo el proyecto. Se afianzarán tanto la programación en Arduino como en el desarrollo de aplicaciones. De esta manera se dará por completado el proceso didáctico que era objetivo este proyecto.

5 ARDUINO

5.1 ¿Qué es Arduino?

Arduino es una plataforma de desarrollo de código abierto[1] que se compone de dos partes, el hardware, denominado tarjeta de Arduino que se refiere a la parte física, compuesta por un microcontrolador re-programable junto con sus periféricos y unos puertos de entrada y salida. Por otro lado, el software, un entorno de desarrollo o entorno de programación que es la herramienta que permite escribir un programa para enviar órdenes a las entradas y salidas, en función de la aplicación que se desee realizar.

Al ser una plataforma de código abierto, cualquier persona puede generar códigos o contenidos, que a su vez pueden ser usados y modificados libremente por todo aquel que quiera descargarlo. Su principal ventaja frente a otras alternativas es la cantidad de información y librerías disponibles para casi cualquier tipo de sensor o dispositivo que se vaya a emplear. Esto es lo que hace que sea tan sencillo utilizar Arduino. Es también una plataforma de hardware libre, lo que significa que tanto los planos o esquemáticos o los componentes que incorporan las tarjetas son de dominio público y cualquier persona puede replicarlos.

5.1.1 Tarjeta Arduino

En la Ilustración 1 aparece el esquema de la tarjeta de Arduino UNO, aunque hay muchas más como por ejemplo Arduino NANO, Arduino MEGA, etc. Se ha elegido la tarjeta más económica y accesible, puesto que es un punto de entrada muy adecuado al mundo de la electrónica digital.

La tarjeta tiene muchas funcionalidades y secciones que se pueden usar en infinidad de aplicaciones, pero solo se van a explicar las básicas y necesarias para llevar a cabo los experimentos a realizar.

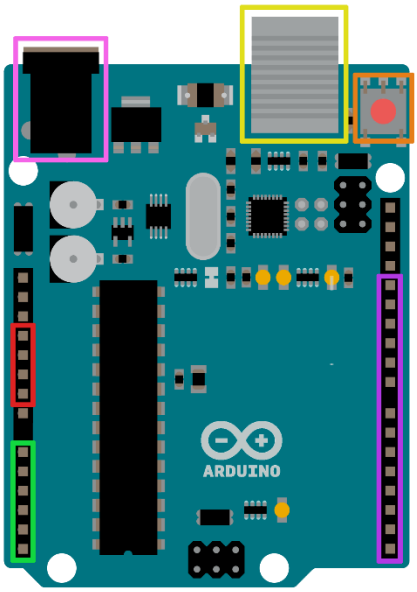

Ilustración 1: Tarjeta Arduino UNO

- **Botón de Reset:**[2] se encuentra en la esquina superior derecha marcado con un cuadrado naranja. Se utiliza para, como su propio nombre indica, hacer un reset del programa que este cargado en esos momentos en la tarjeta. Esto quiere decir, que el código funcionará como si se hubiese cargado por primera.

- **USB Connector:**[2] situado al lado del botón reset, marcado con un cuadrado amarillo, el cual se utiliza tanto para alimentar la placa de Arduino, como para conectarla con el ordenador, y poder cargar así el programa que se ha diseñado previamente.

- **Pines Digitales:**[2] se encuentran a la derecha, marcados con un rectángulo morado. Estos se utilizan como entradas o salidas y solo pueden recibir o enviar un nivel alto (conocido como 1 lógico) con 5V, o un nivel bajo (conocido como 0 lógico) con 0V.

- **Pines Analógicos:**[2] se encuentran en la parte inferior izquierda de la placa y están marcados con un rectángulo verde. Estos tienen la misma funcionalidad que los pines digitales, pero en vez de solo recibir o enviar dos valores, tienen la capacidad de "entender" cualquier valor dentro del rango de 0V a 5V.

- **Pines para Alimentar:**[2] son los que aparecen marcados en rojo, encima de los pines analógicos y en rosa. En los rojos, los dos superiores suministran 3,3V y 5V, mientras que los dos inferiores suministran 0V, que realmente serán la tensión de referencia o masa en las conexiones. En el rosa, que se denomina power Jack, soporta de 7V a 12V.

5.1.2 Entorno de programación

Para desarrollar los códigos que posteriormente se cargan en la tarjeta de Arduino, se utilizará el software llamado Arduino IDE. En este punto se explican todas las características y el modo de empleo.

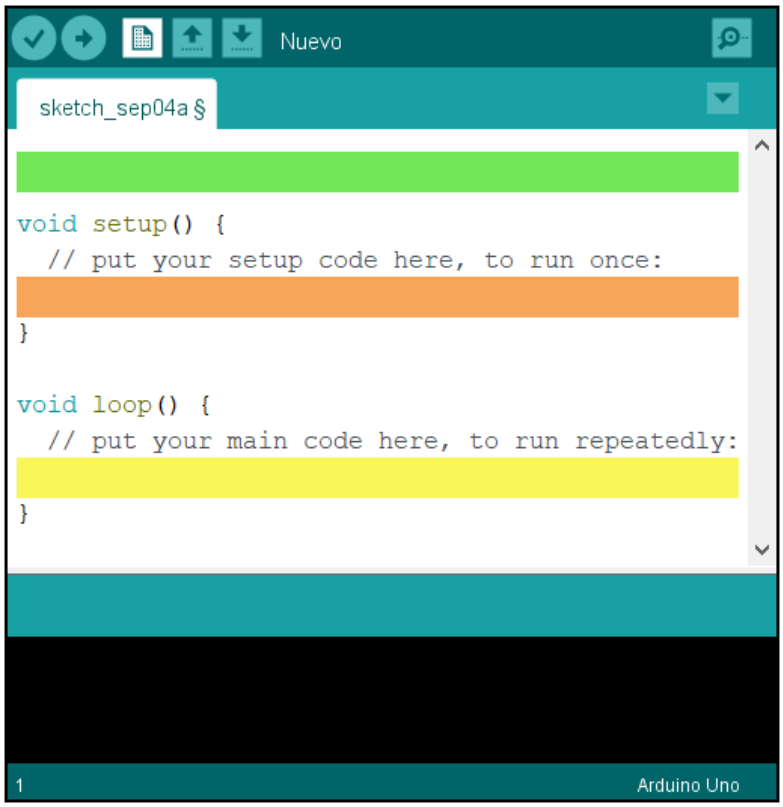

Ilustración 2: Arduino IDE

En la Ilustración 2 aparece el gráfico del software Arduino IDE. La programación a desarrollar consta de tres partes fundamentales, la primera, donde aparece el rectángulo verde, es donde se declaran las constantes o librerías que va a contener el código a desarrollar; las otras resaltadas en naranja y amarillo se refieren a las dos funciones que requiere cualquier aplicación Arduino para funcionar correctamente, las funciones *setup* y *loop* respectivemante. En la función *setup* se declaran las entradas y salidas que se necesitan, así como cualquier tipo de inicialización que requieran las librerías o sensores que se vayan a utilizar. La función *loop* contiene la funcionalidad que se necesite para que la tarjeta Arduino realice las tareas encomendadas en las entradas y salidas, o con cualquier tipo de sensor.

Una vez se haya completado el desarrollo del código, se debe proceder a la comprobación del mismo, es decir, verificar que no se haya cometido ningún error en la pogramación, y por consecuente, cargar el programa en la tarjeta. Para ello Arduino IDE consta de dos botones como los que se muestran en la Ilustración 3; ubicados en la parte superior izquierda.

Ilustración 3: Verificar y Cargar

El boton que aparece en color blanco es el que se utiliza para comprobar los posibles errores que pueda contener el código, mientras que el botón que permanece azul, es el que se debe pulsar para cargar el programa en la placa, cuando el código este libre de errores.

Ahora bien, ¿Qué ocurre si el código contiene algún error? En la Ilustración 4 aparece un ejemplo de un error muy común en la programación, que es no escribir una palabra correctamente. Si esto sucede, o cualquier otro tipo de error, en la parte inferior del entorno de desarrollo, aparecerá en naranja el tipo de error cometido, así como la línea donde se encuentra, subrayada en el mismo color. Un detalle a tener en cuenta, es que a veces el error se indica en una línea, pero la fuente del error es anterior.

```
void loop(){
  digitalWrite(PinLED , HIGH); // Encender LED
  delay(500);                  // Espera de medio segundo
  digitalWrite(PiLED , LOW);  // Apagar LED
  delay(500);                  // Espera de medio segundo
}
```

```
'PiLED' was not declared in this scope

C:\Users\julia\OneDrive\Documentos\Arduino\sketch_jun24a\sketch_jun24a.ino: In function 'void loop()':
sketch_jun24a:11:16: error: 'PiLED' was not declared in this scope
   digitalWrite(PiLED , LOW);  // Apagar LED
                ^~~~
C:\Users\julia\OneDrive\Documentos\Arduino\sketch_jun24a\sketch_jun24a.ino:11:16: note: suggested alternative: 'PinLED'
   digitalWrite(PiLED , LOW);  // Apagar LED
                ^~~~
                PinLED
exit status 1
'PiLED' was not declared in this scope
```

Ilustración 4: Error en código

Si por el contrario, el código está libre de errores, debe aparecer un mensaje de compilado[3], lo cual significa que el software de Arduino IDE, “entiende” todo el código, y por lo tanto puede cargarse en la tarjeta.

Debe tenerse en cuenta que este entorno no tiene la posibilidad de “simular” el código, con lo que es necesario hacer un esfuerzo previo de comprobación de la funcionalidad antes de cargarlo en la tarjeta.

5.2 Introducción a la programación

5.2.3 Tipos de datos

Antes de comenzar a programar, es importante saber el tipo de datos que se están utilizando. A continuación, se explican los más importantes y los necesarios para poder realizar correctamente todos los experimentos.

- **int:**[4] cuando se declara una variable de este tipo se utiliza para representar números enteros, ya sean negativos o positivos, como su valor ocupa 2 bytes en memoria, su rango es de -2^15 a 2^15-1 . A continuación se muestran ejemplos de cómo declararlo:
  - *int a; (a tiene el valor 0).*
  - *int b = 38;*
  - *int c = -21;*

- **float:**[4] las variables de tipo float o de coma flotante se utilizan para representar números no enteros, es decir decimales y ocupan 4 bytes en memoria. Puede ser tanto positivo como negativo, y se declaran de la siguiente manera:
  - *float a = π; (a tiene el valor de 3.1415…).*
  - *float b = 8/9; (b tiene el valor de 0.8888…).*
  - *flota c = -31.58;*

- **char:**[4] este tipo de variables se utiliza para almacenar o números positivos codificados en 8 bits (0 a 255) o el código ASCII de letras o símbolos. Para declararlas es necesario poner su valor entre comillas simples, como se muestra en los ejemplos que aparecen a continuación:
  - *char a* = ‘n’;
  - *char b* = ‘&’;
  - *char c* = ‘A’;

- **string:**[4] las variables de tipo string se utilizan para almacenar cadenas de caracteres (texto), al igual que pueden hacer la misma función de las variables de tipo char y almacenar solo un símbolo o una letra. A continuación, aparecen ejemplos de cómo declararlas:
  - *string a* = “Esto es un tipo de variable”;
  - *string b* = “La fecha de hoy es 12/06/2021”;
  - *string c* = “J”;

Para poder guardar varios datos del mismo tipo en una única variable, se utilizan los arrays. Estos pueden ser unidimensionales o bidimensionales y a continuación se explica con ejemplos como declararlos y usarlos:

- **Array unidimensional:**[5] como su propio nombre indica es una variable de una única dimensión que almacena datos del mismo tipo, en la Ilustración 5 aparece un ejemplo, de un array de tipo int que contiene 5 valores:

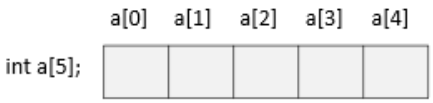

Ilustración 5: Array de 5 posiciones

Se declara con *int a[5];* y para asignar valores, se debe indicar el índice al cual se va a dar un valor, e igualarlo al mismo. Por ejemplo: *int a[0]* = 34; *int a[1]* = 22; *int a[2]* = 5; *int a[3]* = 9; *int a[4]* = 86;

Otra manera de declararlo, sin tener que hacerlo valor a valor es:
int a[5] = {34, 22, 5, 9, 86};

Una vez asignados los valores anteriores se obtendrá un array como el que aparece en la Ilustración 6.

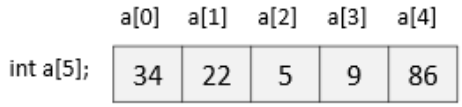

Ilustración 6: Array de 5 posiciones ocupado

- **Array bidimensional:**[5] como su propio nombre indica, las variables de arrays bidimensionales constan de dos índices, el índice de la fila y el índice de la columna, en la Ilustración 7 aparece un array bidimensional de tipo char:

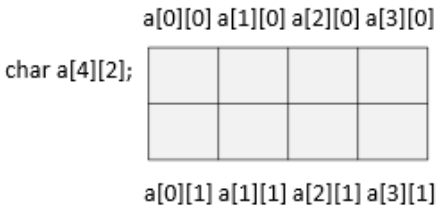

Ilustración 7: Array bidimensional

Se declara con *char a[4][2];* y para asignar valores, se debe indicar el índice al cual se quiere dar un valor, e igualarlo al mismo, es decir:
char *a[0][0]* = 'b'; *char a[1][0]* = 'n'*; char a[2][0]* = '$';
Y así, sucesivamente con el resto de valores, aunque también se pueden asignar de manera directa:
char *a[4][2]* = {'b', 'n', '$', 'g', 'L', '?', 'P', 'v'};
Una vez asignados los valores anteriores se obtiene un array como el que aparece en la Ilustración 8.

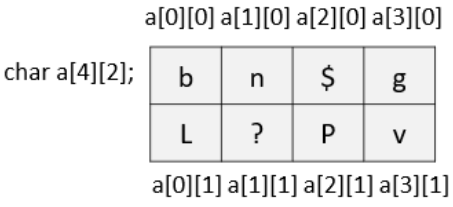

Ilustración 8: Array bidimensional ocupado

5.2.4 Operadores

Para poder realizar cualquier programa, se necesita utilizar operadores que permitan alterar los valores de los datos. A través de estas modificaciones, se consi-

gue asignar los valores deseados a las distintas variables para realizar una gestión correcta de las entradas y salidas de la tarjeta Arduino. Se distinguen varios tipos de operadores como se explica a continuación:

- **Operadores Aritméticos:**[6] se utilizan para realizar las operaciones más simples que se conocen como:

  - Suma: 5 + 8 = 13.
  - Resta: 7 – 2 = 5.
  - Multiplicación: 8 * 2 = 16.
  - División: 6 / 3 = 2.
  - División modular: divide dos números y devuelve el resto: 8 % 3 = 2.

- **Operadores compuestos:**[6] realizan la operación de ambos operadores de izquierda a derecha y el resultado se almacena en el operador izquierdo. Para una mejor compresión se van a ejemplificar los diferentes tipos:

  - +=: por ejemplo x += y sería x = x + y. Es decir, suma ambas variables, y su resultado se adjudica a la variable de la izquierda.
  - -=: por ejemplo x -= y sería x = x - y. Es decir, resta ambas variables, y su resultado se adjudica a la variable de la izquierda.
  - *=: por ejemplo x *= y sería x = x * y. Es decir, multiplica ambas variables, y su resultado se adjudica a la variable de la izquierda.
  - /=: por ejemplo x /= y sería x = x / y. Es decir, divide ambas variables, y su resultado se adjudica a la variable de la izquierda.
  - %=: por ejemplo x %= y sería x = x % y. Es decir, realiza la división modular de ambas variables, y su resultado se adjudica a la variable de la izquierda.

- **Operadores de comparación:**[6] como su propio nombre indica, son operadores cuya función es la de comparar dos variables. Si la comparación se cumple, la sentencia es verdadera, y por lo tanto el resultado será un 1. En caso contario la sentencia es falsa, el resultado será 0. Estos son los distintos tipos de operadores de comparación:

  - Operador igual: x == y.
  - Operador no igual: x != y.
  - Operador mayor que: x > y.
  - Operador menor que: x < y.
  - Operador mayor o igual que: x >= y.
  - Operador menor o igual que: x <= y.

- **Operadores lógicos:**[6] comprueban si el valor de dos o más operandos se cumple o no, siendo el resultado, al igual que los operadores de comparación, verdadero con un 1 o falso con un 0.

- Operador conjugación lógica “&&”: x ==1 && y == 3, si ambas variables tienen ese valor, el resultado de la operación será un 1, si alguna de las dos variables no cumple con dicho valor el resultado será 0.
- Operador de disyunción lógica “||”: x == 1 || y == 3, si se cumple alguno de los dos operandos, o los dos, el resultado será 1, mientras que si no se cumple ninguno el resultado será 0.

- **Operadores a nivel de bit:**[15] se utilizan para manipular a nivel de bits. A continuación, aparecen los diferentes tipos:

  - AND “&”: se entiende como una multiplicación, en la siguiente tabla, aparecen todas las combinaciones posibles.

A	B	Y = A&B
0	0	0
0	1	0
1	0	0
1	1	1

  - OR “|”: se entiendo como una suma, en la siguiente tabla aparecen los resultados para todas las combinaciones posibles.

A	B	Y = A|B
0	0	0
0	1	1
1	0	1
1	1	1

  - XOR “^”: comprueba si los operandos son iguales o diferentes, si son iguales el resultado será 0, mientras que son distintos, será 1. En la tabla aparecen todas las combinaciones posibles.

A	B	Y = A^B
0	0	0
0	1	1
1	0	1
1	1	0

  - Desplazamiento derecha “>>” desplaza los bits del operando hacia la derecha, introduciendo un 0 a la izquierda. Por ejemplo:
    Si *a* = 00010000 y *b* = *a* >> 1, el valor de *b* ahora será 00001000. Desplaza un bit a la derecha.
  - Desplazamiento izquierda “<<” desplaza los bits del operando hacia la izquierda, introduciendo un 0 a la derecha. Por ejemplo:
    Si *a* = 00010000 y *b* = *a* << 2, el valor de *b* ahora será 01000000. Desplaza dos bits hacia la izquierda.

5.2.5 Estructuras de control

Las estructuras de control se utilizan para poder gestionar los diferentes procesos que se quieran controlar y dar a los programas una estructura más lógica. Estas se dividen en dos tipos fundamentales:

- **Selección:** cuando las sentencias concretas que se hayan establecido se cumplan, realiza una serie de operaciones, o bien otra estructura de control. Hay tres tipos fundamentales:

  - **if**:[7] la sintaxis de esta estructura es:

```
if(expresion)
{cualquier tipo de sentencia que se quiera aplicar}
```

    Si la expresión se cumple, es decir, es verdadera, se aplicarán todas las sentencias que se hayan declarado dentro de las llaves. Por ejemplo: se quiere diseñar un programa, que cuando la variable *x* tenga un valor de 5, esta pase a tomar el valor 20:

```
if(x == 5){
x = 20;
}
```

  - **if… else**:[7] la sintaxis de esta estructura es:

```
if(expresión)
{cualquier tipo de sentencia que se quiera aplicar}
else
{cualquier tipo de sentencia que se quiera aplicar}
```

    Su funcionamiento es igual que el de la estructura *if*, pero con la cláusula de *else*, cuya finalidad es que aplique las sentencias que tiene entre las llaves, en caso de que la expresión del *if* no se cumpla. Es decir, que sea falsa. Para que se entienda mejor se va a aplicar el anterior ejemplo, pero ahora si *x* no tiene el valor de 5, *x* incrementará en uno su valor:

```
if(x == 5){
x = 20;
}
else{
x = x + 1;
}
```

  - **switch**:[8] la sintaxis de esta estructura es:

```
switch(variable){
   case valor de la variable:
      sentencias que se quieran aplicar;
   break;
  case…
  break;
}
```

Su funcionamiento consiste en evaluar los posibles valores de una variable, y en función del mismo, aplicar una serie de sentencias. Para una mejor comprensión se va a explicar con un ejemplo: si la variable *x* vale 1, a la variable y se le asigna el valor de 8, si la variable *x* vale 2, a la variable y se le asigna el valor de 9.

```
switch(x){
   case 1:
      y=8;
   break;
  case 2:
      y=9;
  break;
}
```

- **Repetición**: estas estructuras se utilizan cuando se quiere repetir la misma sentencia varias veces. Hay tres tipos fundamentales:
- **for:**[9] la estructura *for* tiene la siguiente sintaxis:

```
for(int i=0;i<a;i++){
sentencia a ejecutar;
}
```

Su funcionamiento consiste en repetir las sentencias tantas veces como indique la variable *i* hasta que se cumpla una condición. La primera parte inicializa a *i* (*i=0)*, la segunda indica la condición a verificar (*i<a*) y la tercera varía a *i*, en este caso se incrementa de uno en uno. La primera vez que se ejecute el bucle, *i* valdrá 0, la segunda 1, y así progresivamente hasta obtener el valor de *a* -1. Para una mejor comprensión se va a explicar a continuación con un ejemplo: el ejemplo consiste en sumar en *x* todos los valores del array unidimensional de *y*, que se va a suponer que tiene 8 valores, y *x* comienza teniendo el valor de 0.

```
for(int i = 0;i<8;i++){
x=x+y[i]
}
```

- **do…while:**[10] la sintaxis de esta estructura es:

```
do{
sentencias a ejecutar;
}
while(i<8)
```

Esta estructura tiene un funcionamiento parecido al del bucle *for*. Las sentencias dentro de las llaves de la parte del *do*, se realizaran una y otra vez, siempre y cuando se cumpla la expresión del *while*. Para una mejor compresión se va a realizar el mismo ejemplo que en el bucle for anterior:

```
do{
x = x + y[i];
i++;
}
while(i<8)
```

5.2.6 Funciones

Cuando se realiza un programa sencillo, comúnmente se tiende a incluir todas las sentencias en el bucle principal del programa. Pero cuando se avanza se desarrollan códigos más complejos, con sentencias que se repitan a lo largo del mismo. En este casi es importante incluir funciones, para poder sintetizarlo, y hacerlo más sencillo.

Una función es una porción de código destinada a realizar una tarea muy concreta. En vez de encadenar sentencias en el bucle principal se llamará a una función para que realice esa tarea en su lugar.

El mismo bucle principal ya se encuentra dentro de una función llamada *loop*, pero no tiene por qué ser la única función del programa, se pueden declarar tantas funciones como que quiera o tantas como tareas concretas se precisen.

La dinámica básica de funcionamiento es la siguiente: se inicia el programa, y después de ejecutarse la función *setup*, se comienzan a ejecutar las sentencias presentes en la función principal (*loop*). Pero si una de estas líneas es una llamada a otra función, ejecutará dicha función. Para lograr hacer esto, Arduino guarda en su memoria la línea donde se realizó la llamada a esa función y de esta manera cuando termine con su ejecución volverá a la función *loop* para continuar con el resto de las sentencias.

Las funciones pueden recibir y devolver datos, de esta manera se pueden pasar datos de entrada a una función para que esta devuelva un resultado. Para ejemplificar esto se plantea un programa que se dedique a realizar operaciones matemáticas,se podría estructurar dispon de una función para cada tipo de operación: funcionSuma, funcionMultiplicacion… La idea de función es útil en estos casos, cuando se realizan operaciones varias veces o de forma cíclica, en vez de encadenar un conjunto de sentencias tantas veces como se quiera realizar esa opera-

ción, se llama a una función para que realice esas operaciones y devuelva el resultado.

Al introducir funciones en el programa hay que tener en cuenta dos cosas, el prototipo y la declaración. El prototipo es una sentencia en la que se especifica el nombre de la función, que recibe y que devuelve. Esta sentencia se debe emplazar en el código antes de la función *loop*. La declaración de la función es la porción de código donde se especifica la tarea concreta que vaya a realizar. Para aclarar eso, se puede pensar en una función que realiza la media de tres números enteros, el prototipo de esta función advierte al compilador de que esa función recibirá tres parámetros de entrada y que estos serán tres números enteros y también que devolverá un parámetro de salida que será un numero con coma flotante y no un entero puesto que la media no siempre dará como resultado un número entero.

```
float funcionMedia(int, int, int);

resultado funcionMedia(a,b,c){
      float resultado;
      resultado = (a+b+c)/3;
      return resultado;
}
```

Como se ha adelantado y como se muestra en el cuadro de texto anterior a la función funcionMedia entran tres datos y sale uno, antes del nombre de la función se escribe el parámetro que devuelve y después del nombre de la función, y entre paréntesis, los parámetros que recibe. Es importante conocer que también puede haber funciones que no reciban ni devuelvan datos, para especificar esto se utiliza la palabra void como si de un parámetro más se tratase, de tal forma que si no devuelve nada la palabra void, aparecerá precediendo al nombre de la función y si no recibe nada, la palabra void aparecerá tras el nombre de la función y entre paréntesis. Un ejemplo claro son las funciones setup y loop presentes en cualquier código de Arduino.

De esta manera, la función generada consta del nombre funcionMedia, de tres parámetros de entrada, que son a, b, y c, que son los datos sobre los que se realiza la media y un parámetro de salida que es donde se almacena el resultado de la operación. Una vez se suman y se dividen entre tres, el valor de la media de esos tres datos se almacena en la variable resultado que es de tipo float. De esta manera al ejecutarse la función funcionMedia esta devolverá el resultado en la variable resultado el cual se puede recoger en una variable. De modo que, cada vez que se quiera realizar la media de tres números enteros en el código, no habrá necesidad de concatenar sentencias, simplemente se llamará a funcionMedia para que realiza los cálculos y devuelva el resultado, en cualquier parte del código donde se quiera implementar, se realizará una llamada:

```
float r = funcionMedia(8,3,4);
```

De esta forma, r pasa a tomar el valor de la media de 8, 3 y 4.

5.2.7 Interrupciones

Una interrupción es un mecanismo que permite asociar una función a un evento concreto. Cuando se da este evento, el procesador pasa a atender la función asociada al mismo, y una vez finalizada su ejecución, volverá al punto donde estaba antes. La función asociada a una interrupción se denomina callback y suele llevar las siglas ISR (Interruption Service Routine).

Para entender la utilidad y necesidad de las interrupciones, se plantea el ejemplo de una máquina expendedora de bebidas. Las tareas que la máquina debe llevar a cabo pueden ser dar la orden "enfriar bebidas", la orden "mostrar información en pantalla" o la orden "atender al usuario". En primera instancia, serviría con encadenar estas órdenes, se enfriarían las bebidas, se mostraría la información correspondiente en pantalla y se comprobaría si algún usuario ha solicitado una bebida. La ventaja que aquí presenta una interrupción es que se puede asociar la función "atenderUsuario" al evento "nueva moneda en el monedero", de tal manera que solo en el caso de que haya una nueva moneda en el monedero se pasará a atender a un usuario. Haciendo esto se libera al procesador de la tarea de comprobación del monedero cuando no hay ningún usuario frente a la máquina, solo se pasará a atender al mismo, cuando este haya introducido una moneda en la máquina. Así, cuando un usuario introduzca una moneda, el procesador dejara de ejecutar las funciones "enfriarBebidas" y "mostrarInformación" para ejecutar la función "atenderUsuario" para servir la bebida. Una vez servida (y terminada la ejecución de la función "atenderUsuario") el procesador volverá a ejecutar las funciones funciones "enfriarBebidas" y "mostrarInformación".

El uso de interrupciones en un programa tiene ciertas ventajas. Si un programa debe dar una respuesta ante un evento, como si una bombilla está encendida o apagada, habrá que ejecutar todo el bucle principal del programa hasta llegar a las líneas del código donde se haga esta comprobación del estado de la bombilla, por tanto, se pierde tiempo, y si la respuesta requiere de inmediatez, las interrupciones se hacen esenciales. Su uso también supone un menor consumo de recursos del procesador y por ende, de energía. El procesador no tendrá que comprobar de forma continua sobre algún pin, será el pin quien avise al procesador de que el evento ha sucedido. Algo que puede ocurrir sin el uso de interrupciones, es que si el evento al que se debe reaccionar es muy corto en el tiempo, debe coincidir el momento en el que este se produce con el momento en el que se ejecutan las líneas de código que realizan la comprobación.

Arduino UNO dispone de dos interrupciones hardware, esto significa que estas interrupciones están asociadas a pines del procesador (en el caso de Arduino UNO pines digitales 2 y 3). La señal del monedero de la máquina de bebidas se conectaría a unas de estos pines asociados de tal manera que al introducir una moneda se dispararía la función asociada a esa interrupción. Cabe destacar que las funciones de atención a interrupciones deben ser lo más cortas posibles, porque mientras se estén ejecutando, no se estará atendiendo al resto de código.

Para definir interrupciones en Arduino se utiliza la función "attachInterrupt" que se encarga de asociar una interrupción (la del pin 2 o 3) a la función que se eje-

cutará cuando esta suceda. Un detalle a tener en cuenta es que un pin digital tiene dos maneras de reconocer un evento: por flanco (rising or falling) o por nivel (high o low), dependiendo del tipo de evento, puede utilizarse un modo u otro.

```
attachInterrupt(2, atenderUsuario, HIGH);
```

Si en vez de atender a un usuario en una máquina de bebidas, se quisiera encender un led cada vez que hay un estado alto en el pin 2, bastaría con sustituir la función “atenderUsuario” por la función que haga parpadear un led.

Existe un tipo especial de interrupciones, las interrupciones programadas o también llamadas Timers. El concepto es el mismo que el de una interrupción normal, pero en este caso, el disparo de la interrupción no viene determinado por un evento hardware (como el cambio de estado en un pin) si no al transcurrir un tiempo preciso, el cual se ha fijado antes. Arduino posee un reloj interno, un cristal de cuarzo que vibra 16000000 veces por segundo (16MHz), lo que significa que el procesador puede determinar tiempo con una precisión de 1/16000000 segundos, a este tiempo se le denomina “tick”. El funcionamiento es el siguiente, cada Timer tiene asociados unos registros en los que se puede configurar cuantos "ticks" han de pasar para que se dispare la interrupción.

La tarjeta Arduino Uno dispone de tres Timers o interrupciones programadas, uno de 8 bits, timer 0, otro de 16 bits, timer 1 y otro de 32 bits, timer 2. Se encargan de ejecutar su función asociada cada cierto tiempo, tiempo que el usuario especificará. Es decir, mientras el programa sigue aplicando las sentencias que le hayan sido configuradas, los Timer cuentan, cuando alcanzan el valor (en tiempo) con el que hayan sido configurados, el procesador para de ejecutar el hilo principal del código para atender la función asociada, de la misma manera que sucede con las interrupciones hardware.

Para utilizar los Timers es necesario tener ciertos conocimientos sobre manipulación de registros, puesto que hay que modificar una serie de bits de los mismos. Como este proyecto está orientado a personas con bajos conocimientos de programación se propone el uso de una de las muchas librearías que la comunidad de Arduino ha desarollado. Esta permite al usuario inexperto hacer uso de las propiedades del timer, sin poseer los conocimientos, a priori, necesarios. La libraría se llama TimerOne y esta disponible a través del Arduino IDE. Através del gestor de librerías se busca: TimerOne, la desarrollada por el usuario Paul Stoffregen. Esta libreria incluye varias funciones para el manejo del timer, entre ellas initialize(period) para especificar el periodo del timer y attachInterrupt(function, period) para asignar al timer su correspondiente ISR.

De esta manera, basta con inicializar el timer en la parte de setup con el tiempo (en microsegundos) con la que se quiere implementar y ligarlo a la función que se quiera ejectuar.

Por ejemplo, se quiere que un LED se apague y se encienda cada 0.5 segundos.

```
void setup(){
Timer1.initialize(500000); //cada medio segundo
            Timer1.attachInterrupt(ISR_LED);
  }

void ISR_LED(){
      if(stado=HIGH){
          estado = LOW;
}
else{
  estado = HIGH;
}

digitalWrite(LED, estado);
}
```

6 MICROCONTROLADOR

Un microcontrolador es un "Chip procesador de propósito especial diseñado y construido para gestionar una determinada tarea específica, y que está constituido por una unidad central de procesamiento, memoria propia, canales de entrada/salida (puertos) y temporizadores."[11]. Para una mejor compresión, se va a proceder a explicar todas sus partes funcionales, que son las que aparecen en la Ilustración 9.

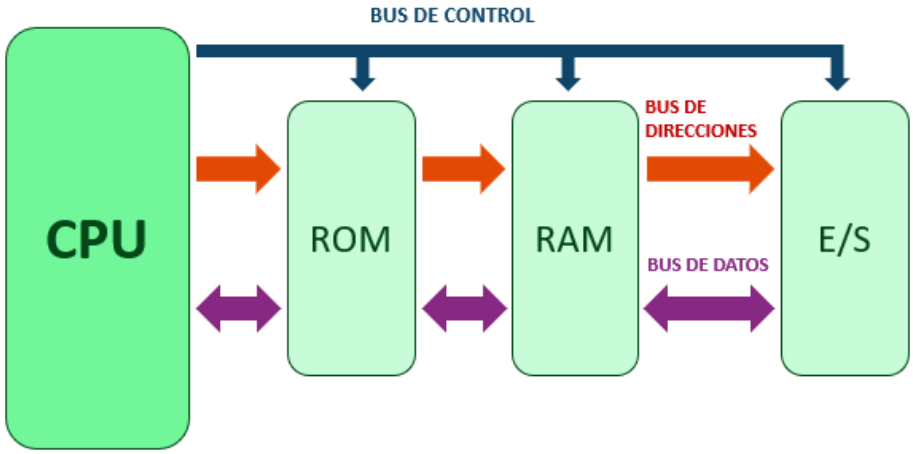

Ilustración 9: Partes funcionales de un microcontrolador

- **ROM:** memoria de solo lectura, en este bloque es donde se almacena el programa que haya sido previamente desarrollado, y por tanto, todas las instrucciones que se deben llevar a cabo. Se conoce también como memoria de programa.

- **RAM:** memoria de acceso aleatorio, en este bloque se almacenan los diferentes valores que toman las variables del programa almacenado, por eso se conoce también como memoria de datos.

- **Entradas y salidas:** este bloque es en el cual se ve reflejado el funcionamiento del programa, es decir, el medio de comunicación entre el microcontrolador y el "mundo exterior". Por ejemplo: la pantalla de un ordenador, el teclado, el ratón, los sensores, un módulo de Bluetooth, etc.

- **CPU:** Es la unidad central de procesamiento, dicho coloquialmente, es el "cerebro" del microcontrolador, es el encargado de gestionar todas las órdenes que haya en el programa, a través de una serie de operaciones lógicas y aritméticas.

- **Bus de control:** a través del cual la CPU envía ordenes al resto de los bloques del microcontrolador.

- **Bus de datos:** a través del cual se envían información entre los bloques para que estos puedan desempeñar las funciones encomendadas.

- **Bus de direcciones:** a través del cual se envían las direcciones en las cuales el microcontrolador va a operar, esto es, leer o escribir datos.

7 MIT APP INVENTOR

Mitt App Inventor 2 es un entorno de desarrollo de aplicaciones para Android. Está orientado hacia una programación visual y en bloques, por lo que resulta muy sencillo para personas que se están iniciando el mundo de la programación.

Esta herramienta se compone principalmente de dos partes, la parte de diseño y la de bloques. En la parte de diseño, se dispone de todos los objetos que van a aparecer en la pantalla en la cual se visualice la aplicación, así como los no visibles, pero que se utilizan para una correcta interacción con los posibles usuarios.

En la parte de los bloques es donde se realizará la programación de todos los objetos que se han incluido en la parte de diseño. Esta parte se llama programación en bloques porque al contrario que en Arduino, esta no se realiza escribiendo manualmente el código, sino, cogiendo los bloques que hay disponibles.

Para poder comenzar a realizar aplicaciones, primero hay que registrarse en su página web: https://appinventor.mit.edu/explore/ai2, la cual se muestra en la Ilustración 10. Aquí, pulsando en el botón naranja de Create Apps! Podremos realizar nuestro registro en la plataforma.

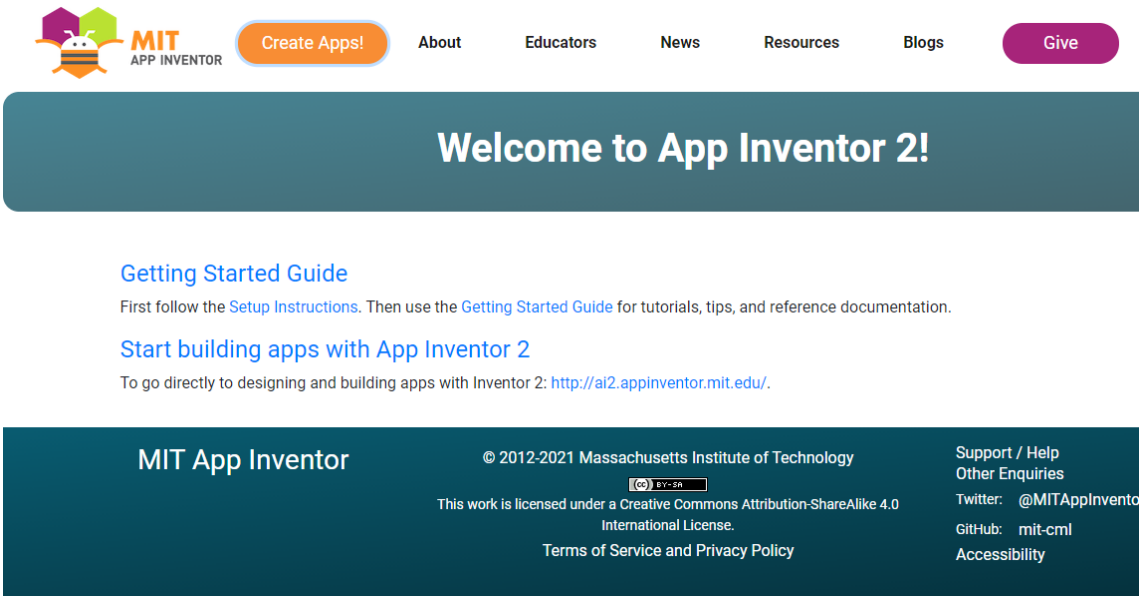

Ilustración 10: Sitio web MIT APP Inventor

Una vez dentro, como aparece en la Ilustración 11 se comienza un nuevo proyecto pulsando sobre el botón rodeado en rojo de Start new Project.

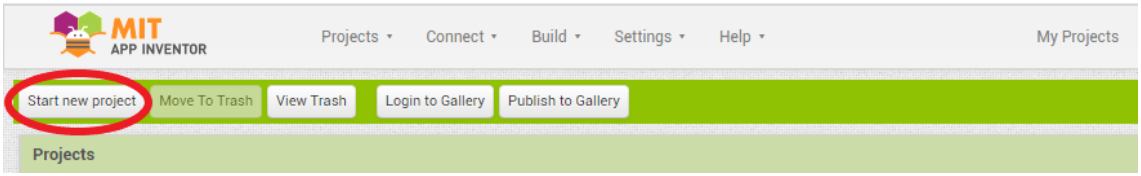

Ilustración 11: Comienzo de nuevo proyecto en MIT APP Inventor

7.1 Diseño

Una vez abierto en nuevo proyecto (Ilustración 12), aparece la parte de diseño, donde se pueden empezar a incluir los bloques que se necesiten.

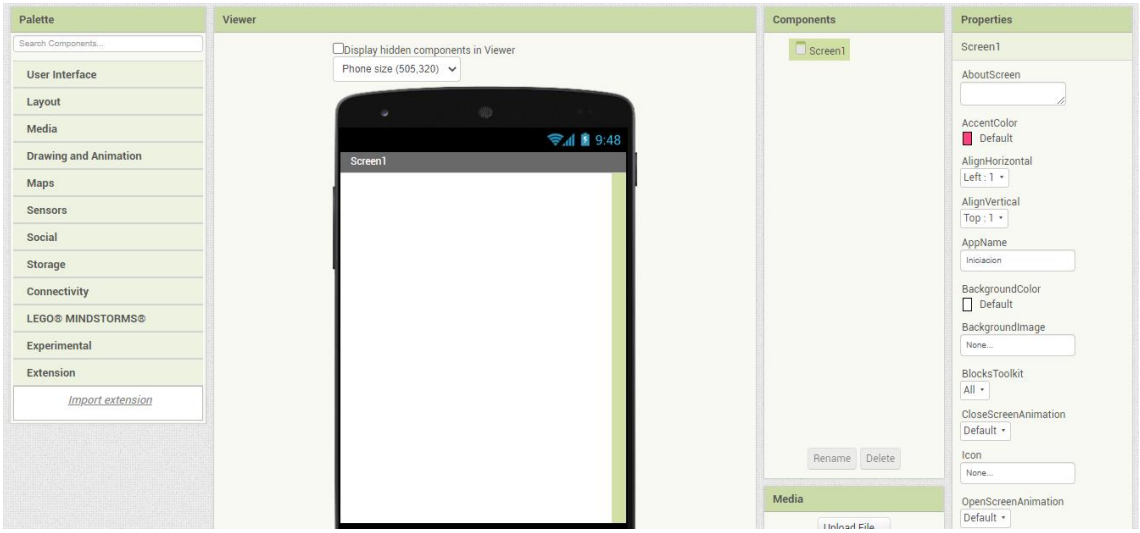

Ilustración 12: Nuevo proyecto en MIT APP Inventor

En la parte izquierda, se sitúan las paletas con todos los posibles objetos que se pueden introducir en cualquier aplicación. Estos aparecen separados por secciones para que sea más sencillo encontrar los elementos que se quieran utilizar.

En la zona central se encuentra la parte de viewer, esto es una pantalla que simula un smartphone para ir visualizando el aspecto que tiene la aplicación que se esté desarrollando.

A la derecha del viewer, se muestra un listado con todos los componentes que han sido añadidos al diseño.

Por último, en la parte izquierda, se encuentran las propiedades que se pueden modificar de los objetos que hayan sido incluidos, así como, color, tipo de letra, posición, etc.

Para empezar a diseñar una aplicación, primero se debe tener claro los componentes que se necesitan para llevar a cabo su correcto desarrollo. Para seleccionarlos, se hace click en su sección y seguidamente el objeto que se quiere in-

cluir. Las funcionalidades de cada uno se pueden ver en la interrogación que aparece a la derecha del componente, como en la Ilustración 13

Ilustración 13 Funcionalidad de cada componente

Para comprender mejor el proceso de diseñar una aplicación, la explicación se va a realizar con la ayuda de un ejemplo. El ejemplo es un contador que debe incrementar la cuenta cuando se pulse un botón, y reiniciar la cuenta cuando se pulse otro.

Lo primero que se debe hacer es decidir cuantos objetos se necesitan para implementar dicha aplicación. En este caso son dos botones y dos cuadros de texto donde aparezca el nombre de cuenta y el número actual de la cuenta. Cada elemento que se debe incluir dentro de una disposición (layout) de la pantalla. En este caso se utilizan 3 disposiciones horizontales, una para cada botón y otra para los cuadros de texto. En el superior parecerá el botón de incremento de cuenta, en el inferior el botón de reset, y en el del medio dos cuadros de texto, uno con un texto fijo, y otro variable. Para definir sus longitudes, colores o posición en la pantalla, se selecciona en la parte de propiedades, y en este ejemplo se ha decidi-do que los botones ocupen un 30% cada uno a lo largo, y de ancho completen to-da la pantalla. Todo esto aparece en Ilustración 14. Mientras que el layout de los textos ocupa el resto de la zona central.

Ilustración 14: Definiendo layout de un objeto

A continuación, se procede a explicar cómo insertar todos los elementos mencionados previamente:

• **Botón 1:** botón para incrementar el contador en 1, como aparece en la Ilustración 15 se ajustan todas sus propiedades según se desee. En este caso se ha utilizado la letra a tamaño 40 (fontsize), que ocupe el tamaño completo del layout, en text ponemos el nombre que queremos que aparezca sobre el botón, y en el apartado inferior su color.

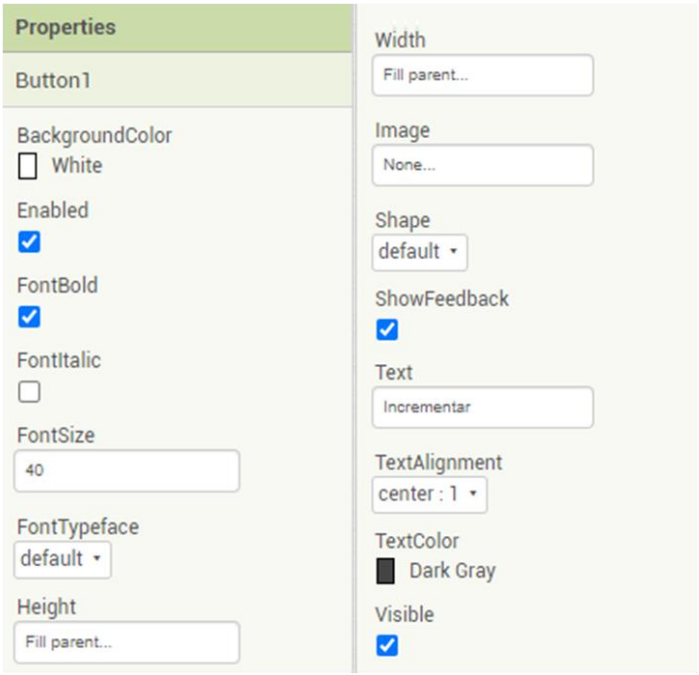

Ilustración 15: Definiendo propiedades de objetos en MIT APP Inventor

En la parte del componente se cambia el nombre, para que cuando se tenga que programar dicho botón en la parte de bloques sea más fácil de identificar. Para ello se pincha sobre la parte inferior, como aparece en la Ilustración 16, donde aparece “rename” y se asigna el nombre: “incrementar”.

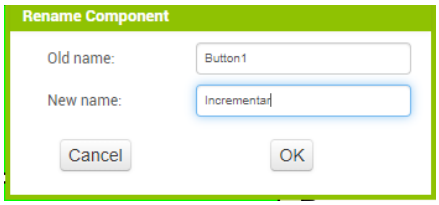

Ilustración 16: Renombrando un objeto en MIT APP Inventor

- **Botón 2: s**e aplican las mismas propiedades que con el botón 1, pero su texto será “reset” en vez de “incrementar”, al igual que el nombre del componente se debe modificar por “reset”.

- **Texto/ Etiqueta 1:** el texto interior no va a variar, va a ser “cuenta”, y va a ocupar un 50% del ancho de la pantalla. El resto de sus propiedades aparecen en la Ilustración 17.

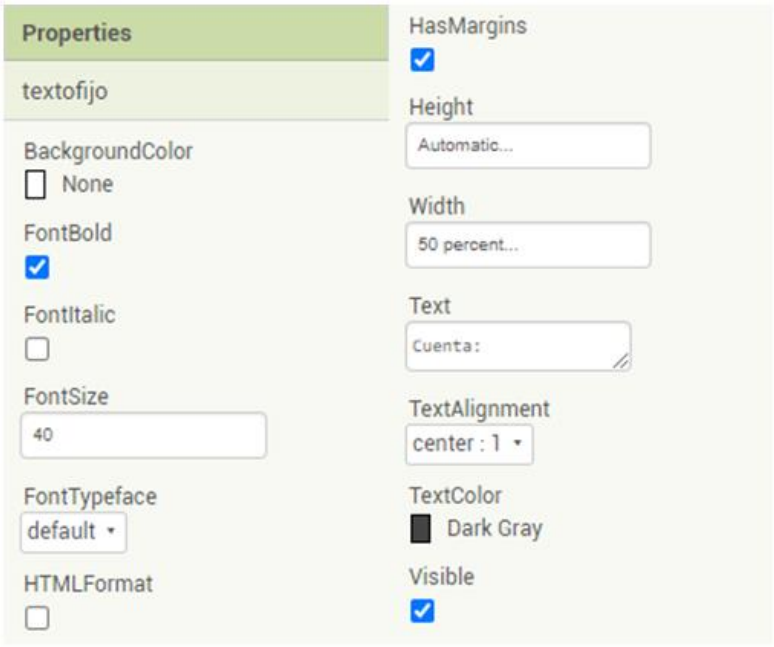

Ilustración 17: Propiedades de etiqueta en MIT APP Inventor

- **Texto/ Etiqueta 2:** se realiza el mismo proceso que con el texto dos, aunque su interior, al ser el resultado de la cuenta, se modificará conforme se incremente la misma.

Una vez realizado todo el proceso descrito previamente, la visualización de la aplicación se da por finalizada y completada así la parte de diseño. Su resultado aparece en la Ilustración 18.

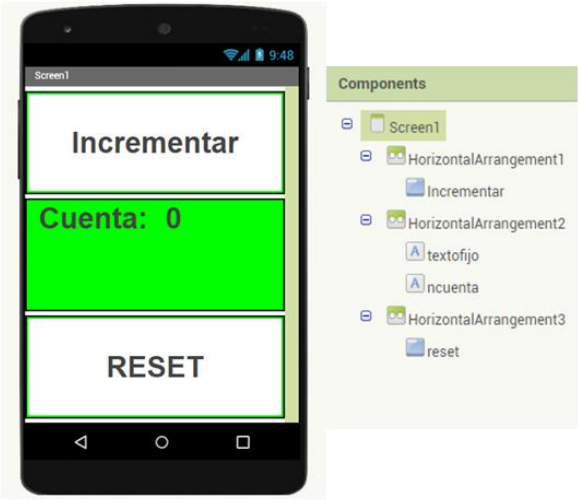

Ilustración 18: Visualización de la aplicación

7.2 Bloques

En esta parte se explica la programación de los componentes seleccionados en el apartado anterior, para que se comporten conforme se había expuesto previamente.

En la parte de bloques, como aparece en la Ilustración 19, se pueden visualizar todas las categorías disponibles de bloques, y seguidamente, los componentes que hayan sido insertados en el diseño.

Si se hace clic sobre cualquiera de ellos, aparecen los bloques, con las funcionalidades que se tienen disponibles, como por ejemplo el botón de incrementar, como en la Ilustración 20.

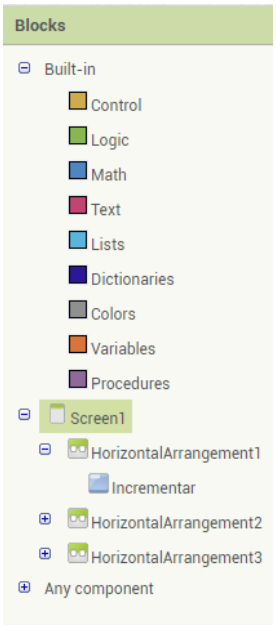

Ilustración 19: Bloques en MIT APP Inventor

Si se pincha sobre cualquiera de las categorías, que son identificables también por su color, aparecen los bloques que contiene la misma. Eso mismo acontece en los elementos del diseño, que permite ver las funcionalidades propias de los mismos, como por ejemplo el botón de incrementar, como muestra la Ilustración 20.

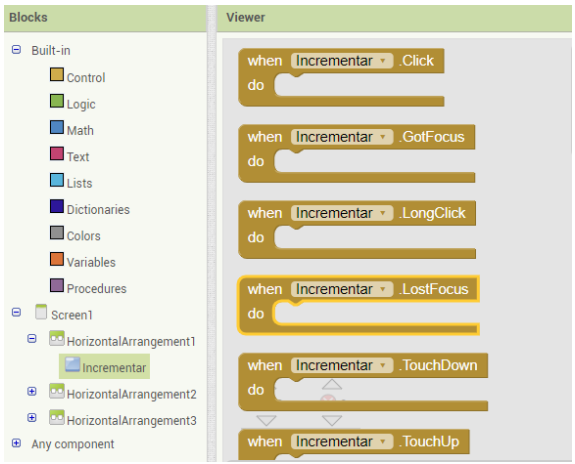

Ilustración 20: Bloques con sus funcionalidades en MIT APP Inventor

Para programar la aplicación, lo primero que se debe hacer es analizar las variables o estructuras de control que se necesita.

En este caso solo se precisa de una variable, que almacene la cuenta. Para ello se busca dentro de las categorías, las variables, como se muestra en la Ilustración 21, y se selecciona la primera opción, que permite inicializar una variable global, a la cual se le asigna el nombre de cuenta.

Ilustración 21: Variables en MIT APP Inventor

Una vez se haya seleccionado y cambiado su nombre, se escoge la opción de igualar la variable a 0, dentro de la categoría Math, como se muestra en la Ilustración 22.

Ilustración 22: Igualando variable a 0 en MIT APP Inventor

Por consiguiente, se programa lo que debe ocurrir cuando se pulsa el botón de incrementar cuenta, que es sumar uno a la cuenta. Para ello se selecciona el botón incrementar, como aparece en Ilustración 21 y se escoge la primera estructura de control disponible. Dentro de la misma se escoge en la categoría de variables la opción de asignar un valor a la variable, y dentro de la categoría de matemáticas se suma uno a la misma. En la Ilustración 23 aparecen los tres bloques mencionados previamente unidos.

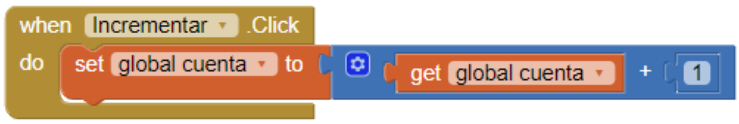

Ilustración 23: Unificando bloques en MIT APP Inventor

Para poder asignar el valor de la cuenta global al cuadro de texto que corresponde, se selecciona el nombre de dicho cuatro (ncuenta), y se escoge la opción que permita asignar al cuadro de texto el valor de dicha variable, como se muestra en la Ilustración 24.

Ilustración 24: Asignando valor a cuadro de texto en MIT APP Inventor

Como se puede observar, este bloque precisa estar sujeto a una estructura de control, que lógicamente debe ser el bloque anterior, puesto que el cuadro de texto debe actualizarse siempre que el botón sea pulsado. De esta manera queda finalizada la programación del botón de incrementar (Ilustración 25).

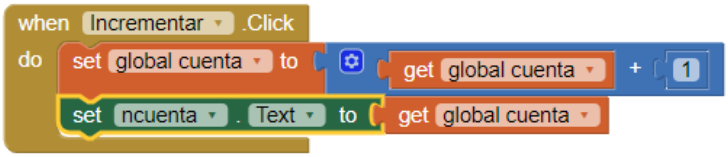

Ilustración 25: Bloque con estructura de control en MIT APP Inventor

Para programar el botón de reset, hay que realizar el mismo proceso que con el botón de incrementar, pero en vez de incrementarse en uno el valor de global cuenta, se le debe asignar el valor de 0, puesto que su funcionalidad es resetear dicha variable, esta programación se muestra en la Ilustración 26 .

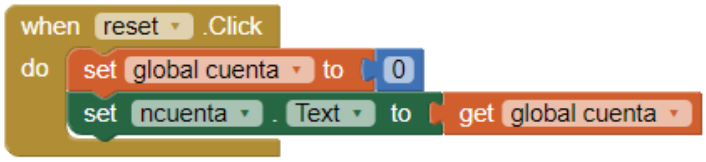

Ilustración 26: Programación del botón Reset en MIT APP Inventor

Una vez terminada toda la programación, se da por finalizado el desarrollo de la aplicación con un resultado final como el que aparece en la Ilustración 27.

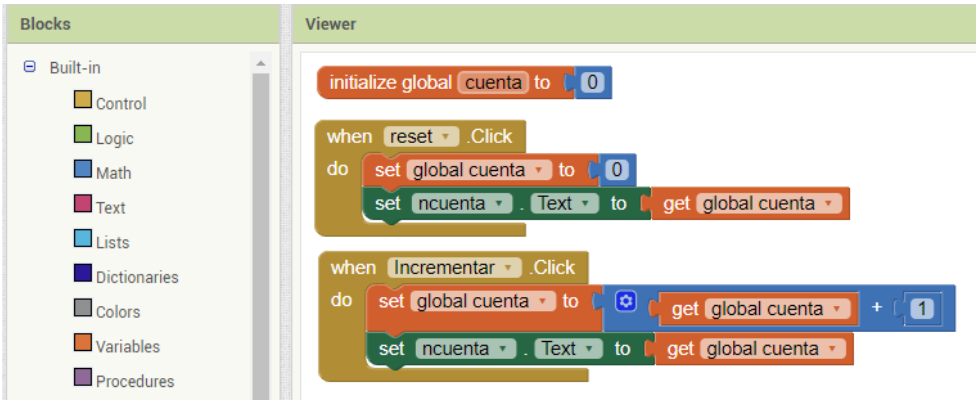

Ilustración 27: Resultado de la programación de la aplicación en MIT APP Inventor

7.3 Descargar la aplicación

Una vez completado el desarrollo de la aplicación se procede a su instalación en el smartphone. Para ello se dispone de dos opciones:

1. **Generar un archivo .apk:** en esta opción se genera un archivo .apk, que se instala en el smartphone para poder mantener la aplicación en el teléfono. Esta opción se suele usar cuando se está seguro de que la aplicación funciona perfectamente.

   Para disponer del archivo .apk, en la parte superior de la página web, se selecciona el apartado de Build como aparece en el cuadrado amarillo de la Ilustración 28.

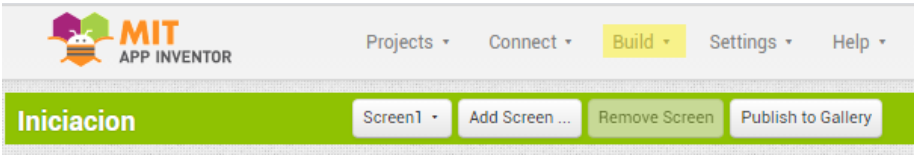

Ilustración 28: Apartado Build en web de MIT APP Inventor

Aquí aparecen dos opciones, como se puede observar en la Ilustración 29, donde se debe escoger la de generar el archivo .apk.

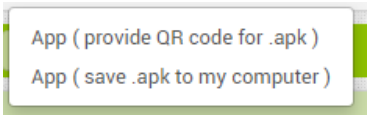

Ilustración 29: Opciones del apartado Build en web MIT APP Inventor

Esta se genera automáticamente, y en la pantalla aparece un código QR, como el que se muestra en la Ilustración 30. A continuación, se abre la aplicación en el smartphone (que debe descargarse en Play Store), se escanea el código, y automáticamente se descargará el archivo .apk con la aplicación desarrollada lista para ser instalada.

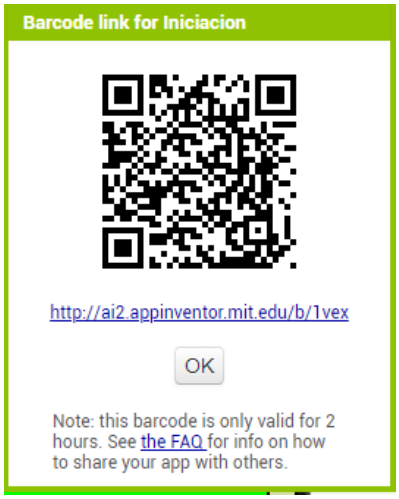

Ilustración 30: Código QR generado por la web de MIT APP Inventor

2. **Visualizar la aplicación momentáneamente:** esta opción se utiliza sobre todo para hacer pruebas de la aplicación, ya que permite utilizarla por un breve periodo de tiempo.

   Para ello, se selecciona la opción de Connect, como aparece en Ilustración 31.

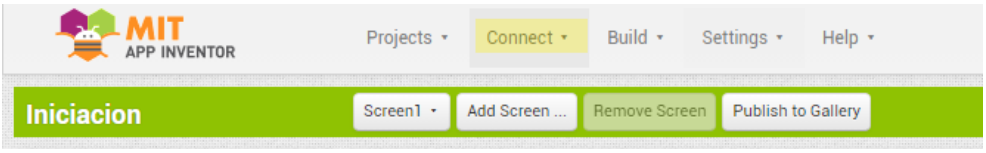

Ilustración 31: Apartado Connect en web de MIT APP Inventor

Aquí se despliegan una lista de opciones y se escoge la primera. Como en la Ilustración 32.

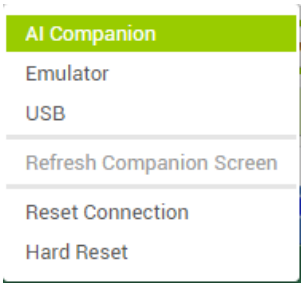

Ilustración 32: Opciones del apartado Connect en web de MIT APP Inventor

Posteriormente debe aparecer un código QR en la pantalla, el cual se escanea con la aplicación de Mit App inventor que debe estar en el teléfono móvil, y así se puede disponer de la aplicación desarrollada por un periodo de tiempo.

8 MATERIALES DE LOS EXPERIMENTOS

8.1 Placa de desarrollo

En la realización de este proyecto y como se ha adelantado en el documento se hace uso de la placa de desarrollo **Arduino UNO Rev3** la cual representa un punto de partida ideal para cualquiera que se quiera iniciar en el mundo de la electrónica digital y los sistemas embebidos.

La tarjeta dispone del microcontrolador ATmega328P a 16MHz el cual cuenta con 14 entradas/salidas digitales de las cuales 6 pueden ser utilizadas como salidas PWM y 6 entradas analógica, además de conexión USB y un conector Jack para la alimentación.

Un motivo de peso para elegir esta tarjeta es su bajo precio y la comunidad de usuarios existente. A pesar de que su precio contenido todavía se puede reducir el costo construyéndola uno mismo[14] o adquiriendo una tarjeta clónica en cualquier web pudiendo conseguirla por menos de 3€.

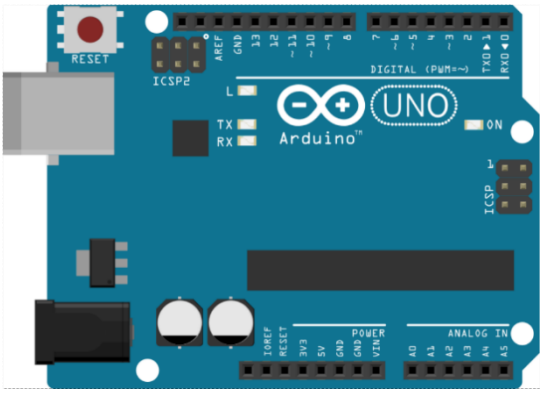

Ilustración 33: Tarjeta Arduino UNO

- **Protoboard**

  Es una placa de pruebas que permite la inserción y conexión de cables y componentes electrónicos, lo que permite montar y probar un diseño sin necesidad de realizar soladuras.

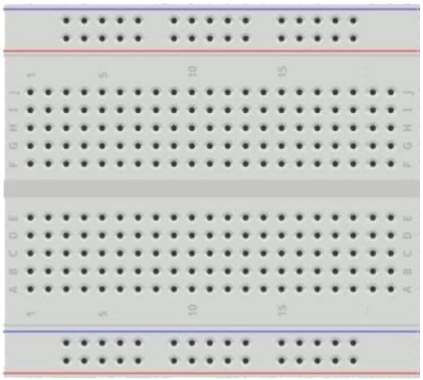

Ilustración 34: Protoboard

8.2 Sensores

En este proyecto se van a utilizar dos sensores, uno de humedad y temperatura, y otro de ultrasonido, su funcionamiento es el siguiente:

- **Sensor DTH22:**[12] es un sensor de temperatura y humedad que puede medir temperaturas entre los -40°C y 80 °C. Consta de 4 pines, el que se conecta a la alimentación (5V), el que se conecta a GND, el pin de dato nulo y el pin por el cual se obtienen los datos a través de un protocolo de comunicación de bus único que solo emplea una línea de transmisión. Es capaz de hacer mediciones cada dos segundos con un error de <±0.5 °C y de 2% RH. En su interior contiene un microprocesador junto con un sensor capacitivo para la humedad y un termistor para la temperatura.

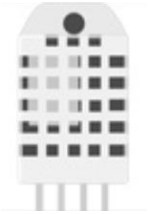

Ilustración 35: DHT22

- **Sensor HC-SR04:** es un sensor de ultrasonido que permite calcular la distancia a la que se encuentra un objeto. Para ello dispone de cuatro pines, el que se conecta a la alimentación (5V), el que se conecta a tierra (GND), el de trigger (envía los pulsos de ultrasonido) y el de echo (recibe el pulso), estos dos últimos se comunican con Arduino a través del protocolo de comunicación serie I2C. Para poder calcular la distancia a la que está el objeto, hay que tener en cuenta: el tiempo de recepción en µs; el recorrido del pulso, tanto de ida como de vuelta; la velocidad del sonido 343 m/s y la duración que tiene el pulso, en este caso 10 ms. [13]

$$v = \frac{distancia\ recorrida}{tiempo} = \frac{343m}{s} * \frac{1s}{1000000us} * \frac{100cm}{1m}$$
$$= \frac{2 * d\ (distancia\ en\ cm)}{t(us)}$$

Con la fórmula anterior se calcula en centímetros por microsegundos la velocidad del sonido, que es la velocidad de la onda emitida por el sensor. Como esta onda recorre aproximadamente la misma distancia desde el sensor al objeto que del objeto al sensor, se obtiene como resultado total, que la velocidad es igual al doble de la distancia al objeto, entre el tiempo que tarda el sensor en recibir la onda.

$$d(distancia\ del\ obejeto) = \frac{v * t}{2} = 0{,}01715 * t$$

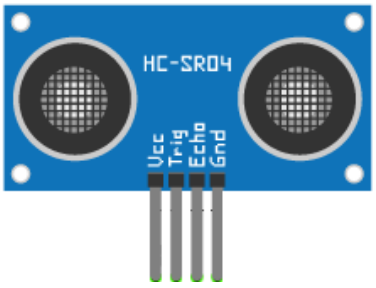

Ilustración 36: HC-SR04

8.3 Módulos

- **HC-06**

Es un módulo Bluetooth diseñado para establecer comunicación inalámbrica de corto alcance. Puede interactuar con casi todos los microcontroladores o procesadores, ya que utiliza la interfaz UART. Este módulo tiene la capacidad de

transmitir datos a una velocidad de hasta 2,1 Mbps y funciona con el protocolo de comunicación Bluetooth 2.0.
Utiliza 4 pines: dos para la alimentación (cc y GND) y dos para la comunicación (RX y TX).

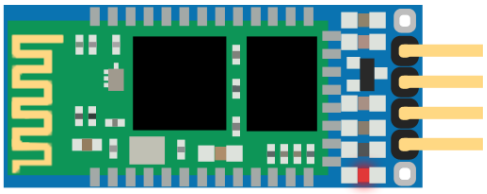

Ilustración 37: HC-06

- **Driver Motores L298N**

  Es un dispositivo que permite controlar tanto la velocidad como el sentido de giro de un motor. El driver L298N utiliza un puente en H para controlar 2 motores, uno de los motores se conecta a los terminales OUT 1 y OUT 2 y el otro a los terminales OUT 3 y OUT 4. Para controlar la velocidad y dirección del primer motor se utilizarán los terminales IN1 e IN2 y para hacer lo propio con el segundo se utilizan los terminales IN3 e IN4. Para elegir el sentido de giro de los motores se alternarán niveles altos y bajos en los terminales de control siempre habiendo un nivel bajo y un nivel alto presentes.

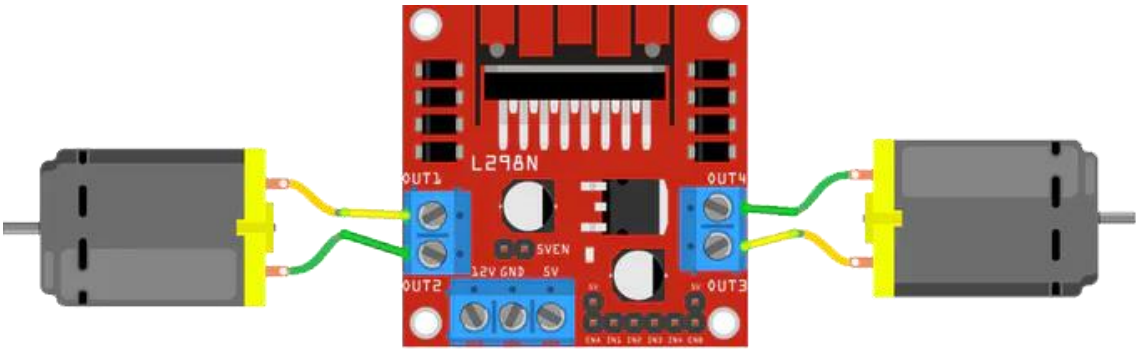

Ilustración 38: L298N

8.4 Alimentación

- **Pilas**

  Estas baterías de referencia IRN18650F1L se encargan de suministrar la energía al driver que controlará los motores, siendo una fuente de energía independiente a la que alimenta la placa de desarrollo.

Ilustración 39: Pilas IRN18650F1L

- **Pila 9v**
  Esta batería de 9v se encarga de alimentar la placa de desarrollo a través de un conector Jack.

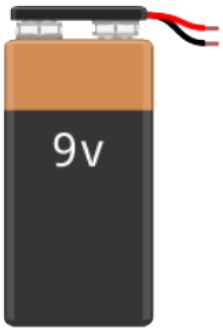

Ilustración 40: Pila 9V

8.5 Mecánica

- **Motores Gebilder**

  Se han elegido unos motores de bajo costo disponibles en cualquier tienda de electrónica. Necesitan una tensión de 3 a 12v para funcionar.

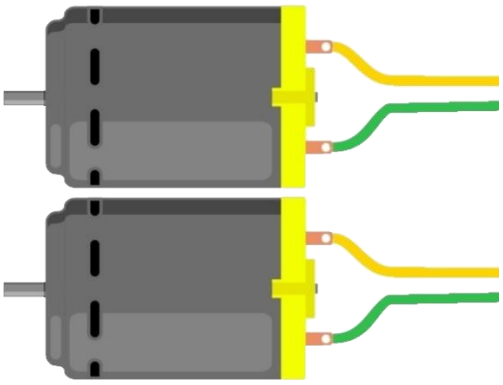

Ilustración 41: Motores Gebilder

- **Portapilas**

  Para situar las pilas en el robot se han elegido unos portapilas independientes que permiten distribuir el peso y el espacio en el robot de la mejor forma posible.

Ilustración 42: Portapilas

- **Ruedas**

  El robot cuenta con dos ruedas, las cuales muchas veces pueden adquirirse junto con los motores, también son de bajo costo y muy fáciles de encontrar en cualquier comercio.

Ilustración 43: Ruedas del robot

- **Rueda giratoria**

  Se ha elegido este tipo de rueda para la parte frontal del robot, no es necesaria, pero puede hacer más sencilla la construcción y facilitar la rotación.

Ilustración 44: Rueda giratoria

9 EXPERIMENTOS

9.1 Experimento 1

En este experimento, se va a iluminar un LED desde una aplicación en un smartphone. Para ello se necesita desarrollar la aplicación móvil, desarrollar el código necesario en Arduino y los siguientes componentes para el montaje físico:

- Placa Arduino.
- Módulo Bluetooth.
- LED.

Mit App Inventor

Para desarrollar la aplicación móvil, primero se realiza la parte de diseño que aparece en la Ilustración 45, consta de dos disposiciones horizontales, la superior en la cual aparece un listpicker, cuya utilidad es la de generar un listado de elementos, en este caso todos los dispositivos Bluetooth que el smartphone tenga emparejados, para poder elegir el del módulo Bluetooth y poder establecer una conexión con la placa de Arduino.

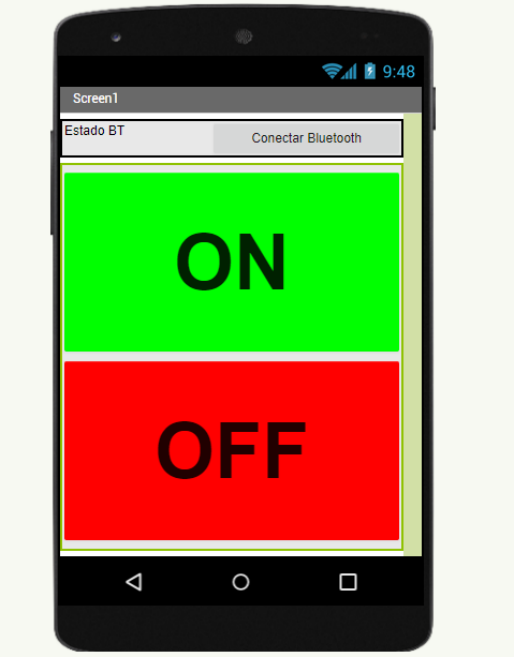

Ilustración 45: Diseño de la aplicación del experimento 1

En cuanto al diseño del módulo de Bluetooth: se dispone de dos cuadros de texto, el de la derecha que se programará para que indique si la conexión se ha establecido o no. Si el dispositivo ha sido correctamente conectado, aparecerá un

“Conectado” en color verde, o si, por lo contrario, ha habido un fallo en la conexión, aparecerá un “No conectado” en rojo.

La segunda disposición consta de dos botones, el de “ON” en color verde, el cual encenderá el LED cuando sea pulsado, y el de “OFF” en rojo, el cual apagará el LED cuando este sea pulsado.

En la parte de bloques, se configura primero la conexión con el dispositivo de Bluetooth como aparece en la Ilustración 46. Cuando el listpicker es pulsado, aparecen todos los elementos con los que está emparejado el smartphone, para escoger el que está vinculado a la placa de Arduino.

En el segundo bloque se programa la conectividad. Si la conectividad es correcta, se manda que en el cuadro de texto aparezca en color verde un “Conectado”. Si ha habido algún fallo en la conexión se escribirá un “No Conectado” en color rojo. Es importante tener en cuenta que esta conexión se va a realizar en todos los experimentos, así que no volverá a ser explicada.

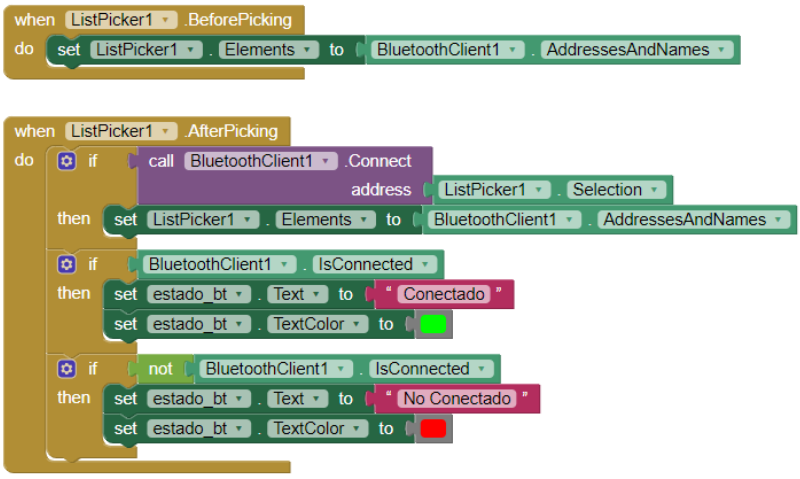

Ilustración 46: Configurando conexión Bluetooth en MIT APP Inventor

En la última parte de bloques de la aplicación, como aparece en la Ilustración 47, se configura el envío de datos por el dispositivo Bluetooth, de manera que cuando es pulsado el botón de ON, se envía el caracter “A”, y se pulsa el botón OFF se envia el carácter “B”.

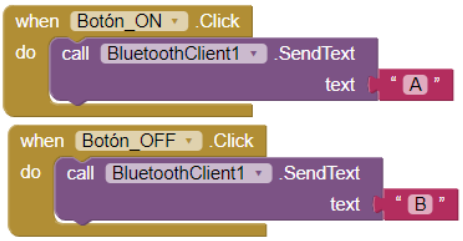

Ilustración 47: Envío de datos a través de Bluetooth en MIT APP Inventor

Arduino

En la programación de Arduino, la cual aparece en la imagen 4, se debe incluir la librería SoftwareSerial, para poder comunicar la placa con el smartphone a través de Bluetooth. Para ello se declara el pin 10 como receptor y el 11 como emisor. Posteriormente se inicializa el dispositivo Bluetooth y se declara el pin 13 de la placa como output para poder mandar información al LED.

En la parte del loop se comprueba si el dispositivo Bluetooth está conectado y por tanto disponible para recibir datos. Si es así, se inicializa la variable inv para que tome los valores que se envían desde el smartphone. Si en esta adquiere el valor de "A", se enciende el led, si se adquiere el valor "B", se apaga.

```
#include <SoftwareSerial.h>
char inv = 0;
SoftwareSerial miBT(10,11);

void setup(){
      miBT.begin(38400);
      pinMode(13, OUTPUT);
}

void loop(){
      if(miBT.available() > 0)
      {
            inv = miBT.read();
            if(inv == 'A')
            {digitalWrite(13, HIGH);}
            if(inv == 'B')
            {digitalWrite(13, LOW);}
      }
  }
```

Montaje

Para realizar el montaje es necesario emplear una resistencia para el led, ya que si se le proporciona directamente 3.3V (voltaje que proporcionan los pines de salida en el modo HIGH del arduino), se puede fundir. En este caso el valor va a ser de 200 Ω. El extremo de esta resistencia se conectará a el pin 13 de la placa de Arduino, mientras que el otro extremo se conecta al ánodo del led, (en serie), por otro lado se conecta el cátodo del led a tierra. Como aparece en la Ilustración 48.

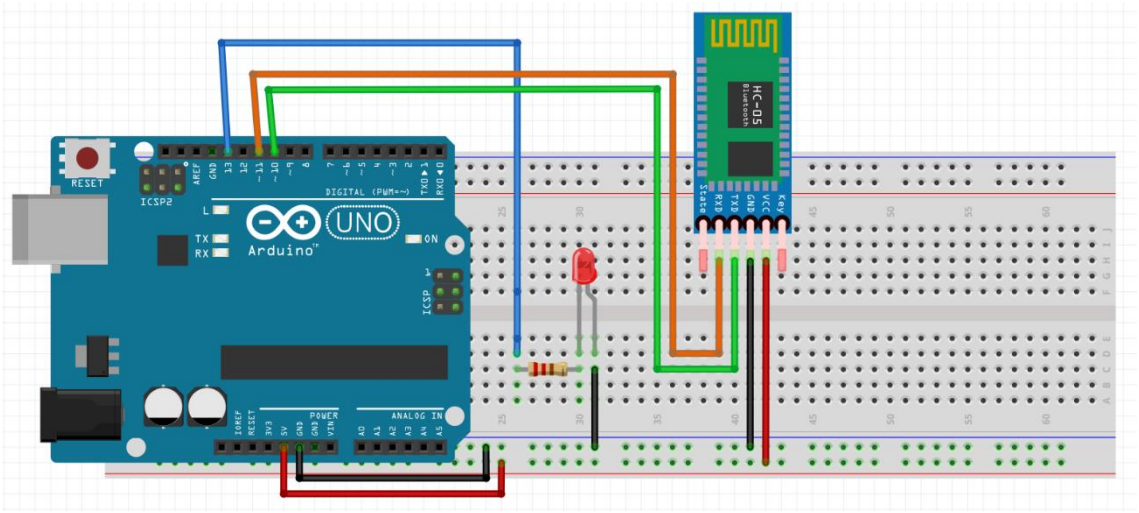

Ilustración 48: Montaje experimento 1

9.2 *Experimento 2*

Este experimento consiste en visualizar en el smartphone la temperatura y la humedad del ambiente, así como la distancia a la que se encuentra cualquier objeto de la placa de Arduino. Para ello se necesitan aparte del código en Arduino, y de la aplicación desarrollada en Mit App Inventor, los siguientes componentes:

- Placa Arduino.
- Módulo Bluetooth.
- Sensor de temperatura y humedad.
- Sensor de ultrasonido.
- Protoboard.

Mit App Inventor

La parte de diseño que aparece en la Ilustración 49, consta de cinco disposiciones horizontales, la superior en la cual aparece un listpick para poder conectar la aplicación con el dispositivo de Bluetooth, como se ha explicado en el experimento anterior. La segunda disposición consta de la imagen de un termómetro, de un campo de texto donde aparecerá el valor de la temperatura, y otro campo de texto con la unidad de medición, en este caso grados centígrados. La tercera disposición consta de la imagen de una gota, de un campo de texto donde aparecerá el valor de la humedad, y otro campo de texto con el porcentaje. Y, por último, la cuarta disposición consta de la imagen de dos objetos, de un campo de texto donde aparecerá el valor de la distancia al obstáculo, y otro campo de texto con la unidad de medición, en este caso centímetros.

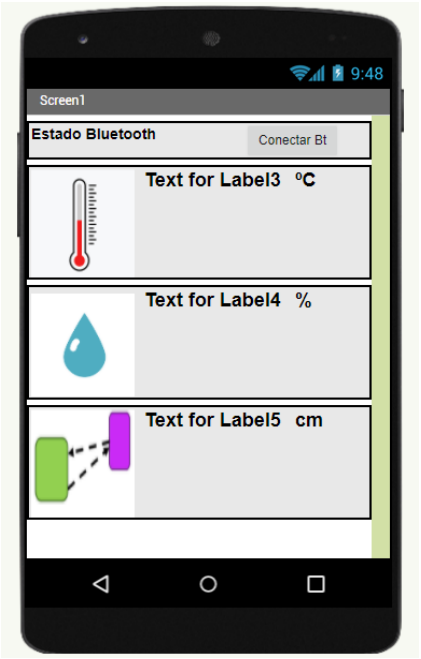

Ilustración 49: Diseño de la aplicación móvil del experimento 2

En la parte de bloques, se configura primero la conexión con el dispositivo de Bluetooth como aparece explicado en el apartado 9.1, a continuación, como aparece en la Ilustración 50, se crea una variable llamada input, la cual recibe los datos que llegan desde el módulo de Bluetooth y se crea una lista vacía. El programa se engloba dentro del clock2, para que cuando termine de recorrerlo, se inicie otra vez. Su funcionamiento es el siguiente: al inicio se comprueba que el Bluetooth esté disponible y pueda enviarnos datos, después ordena que cuando recibamos la cadena de caracteres, siempre que esta contenga una "I", haga una separación de datos para meterlos en la lista, y, por último, ordena la lista con los datos enviados.

Un ejemplo del procedimiento sería el siguiente: se recibe la siguiente línea de caracteres: 25I32I8IA; el programa lo separa en: dato uno (temperatura) 25, dato dos (humedad) 32, dato tres (distancia) 8; y por último el dato cuatro A que se utiliza para confirmar que los datos han sido recibidos en el orden correcto, y la cadena de caracteres no se ha distorsionado.

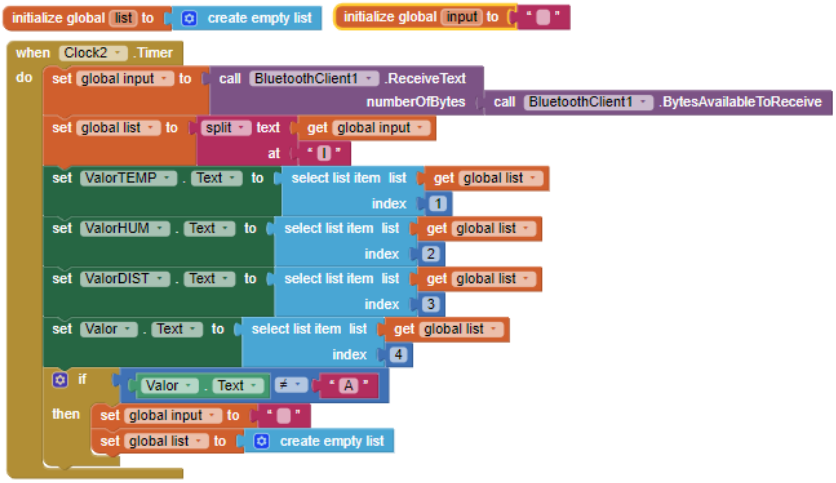

Ilustración 50: Recepción de datos a través de Bluetooth en MIT APP Inventor

Arduino

En la programación de Arduino, la cual aparece en el siguiente cuadro de texto, aparte de la conexión Bluetooth (la cual está ya explicada en el punto 9.1), se incluye la librería DTH, para poder leer el sensor de humedad. A continuación, se especifica el tipo de sensor que se va a utilizar (DTH22) y el pin por el que se leen los datos (pin 3). Respecto al sensor de ultrasonido, se declaran los pines necesarios para ejecutar el de disparo (trigger) y para recibir el rebote de dicho disparo, el receptor (echo). Posteriormente se inicializa el dispositivo Bluetooth y ambos sensores.

```
#include <DHT.h>
#include <SoftwareSerial.h>

const int DHTPin = 3;

#define DHTTYPE = 3;
DHT dht(DHTPin, DHTTYPE);

SoftwareSerial miBT(10,11);
const int Trigger = 4;
const int Echo = 5;

void setup(){
      Serial.begin(9600);
      miBT.begin(38400);
      dht.begin();
      pinMode(13, OUTPUT);
      pinMode(Trigger, UTPUT);
      pinMode(Echo, INPUT);
      digitalWrite(Trigger, LOW)
  }
```

En el siguiente cuadro de texto, aparece la continuación del código, la parte del loop, en la cual se leen los sensores, y se genera la cadena de caracteres previamente explicada en el desarrollo de la aplicación en Mit App Inventor. La fór-

mula que aparece para poder generar la distancia al objeto está explicada en el punto 8.1, en la parte del sensor de ultrasonido, siendo s el tiempo en microsegundos en que tarda en recibirse la onda. Por último, se realiza una espera de 0,2 segundos, y vuelve a empezar.

```
void loop()
{
  float t;
  float h;
  long s;
  long d;

  digitalWrite(Trigger, LOW);
  delayMicroseconds(4);
  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);
  s = pulseIn(Echo, HIGH);
  d = s*0.01715;

  t= dht.readTemperature();
  h= dht.readHumidity();

miBT.println(t+String("I")+h+String("I")+d+String("I")+String("A"));
  delay(200);

    }
```

Montaje

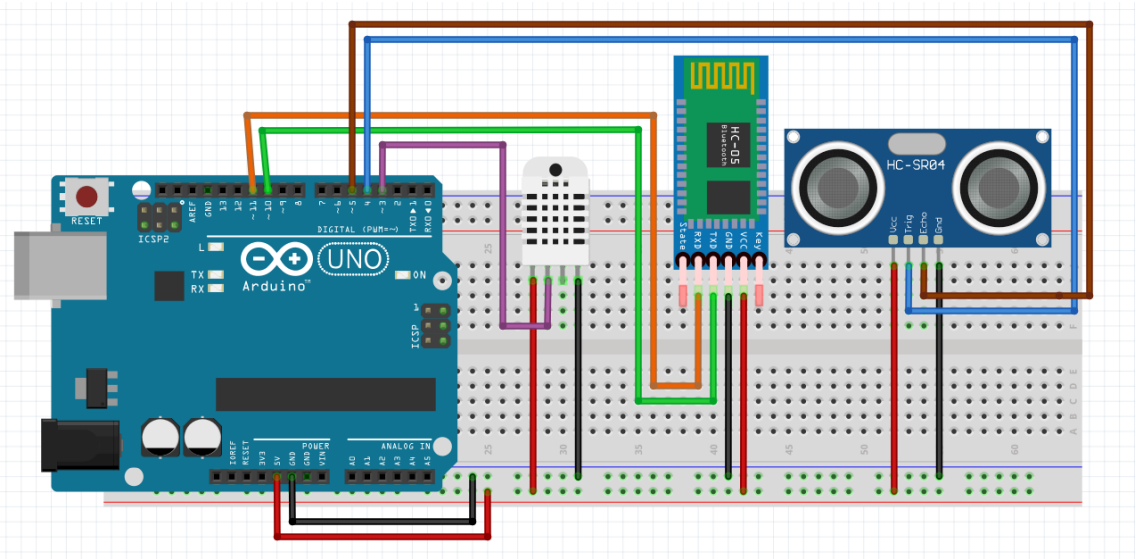

Ilustración 51: Montaje experimento 2

9.3 Experimento 3

Este experimento consiste en mover el robot, que ha sido diseñado previamente en FreeCAD, a través de la aplicación móvil. Para ello se necesitan a parte del código en Arduino, y de la aplicación desarrollada en Mitt App Inventor, los siguientes componentes:

- Placa Arduino.
- Módulo Bluetooth.
- Chasis del Robot.
- Rueda giratoria.
- Dos motores junto con las ruedas acopladas.
- Batería 9 V para alimentar la placa de Arduino.
- Dos pilas 3,6V con sus adaptadores.
- Puente en H para los motores.
- Protoboard.

Mit App Inventor

La parte de diseño que aparece en la Ilustración 52, consta de cuatro disposiciones horizontales, la superior en la cual aparece un listpick para poder conectar la aplicación con el dispositivo de Bluetooth, como se ha explicado en el primer experimento. La segunda disposición consta de un único botón con la palabra UP (arriba en inglés) que se empleará para mover hacia delante el robot. La tercera disposición consta de tres botones el izquierdo con un L (Left), que se va a utilizar para mover el robot hacia la izquierda; el del medio que aparece vacío, cuya función será la de parar el robot; y el de la derecha, con una R (Right), que se empleará para mover el robot hacia la derecha. Por último, la cuarta disposición consta de un único botón con la palabra DOWN, para poder mover el robot hacía atrás.

Ilustración 52: Aplicación experimento 3

En la parte de bloques, primero se configura la conexión con el dispositivo de Bluetooth como aparece explicado en el apartado 9.1. A continuación, como aparece en la Ilustración 53, se configuran el resto de botones de tal manera que: cuando se pulse el botón UP, una 'U' sea enviada al dispositivo Bluetooth, y así con el resto de botones, el botón Left envía una 'L', el de Down envía una 'D', el botón Right una 'R' y por último el botón Quieto una 'S'.

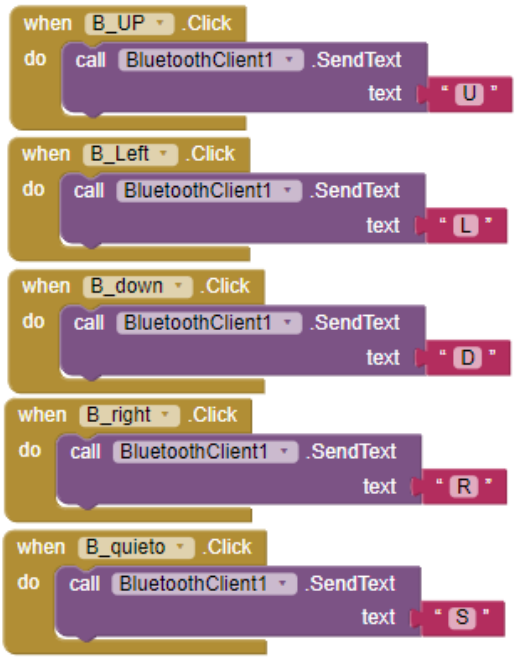

Ilustración 53: Configurción de botones en MIT APP Inventor

Arduino

Para la implementación del código en Arduino, solo se ha incluido la librería de Software Serial, para la conexión del Bluetooth, la cual ya ha sido explicada en el punto 9.1. Además, para controlar los dos motores se necesitan cuatro pines como salidas, dos para el motor A, y dos para el motor B como aparece en el siguiente cuadro de texto.

```
#include <SoftwareSerial.h>
SoftwareSerial miBT(10, 11);

const int mad=5;
const int mai=6;
const int mbd=7;
const int mbi=8;
int inv = 0;

void setup() {
  Serial.begin(9600);
  miBT.begin(38400);
  pinMode(mad,OUTPUT);
  pinMode(mai,OUTPUT);
  pinMode(mbd,OUTPUT);
  pinMode(mbi,OUTPUT);

}
```

Para hacer el código más sencillo y sintetizado, se ha decido implementar cinco funciones, una por cada movimiento. En el siguiente cuadro de texto aparece la función ‘up’, en la se ponen a nivel alto el pin del motor ‘B’ de girar a la derecha, y el pin del motor ‘A’ de girar a la izquierda, para que ambas ruedas giren hacia delante.

```
int up(){
  digitalWrite(mad,LOW);
  digitalWrite(mbd,HIGH);
  digitalWrite(mai,HIGH);
  digitalWrite(mbi,LOW);
 }
```

En el siguiente cuadro de texto aparece la función ‘down’, cuya implementación es la opuesta a la función de ‘up’, puesto que ahora se pretende mover el robot en dirección opuesta.

```
int down(){
  digitalWrite(mad,HIGH);
  digitalWrite(mbd,LOW);
  digitalWrite(mai,LOW);
  digitalWrite(mbi,HIGH);
 }
```

En el siguiente cuadro de texto se muestra el código de la función ‘left’, cuya finalidad es que el gire hacia la izquierda, para ello cada rueda debe de girar en un sentido diferente, y por la orientación de los motores en el robot, se pone a nivel alto el pin del motor A de girar a la izquierda, y el del motor B, de girar a la izquierda también, ya que los motores están en modo espejo, uno frente al otro.

```
int left(){
  digitalWrite(mad,LOW);
  digitalWrite(mbd,LOW);
  digitalWrite(mai,HIGH);
  digitalWrite(mbi,HIGH);
 }
```

En el siguiente cuadro de texto, aparece la implementación de la función ‘right’, la cual, lógicamente, debe ser la opuesta a la función ‘left’.

```
int right(){
  digitalWrite(mad,HIGH);
  digitalWrite(mbd,HIGH);
  digitalWrite(mai,LOW);
  digitalWrite(mbi,LOW);
 }
```

La última función que aparece en el código es la de ‘quieto’, en la cual, ambos motores deben permanecer en reposo. Por tanto, se pondrán a nivel bajo todas las salidas.

```
int quieto(){
  digitalWrite(mad,LOW);
  digitalWrite(mbd,LOW);
  digitalWrite(mai,LOW);
  digitalWrite(mbi,LOW);
 }
```

Por último, se va a explicar la parte del bucle infinito de Arduino, en la cual se gestionan las funciones previamente mencionadas. En el siguiente cuadro de texto, aparece el código cuya funcionalidad es la siguiente: primero se comprueba si el dispositivo de Bluetooth está disponible para recibir datos. Si es así, guarda en la variable ‘inv’, el dato recibido por Bluetooth (que ha sido enviado a través de la aplicación), y con la estructura de control *switch-case*, verifica qué orden debe ejecutarse, y llama a la función correspondiente. Es decir, si desde la aplicación se pulsa el botón de UP, la variable ‘inv’ leerá una U, y por tanto, se ejecutando la función ‘up’.

```
void loop() {
  if (miBT.available()>0){
  inv=miBT.read();
  switch (inv)
  {
    case 'U':
      up();
    break;

    case 'L':
      left();
    break;

    case 'R':
      right();
    break;

    case 'D':
      down();
    break;

    case 'S':
      quieto();
    break;
  }

  }
}
```

9.4 Experimento 4

Este experimento consiste en mover el robot, a través de la aplicación móvil y a su vez visualizar la temperatura y humedad ambiente y la distancia a la que se encuentra cualquier objeto. Para ello se necesitan aparte del código en Arduino, y de la aplicación desarrollada en Mit App Inventor, los siguientes componentes:

- Placa Arduino.
- Módulo Bluetooth.
- Chasis del Robot.
- Rueda giratoria.
- Dos motores junto con las ruedas acopladas.
- Batería 9 V para alimentar la placa de Arduino.
- Dos pilas 3,6V con sus adaptadores.
- Puente en H para los motores.
- Sensor de temperatura y humedad.
- Sensor de ultrasonido.
- Protoboard.

Mit App Inventor

Para el diseño de la aplicación, como se puede observar en la Ilustración 54 se ha utilizado el mismo formato que en el experimento anterior, pero añadiendo otra disposición horizontal donde se visualiza la temperatura, humedad y distancia. Para ello se han incluido seis cuadros de texto. Empezando de izquierda a derecha, primero se encuentra un cuadro de texto con una "T" para indicar que en el siguiente cuadro de texto va a aparecer el valor de temperatura. En el cuadro contiguo se muestra una "H", para indicar que en el siguiente cuadro se muestra el porcentaje de humedad en el ambiente. Por último, en el quinto cuadro hay escrito "cm", para indicar que se va a mostrar la distancia a la que se encuentre cualquier objeto en centímetros.

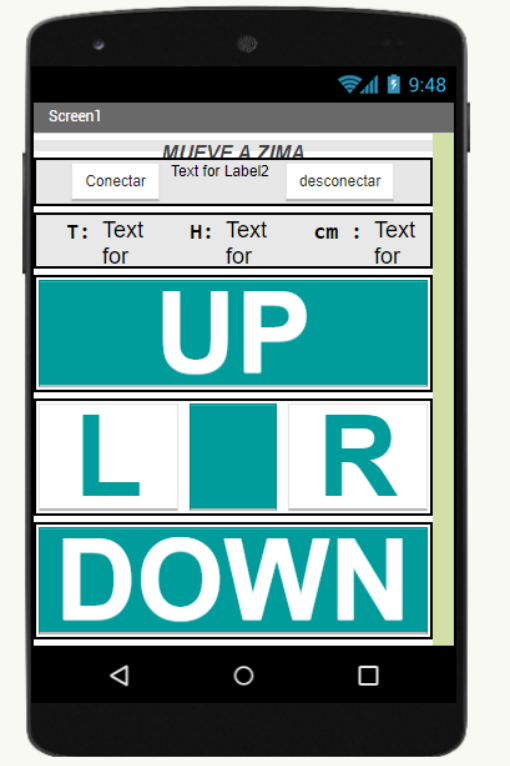

Ilustración 54: Aplicación Experimento 4

En la parte de bloques, la conexión bluetooth, se ha implementado como en todos los experimentos anteriores, y la configuración de los botones como en el experimento 3.

Para poder visualizar los datos de los sensores, se sigue el mismo procedimiento que en el experimento 2. Primero, como aparece en la Ilustración 55, se inicializan las variables. La variable input, la cual almacena la cadena de caracteres que se recibe desde el bluetooth; a continuación, una lista vacía, para separar dicha cadena; y por último, tres variables, una por cada dato que se obtiene de la cadena. Primero el dato comprobar, que se va a encargar de identificar que cadena recibo, si la del sensor de temperatura y humedad o la del ultrasonido; después la variable primero, que almacena el segundo dato de la cadena, que en este caso

será el primero con el contenido de uno de los sensores; y el segundo, que dependiendo de que cadena reciba almacenará un dato del sensor, o simplemente un dato (el por qué se explica en la parte de Arduino).

Ilustración 55: Inicialización de variables

Para gestionar los datos recibidos e identificar a qué sensor pertenece la cadena de caracteres, después de crear la lista con los datos, como se ha explicado previamente en el experimento 2, se comprueba el primer dato, almacenado en la variable comprueba, si este es igual que 'T', los siguientes datos son la temperatura y la humedad, mientras que si comprueba vale 'C', el siguiente dato serán los centímetros a los que se encuentra cualquier objeto. Esto aparece en la Ilustración 56.

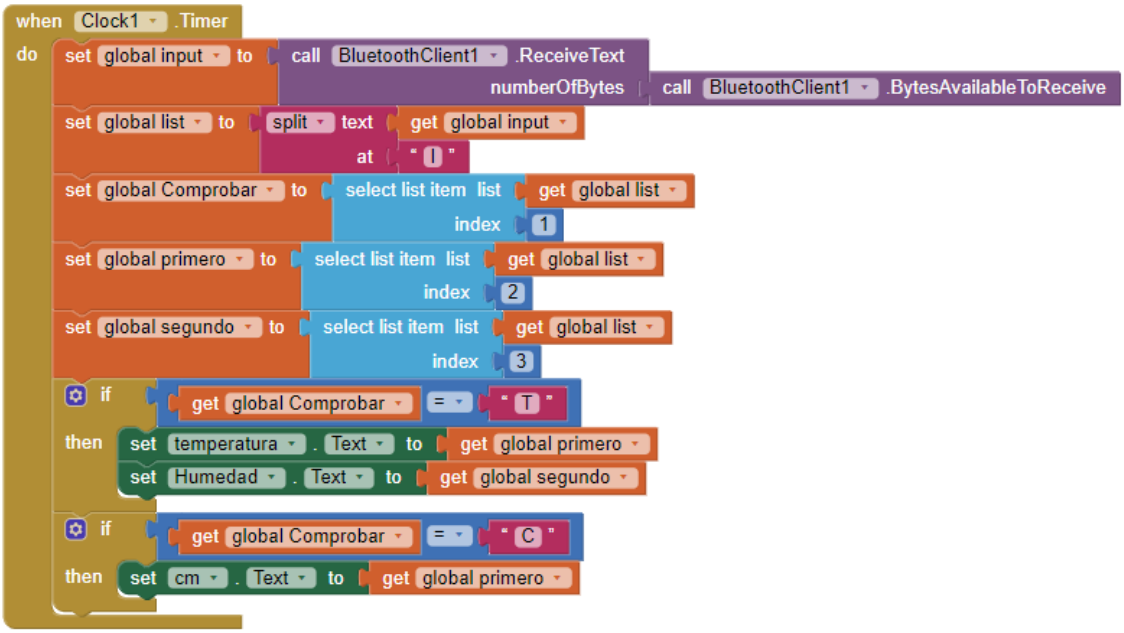

Ilustración 56: Gestión de datos recibidos en MIT APP Inventor

Arduino

Para implementar el código de Arduino, se ha tomado como base el código del experimento anterior, al cual se le ha añadido tanto la librería del sensor de temperatura y humedad, como la de TimerOne, la cual se va a utilizar para a través de la interrupción 1 que tiene Arduino. A continuación, aparecen todas las variables que se deben declarar las cuales han sido explicadas en los experimentos anteriores.

```
#include <TimerOne.h>
#include <DHT.h>
#include <SoftwareSerial.h>

const int DHTPin = 2;
#define DHTTYPE DHT22
DHT dht(DHTPin, DHTTYPE);

SoftwareSerial miBT(10, 11);

const int Trigger = 4;
const int Echo = 3;

const int mad=5;
const int mai=6;
const int mbd=7;
const int mbi=8;

float t;
float h;
long s;
long d;

int inv = 0;
int cnt=0;
int temhum();
int ultrasonido();
```

En la parte de setup, además de inicializar tanto los sensores como el módulo de Bluetooth, se inicializa el timer uno, a una frecuencia de 100 ms, y se asigna a la función ISR_timer como aparece en el siguiente código.

```
void setup() {
  dht.begin();
  Timer1.initialize(100000);
  Timer1.attachInterrupt(ISR_timer);
  Serial.begin(9600);
  miBT.begin(38400);
  pinMode(mad,OUTPUT);
  pinMode(mai,OUTPUT);
  pinMode(mbd,OUTPUT);
  pinMode(mbi,OUTPUT);
  pinMode(Trigger, OUTPUT);
  pinMode(Echo, INPUT);
  digitalWrite(Trigger, LOW);
}
```

Para terminar de completar el código, además de todas las funciones de movimiento explicadas en el experimento anterior, se han añadido otras tres.

La función del sensor de temperatura y humedad, que lee los datos del sensor, y envía por Bluetooth la cadena de caracteres correspondiente.

```
int temhum(){
  t= dht.readTemperature();
  h= dht.readHumidity();
  miBT.println(String("TI")+t+String("I")+h);
}
```

La función del sensor de ultrasonido, que interpreta el dato recibido para pasarlo a centímetros, y envía la cadena de caracteres consecuente. Si la distancia a la que se encuentre el sensor es menor a 8cm, se llama a la función ‘down’, para que el robot retroceda.

```
int ultrasonido(){

  digitalWrite(Trigger, LOW);
  delayMicroseconds(4);
  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);
  s = pulseIn(Echo, HIGH);
  d = s/59;
  miBT.println(String("CI")+d+String("In"));
  if(d<8){
    down();
  }
}
```

La función de la interrupción, que simplemente incrementa un contador.

```
void ISR_timer()
{
  cnt++;
}
```

Para finalizar el código, se ha añadido la llamada a las funciones de los sensores. Como la temperatura y humedad ambiente varía muy poco en el tiempo, la llamada a su función se realizará cada vez que el contador del timer llegue a 10, es decir, cada segundo; mientras que de la distancia a los objetos se debe tomar datos con mayor frecuencia para evitar que el robot colisione. Por eso su lectura se realiza cada 0.1 segundos.

```
void loop() {
  if (miBT.available()>0)
  {
     if(cnt==10){
     cnt=0;
      temhum();
     }
     else{
       ultrasonido();
     }
    inv=miBT.read();
    switch (inv)
    {
      case 'U':
        up();
      break;

      case 'L':
        left();
      break;

      case 'R':
        right();
      break;

      case 'D':
        down();
      break;

      case 'S':
        quieto();
      break;
    }
  }
}
```

Montaje y resultados

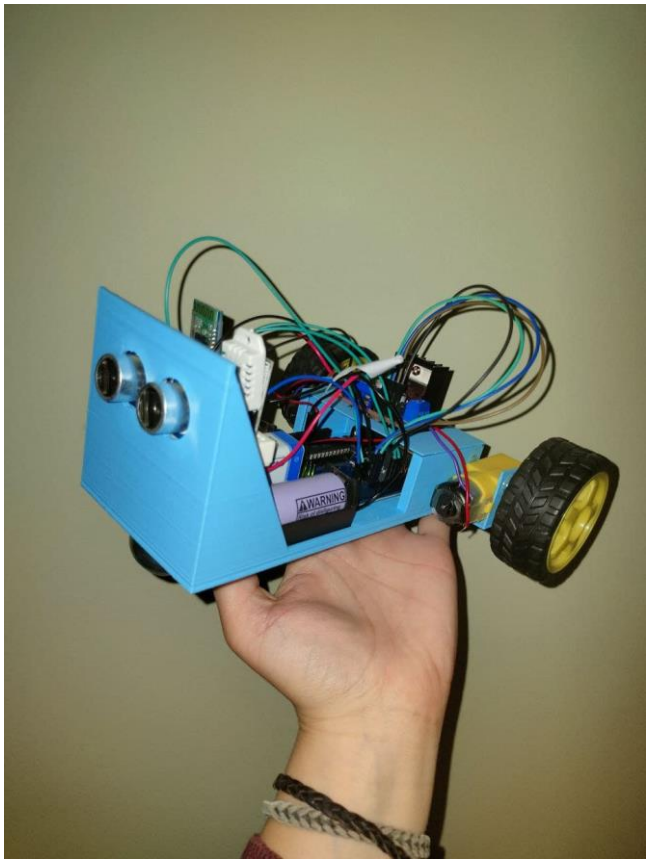

Ilustración 57: Montaje Experimento 4

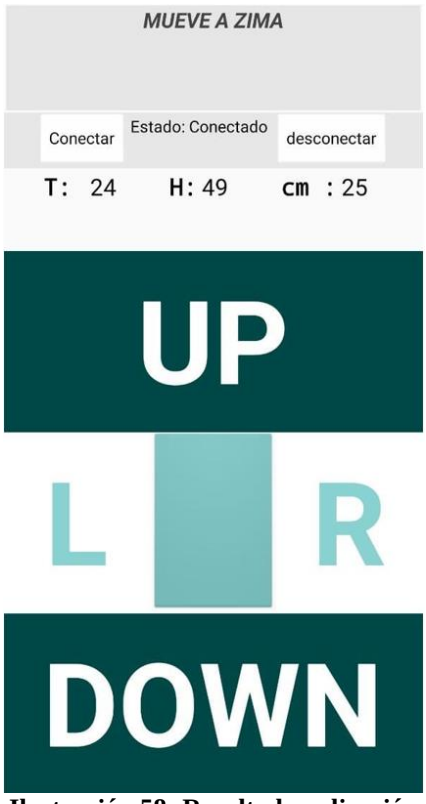

Ilustración 58: Resultado aplicación

10 ANEXOS

10.1 Chasis del Robot

Para implementar el chasis del robot, se ha decidido diseñarlo en 3D con la herramienta de FreeCAD, debido a que es una herramienta de código libre, y por lo tanto, hay mucha información, lo cual hace que su aprendizaje sea más sencillo.

Además, esto permite tener un diseño personalizado, donde todos los elementos necesarios para implementar el robot tienen su espacio definido y los costos de imprimir un diseño ya creado son mucho menores que comprar un chasis prefabricado.

A continuación, se va a explicar las partes que lo componen, al igual que el lugar que debe ocupar cada uno de los elementos del robot.

Se va a comenzar por el alzado, el cual aparece en la Ilustración 59. En él se pueden distinguir dos perforaciones con forma circular detrás de los cuales se va a ubicar el sensor de ultrasonidos. Estos orificios van a permitir que el sensor lance el disparo y lo reciba, para poder calcular la distancia a la que se encuentra cualquier objeto.

En la parte inferior de los orificios, se encuentran dos pilares que van a dar soporte a la mini protoboard, mientras que los pilares contiguos, localizados en la parte trasera del chasis van a funcionar como soporte de otra pieza que se explicará posteriormente.

Por último, en el alzado, en los laterales del chasis, se ubican los soportes para los motores.

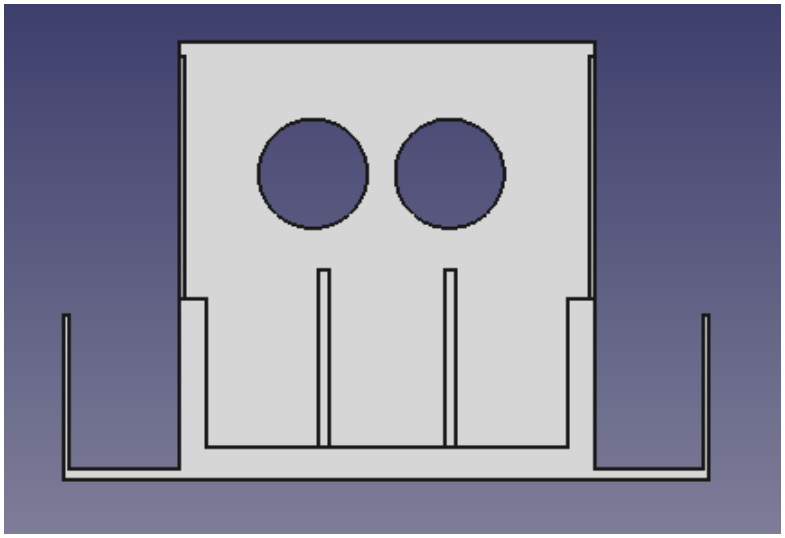

Ilustración 59: Alzado del chasis del robot

En la Ilustración 59, se muestra la planta del diseño. Si se describe de izquierda a derecha, lo primero que se observa son dos rectángulos en color gris, y dentro, dos agujeros. Estos últimos se van a utilizar para insertar los tornillos que sujetan la rueda giratoria, mientras que, en los rectángulos, se van a ubicar dos portapilas, sobre las que obviamente se van a colocar las pilas que actuarán de alimentación para el puente en H.

Entre los rectángulos mencionados previamente, se encuentra otro, también de color gris, pero de menor dimensión, sobre el cual se depositará la batería que alimenta, a través del Jack, la tarjeta de Arduino.

A continuación de los rectángulos, se encuentra un cuadrado con un hueco en el centro, donde se va a ubicar la tarjeta de Arduino. El orificio se ha implementado simplemente para reducir al máximo el gasto en material a la hora de imprimir el chasis.

Por último, en la Ilustración 60 se pueden ver en los extremos los soportes de los motores que han sido ya mencionados la Ilustración 59.

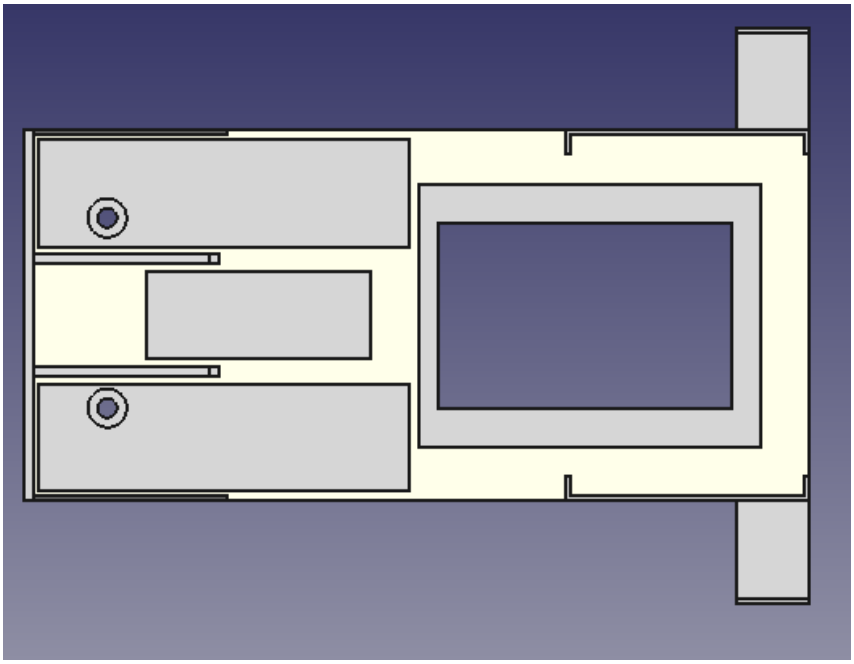

Ilustración 60: Planta del chasis del robot

En la Ilustración 61, aparece el perfil. A la izquierda se pueden intuir los soportes de la mini protoboard, que se localizan detrás de las paredes del chasis. Por otro lado, en el lado opuesto se observa el soporte de la pieza secundaria, y delante, el de los motores. Ambos tienen una perforación con forma ovalada, que permite atornillar tanto los motores, como la pieza secundaria para proporcionar una sujeción adecuada.

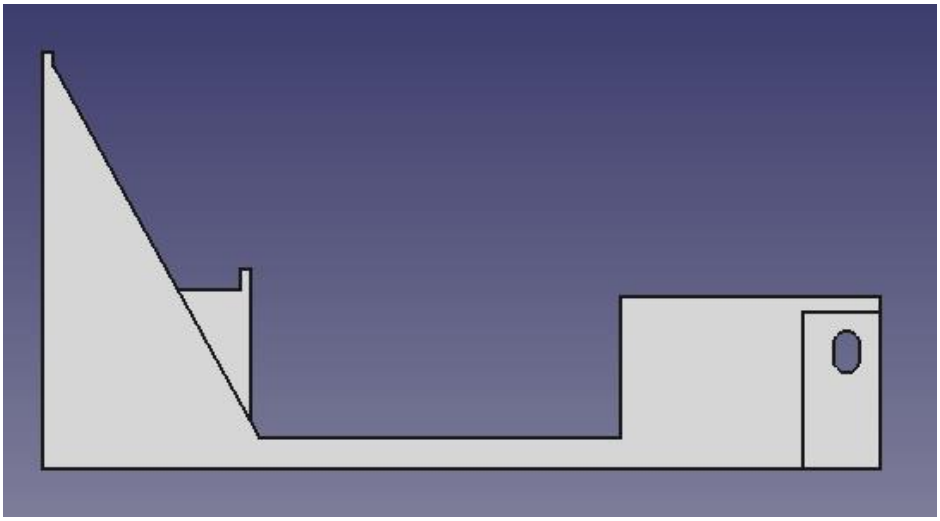

Ilustración 61: Perfil del chasis del robot

Por último, aparece en la Ilustración 62, en la cual se puede observar el mismo orificio que en los soportes del motor, ya que se va a sujetar con el mismo tornillo.

En la parte superior, hay una perforación con forma cuadrada, sobre la cual se ubica el puente en H.

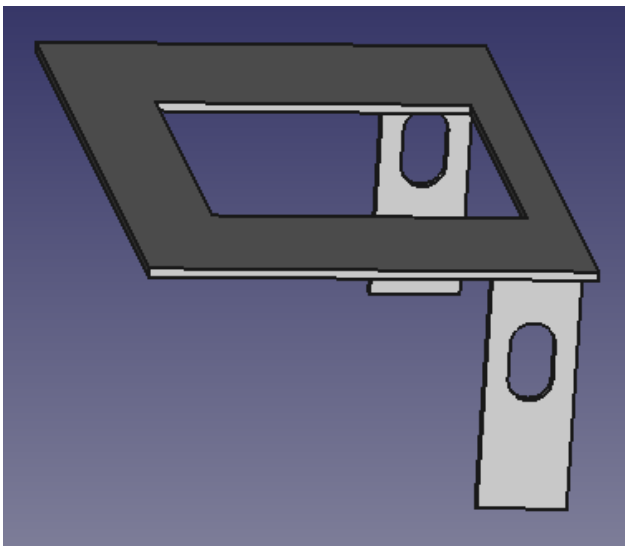

Ilustración 62: Pieza secundaria del chasis del robot

Para finalizar con el anexo, se va a incluir la imagen de ambas piezas desde una perspectiva que permita ver en su conjunto, el diseño completo del chasis (Ilustración 63)

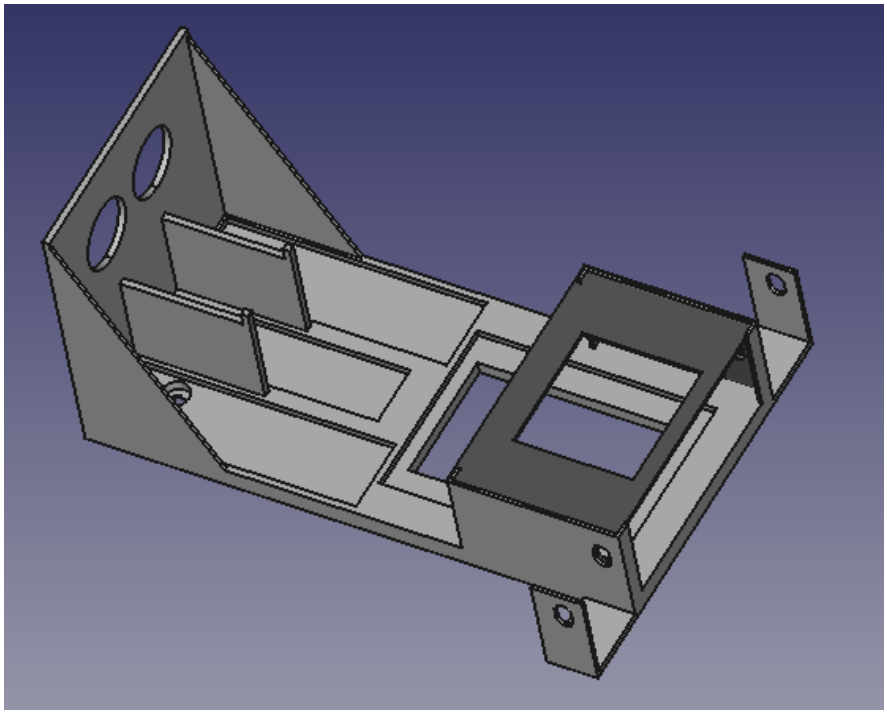

Ilustración 63: Chasis del robot completo

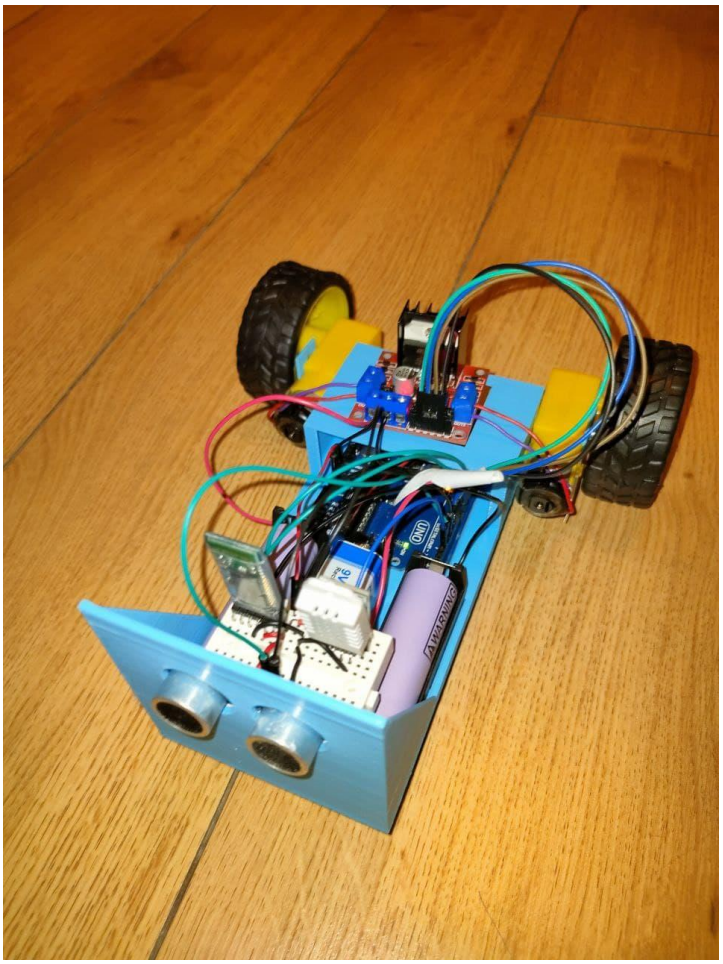

Ilustración 64:Montaje completo

11 CONCLUSIONES

El desarrollo de este proyecto ha servido para afianzar conocimientos básicos y trabajar una parte la didáctica, campo que no tiene mucho peso en los grados de ingeniería. Al tratar de exponer conceptos del mundo de la electrónica a alguien sin la base de conocimiento necesaria, se pone en relieve el nivel de especialización existente en este campo.

A pesar de ello, algo a destacar por encima de todo lo demás es la accesibilidad que el mundo de la electrónica presenta hoy en día y como después de la aparición de Internet han ido surgiendo comunidades y herramientas que permiten afrontar problemas muy técnicos o especializados a personas al margen de este mundo, buenos ejemplos de este son el diseño e impresión 3D o la democratización del mundo de los microcontroladores tras la aparición de Arduino.

La utilización de estas herramientas, que pertenecen a la filosofía de código abierto, ha permitido desarrollar y orientar este trabajo con el fin de realizar un proyecto que atrajese la atención del alumnado.

El uso en conjunto de todos los materiales y herramientas empleados en este proyecto, han permitido exponer conceptos básicos y útiles que puedan servir como un punto de entrada a la materia y que habiliten al alumno para desarrollar proyectos más avanzados y comprobar cómo puede ampliar sus conocimientos.

12 PLIEGO DECONDICIONES

Para la ejecución de este proyecto se ha precisado de los siguientes materiales y plataformas de desarrollo:

Materiales

- Ordenador con sistema operativo Windows 10.
- Tarjeta Arduino UNO rev3.
- Cable USB 2.0 A macho a B macho.
- Miniprotoboard
- LED y resistencia
- Sensor de temperatura y humedad DTH22.
- Sensor de ultrasonido HC-SR04.
- Módulo de Bluetooth HC-06.
- Puente en H L298N.
- Portapilas.
- Dos pilas de 3.7V IRN 18650 F1L.
- Pila recargable 9V Li-Ion marca Keenstone.
- Filamento PLA.
- Cables Dupont macho-hembra.
- Cables Dupont macho-macho.
- Conector clip para pila 9V a DC Jack.
- Rueda para motores.
- Rueda giratoria.
- Motores 3-12V.

Plataformas de desarrollo

- Arduino IDE.
- FreeCAD.
- Mit App Inventor 2.
- Fritzing.

13 PRESUPUESTO

Los costos de los materiales empleados a lo largo del proyecto aparecen adjuntos en la siguiente tabla. Hay que tener en cuenta que estos precios son orientativos puesto que depende del lugar donde se adquieran.

Materiales	Unidades	Precio €
Ordenador con sistema operativo Windows 10.	1	-
Tarjeta Arduino UNO rev3.	1	10
Cable USB 2.0 A macho a B macho.	1	-
Miniprotoboard	1	2
LED	1	-
Sensor de temperatura y humedad DTH22.	1	6
Sensor de ultrasonido HC-SR04.	1	4
Módulo de Bluetooth HC-06.	1	7
Puente en H L298N.	1	9
Portapilas.	2	1
pila de 3.7V IRN 18650 F1L.	2	2
Pila recargable 9V Li-Ion marca Keenstone.	1	3
Filamente PLA.	-	-
Cables Dupont macho-hembra.	-	-
Cables Dupont macho-macho.	-	-
Conector clip para pila 9V a DC Jack.	1	-
Rueda para motores.	2	2
Rueda giratoria.	1	1.5
Motores 3-12V.	2	2
	Total	54.5 €

Por otro lado se encuentran los costos no materiales, como puede ser las horas invertidas, las plataformas de desarrollo utilizadas, etc. En la siguiente tabla se ajusta el coste aproximado.

Recursos	**Horas**	**Precio/Hora**
Arduino IDE	-	-
FreeCAD	-	-
Fritzing	-	-
Mitt App Inventor 2	-	-
Diseño chasis	30	20 €
Desarrollo aplicaciones	30	20 €
Desarrollo código Arduino	40	20 €
Desarrollo temario a impartir	100	20 €
Total	200 €	4.000 €

14 BIBLIOGRAFÍA

[1] ^ "Open Source". *ieee.org.*
[2] ^ "Arduino UNO pinout diagram". *arduino.cc.*
[3] ^ "Sketch build process ". *arduino.github.io.*
[4] ^ "C Reference Manual - Data Types". *gnu.org.*
[5] ^ "C Reference Manual - Arrays". *gnu.org.*
[6] ^ "C Reference Manual – Arithmetic Operators". *gnu.org.*
[7] ^ "Control Structures". *cs.fsu.edu.*
[8] ^ "Control Structure - Switch". *arduino.cc.*
[9] ^ "Control Structure - For" *arduino.cc.*
[10] ^ "Control Structure - DoWhile" *arduino.cc.*
[11] ^ "Microcontrolador" *raing.es.*
[12] ^ "DHT22 datasheet" *adafruit.com.*
[13] ^ "Medir distancia con ultrasonidos" *luisllamas.es*
[14] ^ "Building an Arduino on a Breadboard" *arduino.cc*
[15] ^ "Operadores a nivel de bit" *oracle.com*

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá