Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías Industriales

Puesta en marcha y programación de un robot animatrónico.

Autor: Jaime Haldón Vallellano

Tutores: Ignacio Alvarado Aldea

Jose María Maestre Torreblanca

Dpto. de Ingeniería de sistemas y automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado
Grado en Ingeniería en Tecnologías Industriales

Puesta en marcha y programación de un robot animatrónico.

Autor:
Jaime Haldón Vallellano

Tutores:
Ignacio Alvarado Aldea
Profesor titular
Jose María Maestre Torreblanca
Catedrático de Universidad

Dpto. de Ingeniería de sistemas y automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo de Fin de Grado: Puesta en marcha y programación de un robot animatrónico.

Autor: Jaime Haldón Vallellano

Tutores: Ignacio Alvarado Aldea
Jose María Maestre Torreblanca

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis maestros

A mis amigos

A todo aquel que estuvo, está o estará presente en este camino…

Agradecimientos

En primer lugar, me gusataria agredecer a mi familia por apoyarme y estar siempre ahí respaldandome, sin lo cual todo esto nunca hubiera sido posible. A mis padres especialmente, porque sin el ánimo constante (y tan constante) de mi madre o el apoyo de mi padre, cuando todo se hacía algo cuesta arriba, esta etapa hubiese sido muy difícil. Sin olvidarme de los consejos de mi tío Pepe, o las felicitaciones y sonrisas de mis abuelos, hermanos o primos cuando este Currito iba dando sus primeros pasos, y en general a lo largo de todo el Grado. ¡Mil gracias!

A mis amigos, por ser un pilar fundamental durante todo este tiempo, y a todos aquellos compañeros de clase por echar una mano y hacer más fácil esta aventura. Gracias a mi grupo Bolas, por poner un sinfín de risas y momentos inolvidables durante todos estos años, por apoyamos mutuamente en cada día de biblioteca o pandemia, sin ello hubiese sido muy complicado llegar hasta aquí.

Gracias también a todos los profesores que de una u otra forma me han formado profesional y personalmente a lo largo de este camino, especialmente en los difíciles momentos de pandemia, con un esfuerzo extra.

Agradecer para terminar, a Ignacio Alvarado, a Pepe Maestre y al equipo de Robots Can Cry por hacer posible este genial proyecto, en el cual estuve motivado desde el primer día y me ha aportado mucho durante su desarrollo. No me imaginaba un mejor proyecto para cerrar esta etapa.

¡Gracias por todo a todos!

Jaime Haldón Vallellano

Sevilla, 2021

Resumen

En las últimas décadas, el mundo de la tecnología ha avanzado a pasos agigantados, mejorando y facilitando la vida de las personas. Cada día la barrera entre ciencia-ficcion y realidad va disminuyendo, y así lo demuestran campos como la Automática o la Robótica, haciendo que la interacción humana con la máquina sea cada vez más fácil y llamativa, al igual que sorprendente.

Este proyecto, así lo pondrá de manifiesto de la mano de Currito, un robot-animatrónico, basado en la mascota de la Expo-92. Para que eso sea posible, es necesario un amplio trabajo en ámbitos de electrónica y programación, mayoritariamente, los cuales serán desarrollados a lo largo de este trabajo.

Para ello, se trabajará con dos de las herramientas más de moda en los últimos años para creación y desarrollo de proyectos electrónicos, Arduino y Raspberry Pi, no solo de forma independiente sino conjunta, que harán posible la interacción de Currito con el entorno que le rodea, a través de los diversos sensores y actuadores que este contiene.

Abstract

In the last decades, the field of technology has advanced by leaps and bounds, improving and giving more facilities to people's lifes. Everyday, the barrirer between science-fiction and reality is becoming smaller, as fields like Automatic or Robotics often proves, making the interaction of humans with machine easily and startling, as well as surprising.

This project will show that, through the animatronic-robot Currito, based on the Expo-92's pet. In oder to making this possible, its necessary a huge work, mostly in terms of electronic and programming, which is going to be developed along this memory.

For this job, it will be used two of the most famous gadgets for the creation and development of electronic proyects, Arduino and Raspberry Pi, not only in an independent way one from the other, but also jointly, which will make possible the interaction of Currito with the environment that sorrounds him, through the numerous sensors and actuators he contains.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGOS

Notación

A	Amperios
ABS	Acrylonitrile Butadiene Styrene
AC	Alternating Current
cm	Centímetros
CSI	Camera Serial Interface
dB	Decibelios
DC	Direct Current
fps	frames por segundo
g	Gramos
GHz	gigahercios
GPIO	General Purpose Input Output
GND	Ground
HDMI	High-Definition Multimedia Interface
Hz	Hercios
IDE	Integrated Development Environment
Kb	Kilobytes
KHz	Kilohercios
LED	Light Emiting Diode
LPDDR	Low-Power Double Data Rate
m	metros
mA	Miliamperios
mm	milímetros
MP	Megapíxeles
PWM	Pulse Width Modulation
PWRIN	Power Input
RAM	Random Access Memory
SO	Sistema Operativo
SPST	Single Pole Single Throw
SSH	Secure Shell
USB	Universal Serial Bus
V	Voltajes
VCC	Voltage Common Collector
VNC	Virtual Network Computing
µs	microsegundos
º	Grados sexagesimales
:	Tal que
<	Menor o igual
>	Mayor o igual

1 INTRODUCCIÓN

"¿Heredarán los robots la Tierra? Sí, pero serán nuestros hijos"

- Marvin Minsky -

En este primer capítulo, se va a dar una visión general acerca del porque se ha decidido realizar el presente proyecto y cuáles son los objetivos de este, además de estructurar la presente memoria indicándose de que tratarán cada uno de los apartados de los que esta se compone.

Figura 1.1: Aspecto Currito.

1.1 Objetivo

Este proyecto nace de la mano de la empresa Robots Can Cry, fundada por Javier Coronilla, dedicada a la industria de la animatrónica. Esta, se centra en el desarrollo de proyectos del campo de la robótica y eléctronica que simulan la apariencia y comportamiento humano, de personajes reales o de ficción, mediante control remoto o programación. En este caso, el diseño elaborado, mostrado en la imgen superior, está basado en la mascota de la Expo-92 celebrada en Sevilla, Curro.

Por tanto, lo que se pretende es trabajar todo el proceso, partiendo del diseño, desde el montaje electrónico hasta la simulacion del comportamiento humano del animatrón con el entorno, pasando por la programación o proceso necesario para este fin, observándose en todo este camino las dificultades y limitaciones de este.

Para ello, se realizará un análisis, tanto de los componentes electrónicos necesarios que permitirán el comportamiento animado del robot, como del software y programación que harán posibles que dichos componentes trabajen conjuntamente, y de forma adecuada, para un correcto comportamiento del Currito en la vida real y el entorno. Además, cabe destacar la importancia de la buena coordinación de todos los sensores y actuadores implicados en este proyecto para que el comportamiento resultante sea lo mejor posible.

Concretamente, se pretende que Currito pueda interactuar con su entorno de manera que sea capaz de reaccionar ante estímulos externos, haciendo uso de los múltiples sensores y actuadores que incorpora. Entre otras acciones, se pretende que sea capaz de detectar obstáculos en su camino y esquivarlos adquiriendo una nueva dirección mediante el giro de sus 2 ruedas, haciendo uso para ello, del sensor ultrasónico que este incorpora. Además de eso, Currito será capaz de:

- Tomar fotos y videos en su recorrido por el mundo que le rodea.
- Mostrar sus emociones por medio de colores, a través del anillo LED que incorpora.
- Simular un corpotamiento humano mediante el movimiento de su cuerpo (cejas, cresta, cuello…)
- Responder ante sonidos externos, respondiendo a ellos.
- Emitir sonidos en funcion del estado en que se encuentre.

1.2 Estructura de la memoria

En esta memoria, se comenzará con una descripción física del Currito. Se mostrará tanto un esquema de su apariencia, indicándose cada uno de las partes diferenciales de este, como características físicas del mismo acerca de las dimensiones, peso o forma de este animatrón.

Tras ello, se analizarán cada uno de los componentes electrónicos de los que se compone el robot, desde los cerebros como pueden ser las placas Raspberry PI y Arduino, hasta el botón basculante que permite el encendido del mismo.

Después de dicha descripción del harware del sistema, se introduciran los diferentes entornos software utilizados para la programación de cada uno de los componentes descritos en el apartado anterior.

Pasaremos luego finalmente, a explicar como se programan las distintas secciones del robot en los programas citados en el apartado anterior. Se mostrarán las declaraciones y librerías necesarias para el manejo de estos, así como un ejemplo para cada uno de dichas secciones. Además, se indicará como realizar la comunicación de Arduino y Raspberry por puerto serie con su ejemplo correspondiente, herramientas con las que se programará, y finalmente se elaborará el programa completo “esquiva-obstáculos”.

Para acabar, se valorará y concluirá el proyecto, relatándose la experiencia durante el desarrollo de este, para dar poúltimo paso a la sección de anexos donde encontraremos información de interés como puede ser códigos completos de los programas elaborados, presupuestos o procedimientos secundarios necesarios para la realización de este proyecto.

2 APARIENCIA Y COMPONENTES DE CURRITO

"No juzgue nada por su aspecto, sino por la evidencia.
No hay mejor regla."

- Charles Dickens -

En relación con la apriencia física de Currito, se va a mostrar a continuación un esquema del mismo donde se diferencian y enumeran cada una de sus partes. Posteriormente se describirá el material del que se compone y como fue elaborado. Además, se van a describir las características técnicas de los componentes electrónicos que equipa el Currito en su totalidad, aspecto relevante a conocer a la hora de trabajar con estos para evitar un mal funcionamiento de cualquiera de ellos. Al final de este capítulo y, tras haber descrito cada uno de dichos componentes, pasaremos a indicar el conexionamiento de cada uno de ellos a las 2 placas que se encargarán del control de estos, Arduino Uno y Raspberry Pi.

2.1 Características físicas

Para comenzar, se va a mostrar en la siguiente tabla, el peso y las dimensiones del robot:

Alto	37.8 – 46.0 cm (según inclinación de la cresta)
Ancho	23.8 cm
Largo	24.3 cm
Peso	2.72 kg

Tabla 2.1: Dimensiones y peso de Currito

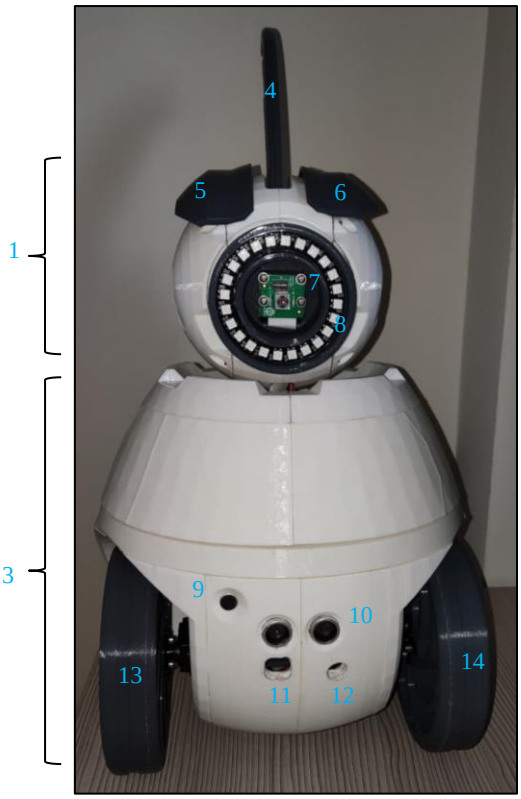

Figura 2.1: Imagen frontal de Currito

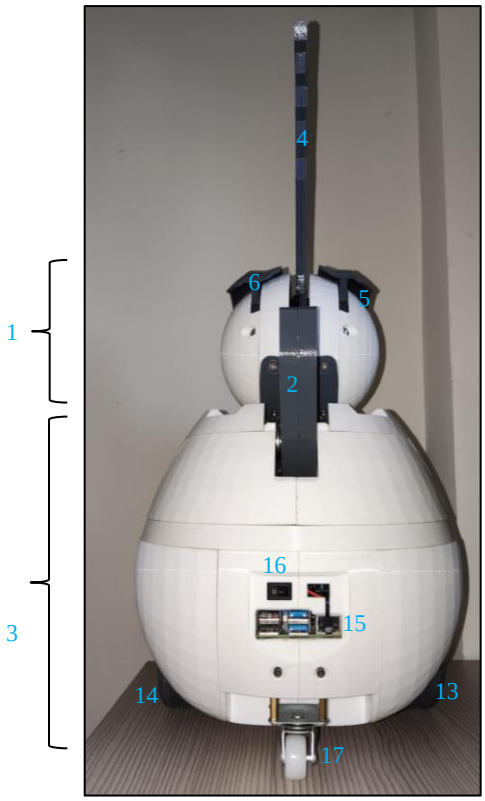

Figura 2.2: Imagen posterior de Currito

Figura 2.3: Imagen del perfil de Currito

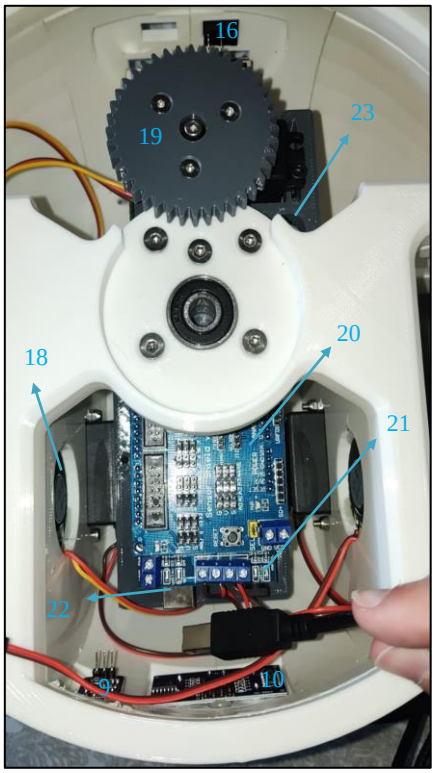

Figura 2.4: Imagen interior de Currito

Cada una de las partes enumeradas, se corresponden con los elementos que se listan a continuacion:

1.- Cabeza
2.- Cuello
3.- Cuerpo
4.- Cresta
5.- Ceja derecha
6.- Ceja izquierda
7.- Cámara Raspberry Pi
8.- Anillo LED
9.- Micrófono
10.- Sensor ultrasónico
11.- Abertura para conexión de USB a Arduino
12.- Abertrura para conexión de Jack a Arduino.
13.- Rueda izquierda
14.- Rueda derecha
15.- Puertos de conexión a Raspberry Pi
16.- Interruptor basculante
17.- Rueda loca
18.- Altavoz
19.- Engranaje para rotación del cuerpo
20.- Sensor Shield V5
21.- Motor Shield L298P
22.- Arduino Uno
23.- Raspberry Pi 4B

El material del que está construido el animatrón es ABS (Acrylonitrile Butadiene Styrene), uno de los materiales más usados y populares en la industria de la impresión en 3D, procedimiento por el que fueron creadas cada una de sus piezas. Buena rigidez, gran resistencia al impacto o ligereza, son algunas de sus características más destacables, y lo convierten en un gran candidato para este proyecto, a la vez que económico.

2.2 Elementos electrónicos

2.2.1 Arduino Uno

Figura 2.5: Placa Arduino Uno [1]

La placa Arduino Uno contituye el cerebro de nuestro animatrón, junto con la Raspberry Pi.

Se trata de una placa electrónica de Arduino, una compañía dedicada al desarrollo de hardware y software libre. Existen numerosos tipos de placas de Arduino. En este caso usaremos la placa Arduino Uno, una de las más usadas y que es compatible así con múltiples escudos, como usaremos en este proyecto, lo que sumado a sus especificaciones la hacen ideal para este, cuyas más importantes, se muestran en la siguiente tabla:

Micocontrolador	Atmel ATmega328 a 16 Mhz
Memoria SRAM	2 Kb
Memoria EEPROM	1 Kb
Memoria flash	32 Kb (0.5 dedicados al arranque)
Límites de voltaje de entrada	6-20V
Voltaje de entrada recomendado	7-12V
Intensidad de corriente para E/S	40mA

Tabla 2.2: Características Arduino Uno. [1]

En cuanto a los pines de E/S, esta placa contiene 14 pines digitales, de los cuáles 6 son PWM (3, 5, 6, 9, 10 y 11), y 6 analógicos. Además, contiene 2 pines de alimentación de salida de 5V y de 3.3V para alimentar los dispositivos que conectemos a la placa, además de 2 pines GND (masa).

En la parte izquierda de la placa encontramos los conectores de alimentación. Podemos hacer uso de uno u otro en funcion de la cantidad de dispositivos que conectemos a la placa, situándose en la parte inferior un Jack de alimentación externa que permite un voltaje de 6 a 12 V recomendados (7 a 12V como límites), en el caso de conectar muchos dispositivos, y en la parte inferior el conector USB Type-B, que se conectará al ordenador para cargarle a la placa el programa a ejecutar a la placa, y que permite una alimentación de solo 5V.

Tambien contiene un botón de reseteo de la placa, para reiniciar el programa cargado en la placa, además de varios LEDs, como uno que indica si la placa está encendida, situado en la parte superior derecha. Tiene una longitud de 6,9cm, un ancho de 5,3 cm y un peso aproximado de 25g.

En este proyecto no conectaremos directamente los sensores o actuadores directamente a esta placa, sino que utilizaremos 2 escudos o placas de distribución que serán descritas a continuación.

2.3 Módulo controlador de motores L298P

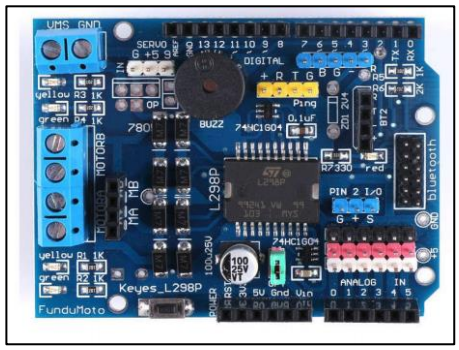

Figura 2.6: Placa controladora de motores L298P [2]

Esta placa de distribución, de dimensiones 6,8x5,8cm, nos permite usar los 14 pines de la placa Arduino Uno asignando un fin concreto para alguno de ellos (aunque pueden ser usados de manera tradicional como en la placa Arduino Uno):

- Los pines 10, 11, 12 y 13, son de salida y están dedicados al manejo de motores, cuyas cuatro conexiones se sitúan a la izquierda en los 4 modulos azules. Los pines 10 y 11 controlarian la velocidad de los motores, y los pines 12 y 13 la dirección de estos.
- El pin digital 9 se dedica a la conexión de un servomotor, cuyas 3 conexiones (tierra, alimentación y señal) se sitúa en la parte superior izquierda, de color blanco.
- El pin digital 4 se dedica al meno de un buzzer, situado en la parte central superior.
- Contiene en la parte derecha 2 módulos de conexión para bluetooth.
- Los 6 pines analógicos de la placa Arduino Uno ahora tiene cada uno una conexión a tierra y a alimentación. Se situan las conexiones en la parte inferior derecha (en negro tierra, en rojo alimentación y en blanco señal)
- Contiene una conexión PWRIN para alimentación externa en la esquina superior izquierda (VCC y GND) y un botón de reset en la parte inferior.

De esta placa utilizaremos concretamente la conexión a motores, para las 2 ruedas de Currito, que manejaremos con los pines 10, 11, 12 y 13 como se ha indicado, con control de velocidad mediante PWM (los pines 10 y 11 tienen esta caacterística).

Entre las especificaciones técnicas más importantes de este módulo tenemos las siguientes:

Chip driver controlador de motores	L298P
Voltaje de trabajo, alimentada a través del Arduino Uno	5V
Voltaje de trabajo con alimentación externa por VIN	6.5-12V
Voltaje de trabajo con alimentación externa por la conexión PWRIN	4.5-24V
Intensidad de corriente de trabajo de la placa	<36mA
Intensidad de corriente de salida por motores	<2A

Tabla 2.3: Características Motor Shield L298P [2]

2.4 Módulo Sensor Shield V5

Figura 2.7: Placa Sensor Shield V5 [3]

Esta placa, al igual que la anterior, es una placa de distribución de pines. En este caso los 14 pines digitales tienen una conexión a tierra y a alimentación independientes, al igual que los 6 pines análogicos. Conexiones de ambos bloques en la parte central de la placa. Esto hace posible la conexión de múltiples sensores y actuadores al Arduino, como es el caso, de ahí su elección en este proyecto. Tiene unas dimensiones de 5,8x5,8cm.

Además de eso, la placa contiene un botón reset en la parte central izquierda, una conexión para alimentación externa (VCC y GND) y otros módulos de conexión entre los que destacan los siguientes:

- Puerto para un módulo LCD paralelo o uno serial, ambos en la parte superior de la placa, a manejar con los pines digitales 2, 3, 4, 5 y 6, o 2, 3 y 4 respectivamente.
- Puerto para conexión de módulo bluetooth en la parte central derecha, a controlar con los pines digitales
- Puerto para conexión de tarjeta SD en la parte inferior izquierda, a manejar con los pines digitales 10, 11, 12 y 13.
- Puerto URF01+ para conexión de sensor ultrasónico en l aparte inferior derecha, a controlar mediante los pines analógicos A0 y A1.

2.5 Raspberry Pi modelo 4B

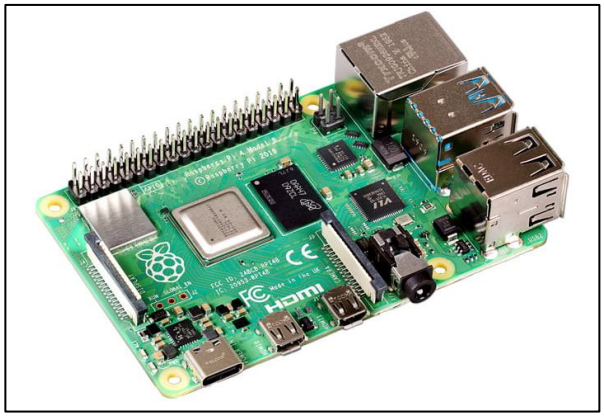

Figura 2.8: Raspberry Pi 4B [4]

La raspberry Pi 4B, junto con la placa Arduino Uno, constituye el cerebro de nuestro animatrón. Se trata de un mini ordenador, de bajo costo, pero de gran potencia, de dimensiones 8.6cm de largo y 5.7cm de ancho, con un peso alrededor de los 45g.

Existen varias versiones en funcion de la memoria RAM, siendo las características de la utilizada en nuestro caso la que se describe en la siguiente tabla:

CPU	Procesador Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC
GPU	VideoCore VI
Memoria RAM	2GB LPDDR4
Frecuencia de reloj	1.5 GHz
Alimentación	5V DC mediante puerto USB-C o por pines GPIO, con 3A de corriente recomendado.

Tabla 2.4: Características Raspberry PI 4B [4]

En cuanto a la conectividad, esta Raspbrry Pi 4B:

- Permite conexión WiFi Dual Band 2.4 GHz y 5.0 GHz
- Puede comunicarse vía Bluetooth 5.0
- Contiene un puerto Ethernet, 2 puertos tipo USB 2.0 y otros 2 USB tipo 3.0, y 40 puertos GPIO

En cuanto a las opciones multimedia, esta posee:

- 2 puertos micro-HDMI capaces de reproducir a 4K 60fps
- 1 puerto para conexión de cámara Raspberry PI
- 1 puerto para conexión de pantalla MIPI DSI
- 1 puerto 4-pole para audio y video

En el Anexo I de esta memoria se puede encontrar el procedimiento completo para la puesta en marcha de la Raspberry Pi, además de técnicas para su control remotamente.

2.6 Cámara Raspberry Pi V2 8MP

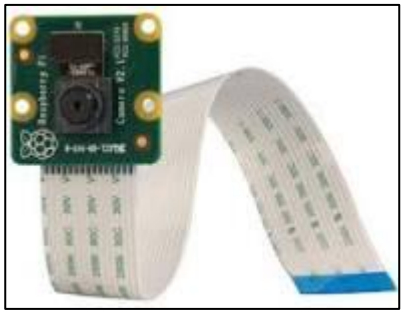

Figura 2.9: Cámara Raspberry Pi V2 8MP [5]

El primero de los dispositivos externos a conectar al Currito es la cámara Raspberry Pi. Esta cámara es compatible no sólo con la Raspberry Pi 4B que vamos a usar, sino con versiones anteriores de esta. Nos permitirá obtener fotos y videos, del entorno en el que Currito se desenvuelva, de gran calidad.

Algunas de las características más importantes de esta, son las siguientes:

Resolución	8 MP
Sensor	Sony IMX 219 PQ CMOS
Resolución máxima de imagen estática	3280 x 2464 píxeles
Velocidad máxima de cuadro en video	30fps a 1080p y 60fps a 720p
Tamaño sensor óptico	1/4”
Dimensiones	2.4 x 2.5 x 0.9cm
Peso	3g

Tabla 2.5: Características cámara Raspberry Pi V2 8MP [5]

Permite además la modificación de parámetros como la apertura, la resolucion, el brillo, la saturación o la simulacion de distintos efectos como modo noche, modo soleado o modo luces fluorescentes entre muchos otros. Es capaz también de calibrar automáticamente la exposición, el balance de blancos y negros, la detección de iluminación. Como vemos es una muy buena opción para que nos muestre la visión desde el Currito.

2.7 Anillo LED: Sparkfun neopixel 24x WS2812B 5050 RGB

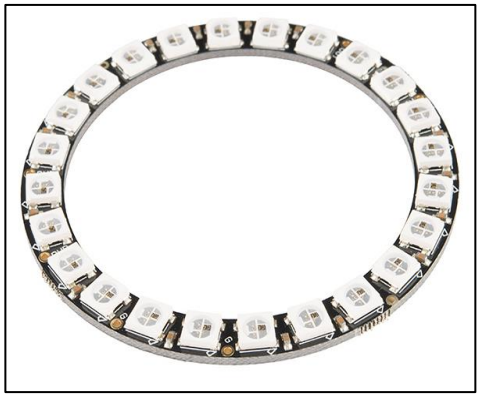

Figura 2.10: Anillo 24 LEDs RGB [6]

Mediante este anillo LED, se manifestarán las emociones del currito. Contiene 24 neopixeles de tipo WS2812B 5050 RGB. Este tipo de chip, contiene 4 pines: VDD, VSS, DIN y DOUT, según se muestra en la siguiente figura:

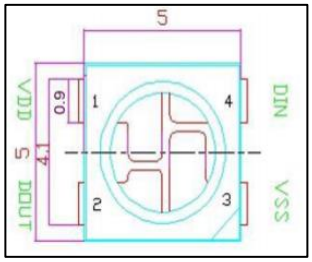

Figura 2.11: Esquema del chip LED WS2812B [7]

El pin VSS se corresponde con tierra y mediante el pin VDD se suministra la alimentación para cad uno de los LEDS. El pin DOUT de cada uno de los LEDs está conectado al DIN del siguiente a lo largo del anillo. A través de estos se transmite la información de un LED a otro, así, comunicándonos con el primero de ellos, cada LED actuará de transmisor de la secuencia para los LED posteriores. Cada uno de estos, en este tipo de chip, puede almacenar 24bits (8 por cada pixel de color R, G o B, 256 tonos cada uno), pudiendose representar más 16 millones de colores. [7]

Algunas de las características técnicas de este anillo de 24 neopixel, son las siguientes:

Voltaje de alimentación requerido	5V DC
Intensidad de corriente de trabajo de cada neopixel	18mA constantes, lo que permite un tono color muy constante aun con variaciones de voltaje a la entrada. 60mA de consumo máximo (color blanco a brillo máximo)
Dimensiones de los chips LED RGB	Modelo 5050: 0.5 x 0.5cm
Dimensiones del anillo	6.6cm de diámetro

Tabla 2.6: Características anillo LED WS2812B 5050 [7], [6]

2.8 Sensor ultrasónico HC-SR04

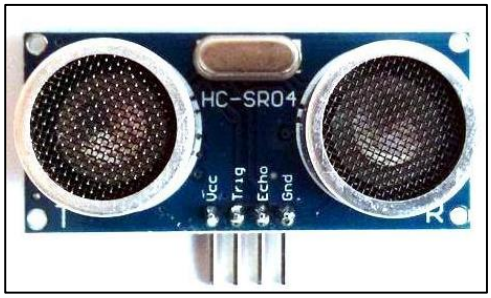

Figura 2.12: Sensor ultrasónico HC-SR04 [8]

Mediante este sensor ultrasónico HC-SR04, el animatrón será capaz de calcular la distancia a la que se encuentra el objeto más cercano en dirección longitudinal al sensor.

Esto lo calcula a partir del transimsor, situado a la izquierda del módulo en la imagen (T), y el receptor, a la derecha (R), enviando por el transmisor una onda de sonido y calculando el tiempo que tarda en llegar de vuelta al receptor. Conociendo la velocidad a la que se propaga dicha onda en el aire (340m/s) sabremos, a partir del tiempo que haya tardado en llegar de vuelta dicha onda, la distancia a la que se encuentra el objeto. Estas ondas de sonido se emiten a 40KHz, por lo que el oído humano no las capta.

Contiene 4 pines:

- Pin VCC: alimentación del sensor.
- Pin TRIG: por el cual se indica el envio de una señal ultrasónica por el transmisor.
- Pin ECHO: por el cual se recibe la señal, a través del receptor.
- Pin GND: conexión a tierra.

Entre las características de este sensor tenemos las siguientes:

Tensión de alimentación	5V DC
Consumo de intensidad de corriente	15mA
Distancia de detección	Desde 2cm a 4m
Precisión	3cm
Ángulo de medición máximo	15º
Dimensiones	45 x 20 x 15 mm

Tabla 2.7: Características sensor ultrasónico HC-SR04 [8]

2.9 Micrófono KY-038

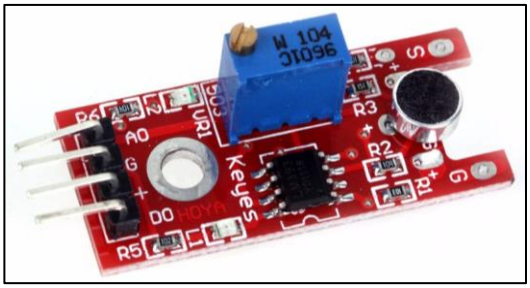

Figura 2.13: Micrófono KY-038 [9]

Este microfono de condensador, permitirá a Currito detectar cualquier tipo de sonido del entomo, pudiéndose ajustar la sensibilidad de este a través de la rotación de la placa dorada del potenciómetro azul del microfono. Hacia la dercha aumentaríamos la sensibilidad y hacia la izquierda la reduciríamos.

Este módulo contiene 4 pines de conexión para su manejo:

- Pines A0 y D0: indicarán un valor HIGH, si el sonido recibido es superior al umbrl indicado en el potenciómetro, de forma analógica o digital respectivamente.
- Pin G: correspondiente a conexión atierra.
- Pin +: correspondiente a la alimentación.

El LED situado en la parte superior (L2), indica la recepción de un sonido (más iluminado a más fuerte el sonido), y el LED situado en la parte inferior (L1), nos indica la alimentación del micrófono (encendido si así lo está el micrófono).

Las características técnicas de este micrófono KY-038 son las siguientes:

Tensión de alimentación	5V DC
Chip principal y micrófono	LM393 y electret
Gama de frecuencias que reconoce	10 a 10000 Hz
Sensibilidad mínima al ruido	58 dB
Dimensiones	36 x 15 x 15 mm
Peso	4g

Tabla 2.8: Características micrófono KY-038 [9]

2.10 Mini altavoz mp3

Figura 2.14: Mini altavoz mp3 [10]

Para la reproducción de sonidos y comunicación de Currito con el exterior, se usará este mini altavoz redondo Mp3, especialmente recomendado para proyectos con Arduino por su fácil manejo, y cuyas características, las cuales se muestran a continuacion, lo hacen ideal para su función en el animatrón:

Potencia	2W
Impedancia	8 ohmios
Diámetro del speaker	40 mm
Grosor	5.3 mm

Tabla 2.9: Características mini altavoz mp3 [10]

Este tiene 2 pines, uno "+", y otro "-", positivo y tierra respectivamente, para la alimentación del mismo, que será de 5V.

2.11 Servomotor Turnigy TGY50090

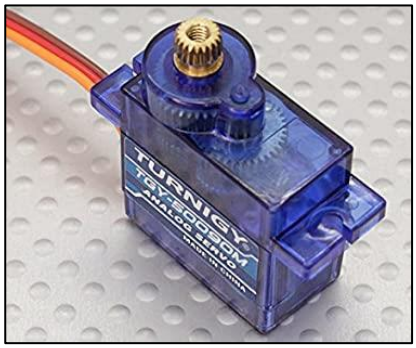

Figura 2.15: Servomotor Turnigy TGY50090 [11]

2 unidades de este servo de la marca Turnigy, serán utilizados para de darle movimiento a las 2 cejas del animatrón. Se trata de un micro-servo analógico capaz de rotar 180º, suficientes para un correcto movimiento de dichas cejas.

Algunas de sus características más relevante se muestran a continuación:

Tension de trabajo	4.8V – 6V DC
Torque según voltaje	De 1,6 kg/cm para 4.8 V a 2 kg/cm para 6.0 V
Velocidad según voltaje	De 0.08 sec/60º para 4.8V a 0.07 sec/60º para 6.0 V
Dimensiones	23.1 x 12.0 x 25.9 mm
Peso	9g

Tabla 2.10: Características servomotor Turnigy TGY50090 [11]

2.12 Servomotor Futaba 3003

Figura 2.16: Servomotor Futaba 3003 [12]

Un servo como este de la marca Futaba, será el encargado de darle movimiento a la cresta y otro, a la boca del animatrón. Se trata de un servo analógico, algo más grande que el anterior, capaz de rotar 180º, suficientes para un correcto movimiento de dicha cresta y boca.

Se muestran en la siguiente tabla las especificaciones más relevantes de este:

Tension de trabajo	4.8V – 6V DC
Torque según voltaje	De 3.17 kg/cm para 4.8 V a 4.10 kg/cm para 6.0 V
Velocidad según voltaje	De 0.23 sec/60º para 4.8V a 0.19 sec/60º para 6.0 V
Dimensiones	39.9 x 20.1 x 36.1 mm
Peso	37g

Tabla 2.11: Características servomotor Futaba 3003 [12]

2.13 Servomotor HXT12K M 11Kg

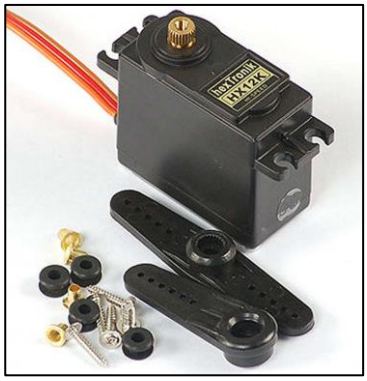

Figura 2.17: Servomotor HXT12K M 11Kg [13]

Este servo de la marca HexTronik, se trata de un servo analógico, algo más grande y potente que los 2 descritos anteriormente. Se utilizarán 4 servos como este. Uno irá dirigido al movimiento del cuello y otro a la rotación del cuerpo del Currito. Los otros 2 restantes, serán los encargados de mover las 2 ruedas del robot, pero en este caso estos serán capaces de girar completamente (360º), a diferencia de los anteriores que solo girarán 180º. acciones que requieren un tipo de servo más potente como este. Estas, son acciones que, como vemos, requieren un tipo de servo más potente como este.

Algunas de las características más relevantes de este servomotor, son las siguientes:

Tension de trabajo	4.8V – 6V DC
Torque según voltaje	De 9.4 kg/cm para 4.8 V a 11 kg/cm para 6.0 V
Velocidad según voltaje	De 0.20 sec/60º para 4.8V a 0.16 sec/60º para 6.0 V
Dimensiones	40.7 x 19.7 x 42.9 mm
Peso	55g

Tabla 2.12: Características servomotor HXT12K M 11Kg [13]

Este servomotor, al igual que los 2 anteriores, contiene 3 pines, uno positivo, otro a tierra, y otro para señal, que será el encargado de mover el servo a la posición deseada.

2.14 Batería LiPo

Figura 2.18: Batería LiPo [14]

Para alimentar el Arduino Uno, así como la placa Motor Shield L298P o el Sensor Shield V5, además de todos los sensores y actuadores conectados a estas placas, se utilizará esta batería de tip LiPo, que cubrirá las necesidades de voltaje y amperaje requerida por estos.

Cabe destacar la importancia de la elección de un correcto amperaje, ya que este varíara durante eñ funcionamiento del robot, en función de los distintos dispositivos que esten trabajando al mismo tiempo, conectados a esta batería. El amperaje que esta nos proporciona es correcto para cubrir dichas necesidades, además de tener una buena autonomía para la actividad a desarrollar por Currito.

Se muestran en la siguiente tabla, las características destacadas acerca de esta batería:

Capacidad	1500mAh
Voltaje	7.4V
Amperaje máximo	1.5A
Tiempo de carga	Alrededor de 2 horas a 1.5A
Dimensiones	73 x 33,5 x 15 mm
Peso	68g

Tabla 2.13: Características batería LiPo a utilizar [14]

Los tipos de conectores de esta batería son los que se muestran en la siguiente imagen:

Figura 2.19: Tipos de conectores de la batería LiPo a utilizar [14]

En nuestro caso, vamos a utilizar el segundo mostrado en la imagen, ya que poseo un empalme al interruptor de ese tipo.

2.15 Interruptor Basculante

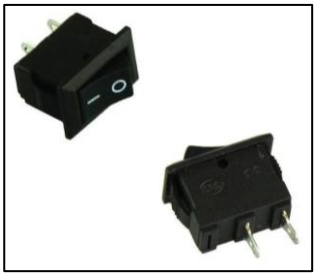

Figura 2.20: Interruptor basculante [15]

El último de los elementos electrónicos que vamos a utilizar, es este interruptor basculante, para el cómodo encendido del Currito, mediante los 2 pines SPST de encendido y apagado que incorpora.

Algunas de las características a tener en cuenta de este interruptor son las siguientes:

Voltaje de operación máximo	250V AC
Intensidad de corriente máxima	3A
Dimensiones	15 x 10 x 17 mm
Peso	9g

Tabla 2.14: Características interruptor basculante [15]

2.16 Conexiones

A continuación, se van a describir todas las conexiones que hay que realizar de los distintos dispositivos electrónicos descritos en este capítulo. Dividiremos estas conexiones según vayan al Arduino Uno o a la Raspberry Pi, indicándose en ambos subapartados como van a ser alimentadas cada una de estas placas.

2.16.1 Placa Raspberry Pi

La alimentación de este mini ordenador se realizará a través del puerto USB-tipo C situado a la izquierda de los puertos micro-HDMI.

A la Raspberry Pi 4B, de los elementos electrónicos descrios más arriba, conectaremos únicamente la cámara Raspberry. Esto se hará a través del conector CSI que esta cámara trae, al puerto CSI de la placa correspondiente, situado entre los puertos micro-HDMI y el conector Jack para audio y video, según se muestra en la imagen:

Figura 2.21: Conexión de cámara a Raspberry Pi [16]

Es importante diferenciar en el conector de la cámara una pestaña azul por una de las 2 caras las cuales deberá quedar mirando hacia la parte de la placa donde se encuentran los puertos USB, tal y como se puede ver en dicha imagen.

En cuanto a otras posibles conexiones que se puedan hacer a esta placa para el desarrollo de este proyecto, caben destacar el uso de un cable micro-HDMI al puerto correspondiente de esta para visualizar en pantalla la interfaz de esta o la conexión de ratón o teclado a través de los puertos USB 2.0 o USB 3.0 para la configuración o programación en esta. También se podría controlar la Raspberry Pi remotamente desde otro ordenador mediante SSH o VNC, como he utilizado en mi caso, algo que será descrito más detalladamente en el anexo I de esta memoria junto con la puesta en marcha la misma.

2.16.2 Placa Arduino Uno

Con la placa Arduino Uno se manejarán el resto de dispositivos electrónicos descritos en este capítulo, a parte de la cámara Raspberry Pi. Todos estos se conectarán de manera indirecta al Arduino Uno, haciendose uso de los 2 escudos mostrados anteriormente: el Motor Shield L298P y el Sensor Shield V5.

Estos 2 módulos se conectarán encima del Arduino Uno. Primero, se conectará el motor shield y encima de este el sensor shield, de manera que las 3 placas compartirán los pines, por lo que habrá que tener especial cuidado de no dar un uso en distintas placas a los mismos pines. Por ejemplo, para el control de motores en el motor shield se utilizan los pines 10, 11, 12 y 13, por lo que no podrán ser usados al mismo tiempo en el sensor shield situado justo arriba.

Hay que tener especial cuidado a la hora de conectar estas 3 placas. Deben quedar alineados los agujeros para el atornillado de estas 3 placas, conectándose los pines de las sucesivas placas a la derecha de la de abajo, estando a la derecha el pin digital 0 y el analógico A0 en ambos laterales de la placa inferior. Puede entenderse esto algo mejor a través de la siguiente imagen donde quedan las 3 placas conectadas en el interior del Currito:

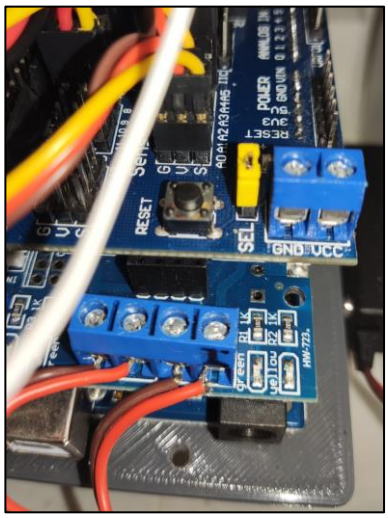

Figura 2.22: Conexión de las placas Arduino Uno, Motor Shield L298P y Sensor Shield V5

Al motor shield únicamente se conectarán los 2 servomotores en cargados del manejo de las 2 ruedas del animatrón, el resto de sensores y actuadores se conectarán al sensor shield de la siguiente forma cada uno de ellos:

- Los 3 pines (tierra, positivo y señal) del **anillo de 24 LEDs RGB** se conectarán a GND, VCC y señal del **pin digital 9** del sensor shield.
- Los 4 pines del **sensor ultrasónico HC-SR04** se conectarán de la siguiente forma: la tierra, VCC y **trigger** del sensor a GND, VCC y señal del **pin digital 2** respectivamente, y el pin **echo** del sensor a la señal del **pin digital 3**. De manera que los pines 2 y 3 se encargarán de controlar este sensor
- El **micrófono KY-038** contiene 4 pines, pero solo necesitamos 3: GND, "+" y, o bien A0 o bien D0, en función de si queremos obtener el valor de sonido captado en forma digital o analógica. En mi caso, me decido por D0, de manera que estos 3 pines del micrófono serán conectados a tierra, VCC y señal del **pin digital 8** del sensor shield.
- El **altavoz** contine 2 pines: negativo y positivo. Estos los conectamos respectivamente a GND y VCC del **pin digital 7** del sensor shield.

- Los 2 servomotores de las ruedas se conectarán a las 4 conexiones para motores que se ven en la imagen superior en el motor shield, de manera que el servomotor de la rueda izquierda (mirando de frente al Currito) se conectará a los 2 puertos de la izquierda y el servomotor de la rueda derecha a los 2 puertos de la derecha. Estos 4 puertos se manejarán con los pines digitales 10, 11, 12 y 13.
- Los 3 pines (tierra, positivo y señal) del servomotor HXT12K responsable de la **rotación del cuerpo**, se conectará a los 3 correspondientes del **pin analógico A0** del sensor shield.
- El servomotor HXT12K encargado del movimiento del **cuello**, será conectado de igual forma al **pin analógico A5** del sensor shield.
- El servomotor Turnigy TGY50090 encargado del movimiento de la **ceja derecha** se conectará al **pin analógico A0** del sensor shield.
- El servomotor Turnigy TGY50090 encargado del movimiento de la **ceja izquierda** se conectará al **pin digital 5** del sensor shield.
- El servomotor Futaba 3003 encargado del movimiento de la **cresta** se conectará al **pin digital 6** del sensor shield.

Estas 3 placas (Arduino y escudos), se alimentarán mediante la batería LiPo descrita anteriormente. Esta se conectará a los puertos para alimentación externa correspondientes (VCC y GND) del motor shield. A través de esta, estarán alimentados tanto el Arduino como el sensor shield, ambos a 5V pues es lo que este motor shield es capaz de transmitir a las demás, suficiente para el correcto funcioinamiento tanto del Arduino uno como de los componentes conectados al sensor shield.

Los 7.4V de esta batería decidimos conectarlos a ese motor shield y no a cualquiera de las otras placas, porque el movimiento de las ruedas por parte de los servomotores que mueven a estas, necesita algo más de 5V para mover al animatrón. Asimismo, nos conviene tener en el sensor shield 5V, que es la tensión a la que trabajan los dipositivos conectados a esta, y no directamente los 7.4V de la batería, que en este escudo podría provocar fallos en los componentes. Con esta configuración todo quedaría alimentado correctamente.

A su vez, esta batería se empalma al interruptur baasculante, de manera que para encender o apagar el Currito no sea necesario conectary desconectar la batería de la placa, sino simplemente pulsar el botón del interruptor correspondiente.

3 SOFTWARE EMPLEADO

Una máquina puede hacer el trabajo de cincuenta hombres ordinarios. Ninguna máquina puede hacer el trabajo de un hombre extraordinario.

- Elbert Hubbard

En este capítulo vamos a presentar y detallar los 2 entornos software que vamos a utilizar para la programación de los distintos componentes del Currito y con ello a este. El primero de ellos, Arduino Uno es el que utilizaremos para programar en el Arduino Uno, y el segundo de ellos, Thonny Python lo usaremos en la Raspberry Pi.

3.1 Arduino IDE

Figura 3.1: Logo Arduino IDE [17]

Arduino IDE es el software de programacaion libre en Arduino. Está escrita en lenguaje Java y es compatible con Windows, Linux y Mac OS y cuyo lenguaje de programación es similar a C++. Su primera versión fue lanzada en 2005 [18]. Se puede descarga de forma gratuita en la página oficial de Arduino: arduino.cc [17].

Esta aplicación permite compilar o verificar el código elaborado y cargarlo en cualquiera de las placas de Arduino. Estos programas qo sketchs que elaboremos en Arduino, cuya extensión es ". ino", tienen una fácil estructura formada por 2 partes o funciones obligatorias: el bloque "void setup", dedicado a la inicialización de variables y asignación de pines y que sólo se ejecuta una vez, y el bloque "void loop", que es un bucle que se ejecuta continuamente una vez cargado en la placa.

A continuacion se muestra un ejemplo del interfaz de un programa elaborado en Arduino IDE:

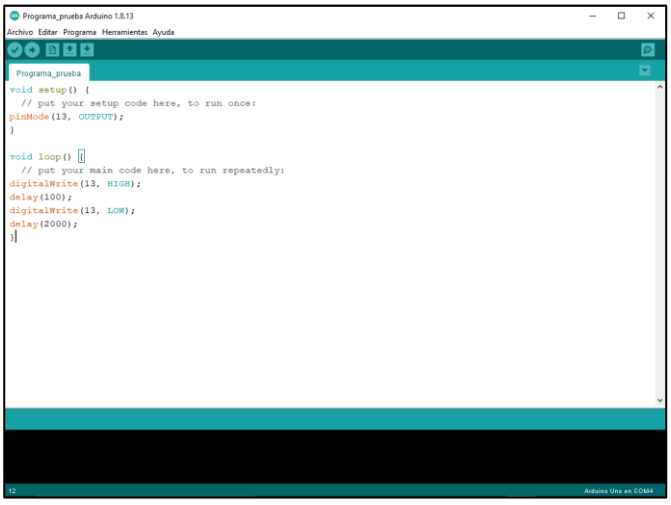

Figura 3.2: Ejemplo interfaz Arduino IDE

Cabe señalar las siguientes secciones de la interfaz de Arduino IDE, a la hora de trabajar con este programa:

- Archivo: lo usaremos para abrir y guardar archivos o ejemplos, entre otras opciones.
- Editar: permite modificar el tamaño de fuente, comentar, descomentar, copiar o cortar código...
- Programa: en esta sección podremos compilar el código, cargarlo a la placa o incluir librerias…
- Herramientas: nos permite, entre otras opciones, configurar que tipo de placa vamos a usar y en que puerto del pc, algo necesario para cargar el código a nuestra placa.
- Ayuda: sección que nos proporciona información adicional sobre el entorno Arduino.
- Enla parte superior izquierda, junto a estas secciones encontramos atajos para compilar, guardar, subir, cargar un programa o abrir uno nuevo.
- En la parte inferior de la interfaz, encontramos la barra de mensajes que nos proporcionara información del estado de nuestro programa al ejecutarlo o subirlo a la placa, entre otras cuestiones.

Contiene además múltiples librerías, desarrolladas por Arduino o por terceros, que podemos utilizar en nuestros proyectos y que facilitan en gran medida la programación, entre las que cabe destacar "Servo.h" o "Adafruit_NeoPixel.h", que utilizaremos en este proyecto. Esta última, se debe descargar ya que no viene en el paquete de librerías con la instalación del sofware, al igual que las de terceros.

Por último, conviene destacar la presencia del monitor serie, una herramienta que usaremos en este proyecto, que podemos abrir con el atajo en la parte superior derecha o mediante el menú Herramientas, y que nos permite mostrar los datos enviados o recibidos por el puerto serie de Arduino, lo que nos permitirá conectamos con la Raspberry Pi, algo que se desarrollará en el siguiente capítulo.

Este software lo usaremos en este proyecto para programar todos los dispositivos conectados a Arduino Uno o a cualquiera de sus 2 escudos (el sensor y el motor shield) y para comunicar Arduino y Raspberry PI por puerto serie.

3.2 Thonny Python IDE

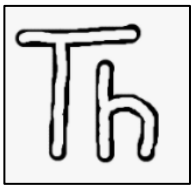

Figura 3.3: Logo Thonny Python IDE [19]

Thonny Python IDE es un entorno de desarrollo de programación en lenguaje Python. Es compatible con Windows, Linux y Mac OS, además de poderse instalar en Raspbian o Ubuntu, mediante el instalador de paquetes. En nuestro caso viene instalada por defecto en el sistema operativo Raspbian, algo que se viene haciendo en este sistema operativo desde 2017 por la gran recepción de esta por parte de la comunidad educativa [19]. Este puede ser descargado de forma gratuita desde la página oficial: Thonny.org [20].

En cuanto al lenguaje a utilizar, Python, se trata de un lenguaje de código abierto, de alto-nivel, interpretado, no compilado, lo que se traduce en más lentitud a la hora de correr el programa, pero permite ser utilizado en otras plataformas (lenguaje multiplataforma). Se trata además de un lenguaje de tipo dinámico, por lo que no será necesario especificar el tipo de dato asociado a una variable, esta variable se adapta a lo que hacemos con ella al escribir el programa.

Este IDE del que vamos a hacer uso, se caracteriza por su sencillez y fácil manejo, con una interfaz simple pero dotada de multitud de funciones para el usuario, la cual se muestra en la siguiente imagen:

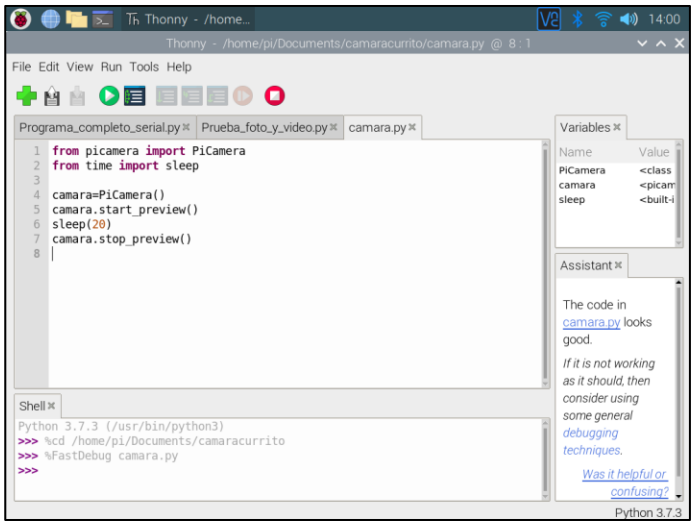

Figura 3.4: Ejemplo Interfaz Thonny Python IDE

Entre las secciones de dicha interfaz a destacar, destacan los siguientes:

- File: permite abrir, guardar archivos o cerrar el programa entre otras opciones.
- Edit: aquí podemos copiar, cortar o pegar código, comentar o descomentar parte del código escrito, o desahcer o rehacer acciones en el editor, entre otras opciones.
- View: en esta sección podemos aumentar o disminuir el tamaño de la fuente o modificar partes de la interfaz como el menú asistente o el menú Shell.
- Run: aquí podemos correr el programa de forma completa, abortar la ejecución de este, o compilarlo de forma completa o paso a paso.
- Tools: donde podemos adminitrar las librerías o cambiar opciones más avanzadas sobre el desmpeño del programa y sus ditintas secciones.
- Help: un menú de ayuda al usuario sobre como utilizar el software o sobre la versión de este.
- Menú Shell: aquí se mostrará información sobre el la compilación o estado del programa en ejecución, además de información sobre errores que impidan eso.
- Menú Assistant: un menú que proporciona más información sobre el estado de nuestro código que el menú Shell, con avisos de posibles mejoras en nuestro código que podrían evitar errores, si las hubiera.
- Menú Variables: esta sección nos proporciona información en tiempo real sobre el valor de las diferentes variables usadas en nuestro programa.
- En la esquina inferior derecha podemos observar la versión actual de la aplicación.

En la zona inferior de las 6 primeras secciones descritas, encontramos atajos para realizar acciones como ejecutar el programa o compilarlo línea a línea, guardar o cargar un programa, o abrir uno nuevo. Podemos tener abierto más de uno en la interfaz y viajar de uno a otro con facilidad.

Se pueden utilizar multitud de librerías, que deben primero ser instaladas en nuestro sistema operativo, las cuales deberemos importar al inicio del sketch.

Este software lo utilizaremos principalmente para programar la cámara Raspberry, además de para comunicar la Raspberry Pi con Arduino por puerto serie.

4 PROGRAMACIÓN DE CURRITO

"Programa siempre tu código como si el tipo que va a tener que mantenerlo en el futuro fuera un violento psicópata que sabe dónde vives"

- Martin Goldin -

En este capítulo vamos a tratar la programación de cada uno de los componentes electrónicos que contiene el animatrón, descritos anteriormente, usando para ello los 2 softwares presentados en el capítulo anterior.

Se comenzará tratando cada uno de estos de forma independiente, para finalmente elaborar un programa completo en el que se trabaje con todos ellos, pasando por la comunicación de Arduino y Raspberry Pi por puerto serie.

Se explicará en cada uno de los programas elaborados cada una de las declaraciones necesarias para el comportamiento de los dispositivos, así como las librerías utilizadas o las funciones adicionales elaboradas.

4.1 Programación de secciones por separado

Para comenzar, pasaremos a programar cada una de las partes del Currito de forma independiente para conocer el funcionamiento de cada una de estas. Comenzaremos por el manejo de los servomotres, asociados a las cejas, crsta, cuello y aquel que permite la rotación del cuerpo. Posteriormente expicaremos como manejar componentes electrónicos como el anillo LED, el sensor ultrasónico, el microfono y el altavoz. Finalmente se mostrará como programar los servomotores de las ruedas con el motor shield, y la cámara de la Raspberry Pi en Python.

4.1.1 Cresta

En primer lugar, vamos a trabajar con la cresta del animatrón, cuyo movimiento era posible gracias al servomotor Futaba 3003 que, como se indicó, está conectado al pin digital 6.

A la hora de trabajar con servos, conviene aclarar como funcionan estos. Por los pines GND y VCC serán alimentados mientras que por el pin Señal, se enviarán pulsos en funcion de cuantos grados queramos que gire nuestro servo. Si estos pulsos son de 1ms en un periodo de 20ms, que es el periodo que el servo espera la racepcion de pulsos, este grara a su mínimo valor (0º), mientras que si el pulso es de 2ms girará lo máximo que pueda (180º). Por lo que enviando señales de entre 1 y 2ms manejaremos los grados que este gire, aunque esto es un valor teórico y puede variar de un servo a otro como veeremos a continuación.

Para trabajar con servos en Arduino, vamos a utilizar la librería "Servo.h" que facilitará el trabajo con estos, por lo que escribimos lo siguiente para incluir esta librería, al principio del programa:

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

Tras esto, ya tenemos incluida la librería. Ahora debemos declarar una variable de tipo "Servo", por cada uno de los servos que vayamos a usar. Así, tendríamos que escribir lo siguiente:

```
Servo servo1;
```

Luego, debemos asociar la variable de nuestro servo al pin correspondiente para poder manejarlo. Para ellos utilizamos la siguiente declaración, en el bloque "void setup":

```
servo1.attach(6,PULSOMIN,PULSOMAX);
```

donde el primer argumento indica en que pin del Arduino lo vamos a conectar (pin digital 6), y el segundo y el tercero, los microsegundos que definimos como pulsos minimos y máximos del servo (típicamente 1000 y 2000 como se dijo anteriormente). En este caso esos microsegundos para que el servo pueda llegar alos 0º y los 180º coinciden con los teóricos: 1000 y 2000 microsegundos respectivamente.

La última declaración importante que debemos conocer es:

```
servo1.write();
```

que nos permite mover el servo a los grados deseados, el cual será su argumento, en grados sexagesimales.

Con todo esto, el programa de ejemplo que creamos es el siguiente, en el que el servo se moverá primero a los 0º, luego a los 90º y tras ellos a 180º cada acción 2 segundos después de la anterior, de forma contnua (en el bloque loop):

```
#include <Servo.h>

 Servo servo1;

int PULSOMAX=2500;
int PULSOMIN=500;

void setup() {
  servo1.attach(6,PULSOMIN,PULSOMAX);
}

void loop() {
  servo1.write(0);
  delay(2000);
  servo1.write(90);
  delay(2000);
  servo1.write(180);
  delay(2000);

}
```

Código 4.1: Ejemplo completo de programación de la cresta

Compilándolo y cargándolo a la placa observamos un correcto funcionamiento del servomotor.

4.1.2 Cejas

Para el caso de las cejas, se procede de forma análoga a la cresta. Ambas cejas van a ser controladas por los servos Turnigy TGY 50090 como se comentó. La ceja derecha se conectará al pin analógico A1 y la ceja izquierda al pin digital 5 como se indicó.

Con respecto al programa elaborado en el subapartado anterior, lo único que debemos modificar a continuación es el pin al que estarán conectados los servos en la declaración "attach", y el valor de los pulsos mínimo y máximo para un correcto recorrido de estas cejas. En este caso, tenemos para la ceja derecha unos pulsos mínimos y máximos respectivamente de 1000 y 1800 microsegundos, mientras que en la ceja izquierda estos serán de 1000 y 2000 microsegundos respectivamente.

Así el programa completo para que la ceja derecha se mueva a los 0º, luego a los 90º ytras ello a los 180º, 2 segundos de diferencia entre acciones, queda como sigue:

```
#include <Servo.h>
 Servo servo1;
int PULSOMAX=1800
int PULSOMIN=1000;

void setup() {
  servo1.attach(A1,PULSOMIN,PULSOMAX);
}

void loop() {
  servo1.write(0);
  delay(2000);
  servo1.write(90);
  delay(2000);
  servo1.write(180);
  delay(2000);
}
```

Código 4.2: Ejemplo completo de programación de la ceja derecha

Mientras que para la ceja izquierda tendríamos:

```
#include <Servo.h>
 Servo servo1;
int PULSOMAX=2000
int PULSOMIN=1000;

void setup() {
  servo1.attach(5,PULSOMIN,PULSOMAX);
}

void loop() {
  servo1.write(0);
  delay(2000);
  servo1.write(90);
  delay(2000);
  servo1.write(180);
  delay(2000);

}
```

Código 4.3: Ejemplo completo de programación de la ceja izquierda

4.1.3 Cuello

Para el movimiento del cuello, será un servomotor HXT12K M 11kg el que se encargue, el cual estará conectado al pin analógico A5 del Arduino. En este caso, los valores de pulsos mínimo y máximo distan algo más del valor teórico, siendo estos respectivamente: 500 y 2500 microsegundos. Quedará entonces el programa completo para el movimiento del cuello como sigue:

```
#include <Servo.h>
 Servo servo1;
int PULSOMAX=2500
int PULSOMIN=500;

void setup() {
  servo1.attach(A5,PULSOMIN,PULSOMAX);
}

void loop() {
  servo1.write(0);
  delay(2000);
  servo1.write(90);
  delay(2000);
  servo1.write(180);
  delay(2000);

}
```

Código 4.4: Ejemplo completo de programación del cuello

4.1.4 Rotación del cuerpo

El siguiente de los servos a configurar, será el encargado de la rotación del cuerpo del animatrón, de modelo HXT12K M 11kg, el cual será conectado al pin analógico A0 en este caso. Los valores de pulso mínimo y máximo a configurar ahora, son respectivamente: 600 y 2400 microsegundos. Quedará entonces dicho ejemplo en este caso como se muestra a continuación:

```
#include <Servo.h>
 Servo servo1;
int PULSOMAX=2500
int PULSOMIN=500;

void setup() {
  servo1.attach(A5,PULSOMIN,PULSOMAX);
}

void loop() {
  servo1.write(0);
  delay(2000);
  servo1.write(90);
  delay(2000);
  servo1.write(180);
  delay(2000);

}
```

Código 4.5: Ejemplo completo de programación de la rotación del cuerpo

4.1.5 Movimiento de todos los servomotores

Una vez trabajado individualmente con cada uno de los servomotores, excluyendo los de las ruedas que serán trabajados al final de este apartado de forma diferente, vamos a pasar a elaborar un pequeño programa en el que se manejen todos los servomotores, conociendo ya los valores de pulsos mínimo y máximos, y los pines a los que estos deben ser conectados.

Se pretende que, en el programa, se muevan los diferentes servos según la siguiente secuencia:

- Primero, el servo de rotación a los 0º, luego a los 180º y después a los 0º.
- Tras ello, que sean los servos de las cejas los que se muevan simltaneamente a los 0º y después a los 180º.
- Luego, el servo de la cresta se moverá a los 0º y despues de ello a los 180º.
- Finalmente, será el servo del cuello el que se mueva de 0º a y 180º.

Toda esta secuencia se repetirá continuamente (bloque "loop"). Cada una de estas acciones se realizarán tras una pausa de 2 seundos la una de la otra, como ocurrió en los ejemplos anteriores. Para evitar errores, mientras en el bucle no se esté moviendo un servo concreto, este no estrá vinculado al pin correspondiente, así en el bloque "setup" no se pndrán tales declaraciones como ocurría en ejemplos anteriores. Sólo se vincularán cada uno de los servos a sus pines cuando vayan a ser manejados, y tras ellos, estos serán desvinculados con la declaración: `servo1.detach();`

El sketch completo con todo lo comentado quedaría como se muestra acontinuación:

```
#include <Servo.h>
Servo srot;
Servo scejad;
Servo scejai;
Servo scresta;
Servo scuello;

//pulsos para servo rotacion
int PULSOMAX0=2400;
int PULSOMIN0=600;

//pulsos para servo ceja derecha
int PULSOMAX1=2000;
int PULSOMIN1=1000;

//pulsos para servos de la cresta y la ceja izquierda
int PULSOMAX23=2000;
int PULSOMIN23=1000;

//pulsos para servo del cuello
int PULSOMAX4=2500;
int PULSOMIN4=500;

void setup() {

}

void loop() {

  //PRIMERO MOVIMIENTO SERVO ROTACION
  srot.attach(A0, PULSOMIN0, PULSOMAX0);
  srot.write(0);
  delay(2000);
  srot.write(180);
  delay(2000);
  srot.write(90);
```

```
  delay(2000);
  srot.detach();


  //MOVIMIENTO SERVOS CEJA DERECHA E IZQUIERDA
  scejai.attach(5, PULSOMIN23, PULSOMAX23);
  scejad.attach(A1, PULSOMIN23, PULSOMAX23);

  scejai.write(0);
  scejad.write(0);
  delay(2000);
  scejai.write(180);
  scejad.write(180);
  delay(2000);
  scejai.detach();
  scejad.detach();


  //MOVIMIENTO SERVO CRESTA
  scresta.attach(6, PULSOMIN23, PULSOMAX23);
  scresta.write(0);
  delay(2000);
  scresta.write(180);
  delay(2000);
  scresta.detach();


  //MOVIMIENTO SERVO CUELLO
  scuello.attach(A5, PULSOMIN4, PULSOMAX4);
  scuello.write(0);
  delay(2000);
  scuello.write(180);
  delay(2000);
  scuello.detach();


}
```

Código 4.6: Ejemplo completo de programación de los 5 servos

Como era de esperar, tras ejecutar el programa y cargarlo en la placa, el comportamiento de todos los servos es correcto.

4.1.6 Anillo LED

Tras haber trabajado con los diferentes servos, pasamos ahora a la programación del siguiente componente electrónico del Currito, al anillo de 24 LEDs, el cual estrá conectado al pin 9 del Arduino.

Lo primro para poder trabajar con este anillo de Neopixel, es instalar la librería de Adafruit correspondiente:

```
#include <Adafruit_NeoPixel.h>
```

Al igual que cuando programábamos los servos, ahora debemos declarar una variable de tipo "Adafruit_Neopixel", asociada a nustra tira LED, la llamaremos "tira". Para la declaración de esta, debemos definir el numero de pixeles de nuestra tira de LEDs y el pin al que se conectará, los 2 primeros argumentos respectivamente. En cuanto al tercer argumento, como nuestra tira es RGB y además de tipo WS2812b, debemos escribir: `NEO_GRB+NEO_KHZ800`. Así, quedaría la declaración:

```
Adafruit_NeoPixel tira= Adafruit_NeoPixel(24,9,NEO_GRB+NEO_KHZ800);
```

Una vez definida la variable, debemos iniciarla, algo esencial para trabajar con esta, con la declaración:

`tira.begin();` sin escribir nada entre los paréntesis.

Para establecer un color concreto en un pixel, utilizamos la llamada:

```
tira.setPixelColor(7,90,30,0);
```

Donde el primer argumento indica el pin a encender de la tira, y los otros 3 la combinación de los 3 colores RGB en ese orden, con valores posibles de 0 a 255 (8 bites) en funcion de a intensidad de cada uno de esos 3 colores que deseemos. Con esa concreta combinación, estaríamos encendiendo el LED número 7 de color amarillo.

Para modificar el brillo de la tira se usará:

`tira.setBrightness(); y` entre paréntesis indicaremos el valor de brillo (de 0 a 255).

Finalmente, es importante el uso de la siguiente declaración, para que la tira muestre el color configurado:

```
tira.show();
```

Conociendo todo esto, ya podemos trabajar con este anillo de LEDs. Para ello elaboraremos un programa que haga lo siguiente:

- En primer lugar, estableciéndose un brillo de intensidad 70, encenderá y apagará a los 50 ms, cada uno de los 24 LEDs en orden, en color azul. Después de esto encenderá todos a la vez de color amarillo. Todo esto, lo ejecutará al principio del programa solo una vez (bucle setup).
- Después, apagará todos los LEDs y establecerá un brillo de 40.
- Encenderá todos los LEDs del color verde.
- Encenderá todos los LEDs del color rojo.
- Luego, apagará de nuevo todos los LEDs y encenderá primero los LEDs de la derecha y tras esto los de la izquierda, apagando todos entre ambas acciones.

Los 4 ultimos puntos descritos los ejecutará continuamente (bloque "loop"), con 1 segundo de pausa entrecada una de las acciones. Para el rápido y fácil encendido de los LEDs, se usarán bloques "for" para recorrer el anillo.

Así, el programa descrito se implementaría de la siguiente forma:

```
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel tira= Adafruit_NeoPixel(24,9,NEO_GRB+NEO_KHZ800);

void setup() {
tira.begin();
tira.show();

tira.setBrightness(70);
 for (int i=0;i<24;i++){
tira.setPixelColor(i,0,0,255);
tira.show();
delay(50);
tira.setPixelColor(i,0,0,0);
tira.show();
  }

for (int i=23;i>=0;i--){
tira.setPixelColor(i,90,30,0);
```

```
tira.show();
  }

delay(1000);
}

void loop() {

//apagamos todos los LEDs
for (int i=0;i<24;i++){
tira.setPixelColor(i,0,0,0);
tira.show();
  }
delay(1000);
tira.setBrightness(40);


//encendemos todos los leds de verde
for (int i=23;i>=0;i--){
tira.setPixelColor(i,0,255,0);
tira.show();
  }
delay(1000);

//encendemos todos los leds de rojo
for (int i=23;i>=0;i--){
tira.setPixelColor(i,255,0,0);
tira.show();
  }
delay(1000);

//apagamos todos los leds
for (int i=0;i<24;i++){
tira.setPixelColor(i,0,0,0);
tira.show();
  }
delay(1000);

//encendemos los leds de la mitad derecha
  for (int i=16;i<24;i++){
tira.setPixelColor(i,0,0,255);
tira.show();
  }
for (int i=0;i<4;i++){
tira.setPixelColor(i,0,0,255);
tira.show();
  }
delay(1000);

//apagamos todos los leds
for (int i=0;i<24;i++){
tira.setPixelColor(i,0,0,0);
tira.show();
  }
delay(1000);

//encendemos los leds de la mitad izquierda
   for (int i=4;i<16;i++){
tira.setPixelColor(i,0,0,255);
tira.show();
  }
```

```
delay(1000);
}
```

Código 4.7: Ejemplo completo de programación de la tira LED

Ejecutamos y corremos el programa en la placa y efectivamente se cumple lo deseado.

4.1.7 Sensor ultrasónico

A continuación, vamos a pasar a programar el sensor ultrasónico. Este contenía 4 pines: VCC, GND, ECHO y TRIGGER. Los 2 primeros sirven para la alimentación, que conectamos a los mismos del pin 2. El TRIGGER será conectado al pin digital 2 y ECHO al pin digital 3.

Con este sensor, calcularemos la distancia al objeto más cercano en la dirección del sensor. Para ilustrar el funcionamiento de este, lo usaremos conjuntamente con la tira LED, de manera que cuando un objeto se situe a una cierta distancia del sensor estos LEDs se iluminen de rojo y sino, de verde. Por lo que habrá que inicializar la tira conforme se vio anteriormente.

Primero debemos conocer como programar este sensor ultrasónico en Arduino. Para ello se utilizarán los pines TRIGGER y ECHO de este. Para conocer la distancia de un objeto, enviaremos un valor alto por el pin TRIGGER y esperaremos un resultado por el pin ECHO, del tiempo en que la onda enviada por el TRIGGER tarde en llegar de vuelta al sensor, tras el rebote en el objeto. Por lo tanto, habrá que definir el pin digital 2 como salida y el pin 3 como entrada:

```
int TRIGGER=2;
int ECHO=3;
pinMode(TRIGGER, OUTPUT);
pinMode(ECHO, INPUT);
```

Para enviar un valor alto (HIGH) por el pin TRIGGER hacemos lo siguiente al tratarse de un pin digital:

```
digitalWrite(TRIGGER, HIGH);
```

La obtencioón deltiempo por el pin ECHO lo hacemos con la siguiente llamada:

```
pulseIn(ECHO, HIGH);
```

Pero lo que nos interesa es manejar valores de distancia a la que se encuentra el objeto, algo más práctico que trabajar con tiempos. Para ello, y conociendo que la velocidad de esa onda de sonido en el aire es de 343m/s y sabiendo que el valor de tiempo que se entrega por el pin ECHO es en microsegundos, hacemos la siguiente operación para pasar ese tiempo a distancia:

$$velocidad\ de\ la\ onda = \frac{2 * d}{t} = \frac{340m}{1s} * \frac{1\ s}{1000000\ \mu s} * \frac{100\ cm}{1m} = \frac{0.034cm}{\mu s}$$

De manera que:

$$d = \frac{t(\mu s)}{\frac{1}{0.034/2}} = \frac{t}{58.8}$$

Multiplicamos por 2 la distancia porque el tiempo que recibimos por el pin ECHO, es el de ida y vuelta al objeto, es decir, el doble de la distancia a la que está el objeto, y nosotros queremos conocer la distancia del objeto no el doble. Si dividimos entre 58.8 el tiempo obtenido por el pin ECHO, tendremos la distancia del objeto en centímetros.

Con esto comprendido, a la hora de elaborar el código será lo siguiente o que debemos escribir para clacular la distancia del objeto más próximo:

```
digitalWrite(TRIGGER, HIGH);
delay(1);
digitalWrite(TRIGGER, LOW);
```

```
DURACION= pulseIn(ECHO, HIGH);
DISTANCIA=DURACION/58.8;
```

El delay de 1 segundo lo recomienda el fabricante como tiempo de envio de la señal por el TRIGGER.

Como ejemplo, elaboraremos un programa en el que haremos lo siguiente:

- Encenderemos en prier lugar los LEDs de la tira de color verde (en el bloque setup), uno a uno cada 70ms.
- Tras esto, y en el bucle "loop", calcularemos la distancia del objeto más cercano.
- Después, evaluaremos el valor de esa distancia en un bucle "while" de manera que, si esa distancia es menor de 20cm, se enciendan las luces de color rojo.
- Si esa distancia vuelve a ser mayor que 20cm se volverán a encender los LEDs de color verde, volviéndose a repetir el ciclo.

Implementando lo comentado en el IDE de Arduino nos quedaría lo siguiente:

```
#include <Adafruit_NeoPixel.h>
Adafruit_NeoPixel tira= Adafruit_NeoPixel(24,9,NEO_GRB + NEO_KHZ800);

int TRIGGER=2;
int ECHO=3;
int DURACION;
int DISTANCIA;

void setup() {
  pinMode(TRIGGER, OUTPUT);
  pinMode(ECHO, INPUT);
  tira.begin();

//encendemos leds en verde
tira.setBrightness(60);
  for (int i=23;i>=0;i--){
tira.setPixelColor(i,0,255,0);
tira.show();
delay(70);
  }

}

void loop() {

  //calculo la distancia del objeto
  digitalWrite(TRIGGER, HIGH);
  delay(1);
  digitalWrite(TRIGGER, LOW);
  DURACION= pulseIn(ECHO, HIGH);
  DISTANCIA=DURACION/58.8;

  while(DISTANCIA<=20 && DISTANCIA>=0){

  //si el objeto esta a menos de 20cm, enciendo los leds de rojo
  for (int i=23;i>=0;i--){
    tira.setPixelColor(i,255,0,0);
    tira.show();
  }

  //vuevo a calcular la distancia del objeto pra ver si salgo del bucle
  digitalWrite(TRIGGER, HIGH);
  delay(1);
  digitalWrite(TRIGGER, LOW);
  DURACION= pulseIn(ECHO, HIGH);
```

```
  DISTANCIA=DURACION/58.8;
}

//si salgo del bucle vuelvo a poner los leds en verde
  for (int i=23;i>=0;i--){
tira.setPixelColor(i,0,255,0);
tira.show();
  }
}
```

Código 4.8: Ejemplo completo de programación del sensor ultrasónico

Compilando el programa y corriéndolo en la placa, este se comporta de forma adecuada como esperábamos.

4.1.8 Micrófono

Pasamos ahora al manejo del micrófono KY-038 del animatrón. Este se conecta al pin digital 7 y al ser un dispositivo que evalúa sonidos del exterior, se debe definir como sigue:

```
int MICRO=7;
pinMode(MICRO, INPUT);
```

Al igual que el caso anterior con el sensor ultrasónico, el microfono lo vamos a ilustrar conjuntamente con la tira LED de manera que cuando este reciba un sonido por encima del umbral establecido en su potenciómetro, se enciendan las luces de color rojo, sino permanecerán verdes. Por lo que habrá que inicializar la tira conforme se vio anteriormente.

Para leer el valor que percibe el micrófono usamos, al tratarse de un pin digital mediante el cual estamos manejando a este, lo siguiente:

```
VALOR=digitalRead();
```

Especificándose entre paréntesis el pin al cual está conectado el microfono (7);

Conociendo esto, ya podemos pasar a elaborar un programa manejando este dispositivo.

El programa ejemplo que elaboraremos para probar el funcionamiento del micrófono ejecutará o siguiente:

- Primero, y en el bloque "setup", encenderá los LEDs del anillo a brillo 70, de color azul cada uno 40ms después de apagar el anterior, y tras esto, encenderá en sentido contrario y de igual forma, pero con un delay de 70ms en vez de 40ms, cada uno de lo LEDs de color verde.
- A continuacion, en el bloque "loop", se recoge el valor del microfono en la variable VALOR y se evalua si está esta en valor alto con respecto al umbral establecido en el potenciómetro y, si es así, se encenderán los LEDs de color rojo, sino permanecerán de color verde.

Queda, por tanto, la implementación del programa especificado, tal y como se muestra a continuación:

```
#include <Adafruit_NeoPixel.h>
Adafruit_NeoPixel tira= Adafruit_NeoPixel(24,9,NEO_GRB + NEO_KHZ800);

int MICRO=7;
int VALOR;

void setup() {
tira.begin();
tira.show();
pinMode(MICRO, INPUT);

tira.setBrightness(70);
 for (int i=0;i<24;i++){
tira.setPixelColor(i,0,0,255);
tira.show();
delay(40);
```

```
tira.setPixelColor(i,0,0,0);
tira.show();
  }

for (int i=23;i>=0;i--){
tira.setPixelColor(i,0,255,0);
tira.show();
delay(70);
  }
}

void loop() {
VALOR=digitalRead(MICRO);

if(VALOR==HIGH){
  for (int i=23;i>=0;i--){
    tira.setPixelColor(i,255,0,0);
    tira.show();
  }
}
 else{
    for (int i=23;i>=0;i--){
    tira.setPixelColor(i,0,255,0);
    tira.show();
  }
 }
}
```

Código 4.9: Ejemplo completo de programación del micrófono

Verificamos y subimos el programa a la placa Arduino y observamos como efectivamente, se cumple lo deseado, mientrs el valor de ruido que hagamos supere el umbral, la tira led se ilumina de rojo, sino permanece verde.

4.1.9 Altavoz

A continuacion, será el turno del mini altavoz mp3 del Currito, el cual conectamos al pin digital 8 del Arduino como se indicó anteriormente. En este caso lo debemos definir como salida dicho pin:

```
pinMode(8, OUTPUT);
```

Para manejar este altavoz simplemente haremos uso de la funcion "tone":

```
tone(8,440,150);
```

Donde el primer argumento es el pin al que esta conectado nuestro altavoz, el segundo la frecuencia en hercios a reproducir por este, y el tercero la duración de la nota.

También es interesante la funcion `noTone(8)`, cuyo argumento es el pin al que tenemos conectado el altavoz y sirve para pausar el sonido en el altavoz.

Como programam de jemplo para el altavoz vamos a intentar reproducir el principio de la melodía de la Marcha Imperial de Star Wars, la cual se ejcutará continuamente cada 2 segundos (bucle "loop"):

```
int sound=8;

void setup(){
pinMode(8, OUTPUT); // Definimos el pin 8 como salida.

}

void loop(){

 //melodia star wars
```

```
  tone(8,440,250);
  delay(250);
  noTone(8);
  tone(8,440,250);
  delay(250);
  noTone(8);
  tone(8,440,250);
  delay(250);
  tone(8,349,175);
  delay(175);
  tone(8,523,75);
  delay(75);
  tone(8,440,250);
  delay(250);
  tone(8,349,175);
  delay(175);
  tone(8,523,75);
  delay(75);
  tone(8,440,440);
  delay(400);
  noTone(8);

delay(2000);
     }
```

Código 4.10: Ejemplo completo de programación del altavoz

Tras varias pruebas, al ejecutar este programa se consigue obtener un sonido muy similar al que deseábamos.

4.1.10 Ruedas

Las 2 ruedas serán impulsadas por 2 servomotores HXT12K M 11kg, pero en este caso, y a diferencia de los usados en la cresta o para la rotación del cuerpo, estos si pueden girar 360º.

Para estas ruedas se usarán los 4 conoctores (para 2 mototres) del motor shield. Esos 4 conectores, se manejan con los pines digitales 10, 11, 12 y 13 del Arduino. Los pines 10 y 11 se encargarán de la velocidad de ambas ruedas, mientras que los pines 12 y 13 controlarán el sentido de rotación de las ruedas.

Debemos definir entonces dichos pines como salidas:

```
 int MOTOR1_DIRECCION_PIN = 12;
 int MOTOR2_DIRECCION_PIN = 13;
 int MOTOR1_VELOC_PIN = 10;
 int MOTOR2_VELOC_PIN = 11;

 pinMode(MOTOR1_DIRECCION_PIN, OUTPUT);
 pinMode(MOTOR2_DIRECCION_PIN, OUTPUT);
 pinMode(MOTOR1_VELOC_PIN, OUTPUT);
 pinMode(MOTOR2_VELOC_PIN, OUTPUT);
```

Como los pines 10 y 11 controlan la velocidad (de 0 a 255) de los servos, usaremos “analogWrite” para dichos pines. Para el caso de los pines 12 y 13, indicaremos simplemente con “digitalWrite” los valores “HIGH” o “LOW” en funcion de si queremos que las ruedas giren hacia delante o hacia detrás respectivamente.

Conocido esto, elaboraremos un programa de ejmplo en el que las ruedas girarán haca delante a velocidad 100 durante 2 segundos, tras esto se paren, y luego vuelvan a ponerse en marcha, pero esta vez en sentido contrario durante otros 2 segundos para que finalmente, tras ello, se vuelvan a parar. Esto lo ejecutará continuamente (bucle “loop”):

```
int MOTOR1_DIRECCION_PIN = 12;
int MOTOR2_DIRECCION_PIN = 13;

int MOTOR1_VELOC_PIN = 10;
```

```
int MOTOR2_VELOC_PIN = 11;

int para = 0;
int velocidad = 100;

void setup() {

  pinMode(MOTOR1_DIRECCION_PIN, OUTPUT);
  pinMode(MOTOR2_DIRECCION_PIN, OUTPUT);
  pinMode(MOTOR1_VELOC_PIN, OUTPUT);
  pinMode(MOTOR2_VELOC_PIN, OUTPUT);


}

void loop() {

  digitalWrite(MOTOR1_DIRECCION_PIN,HIGH);
  digitalWrite(MOTOR2_DIRECCION_PIN,HIGH);

  analogWrite(MOTOR1_VELOC_PIN,velocidad);
  analogWrite(MOTOR2_VELOC_PIN,velocidad);

  delay(2000);

  analogWrite(MOTOR1_VELOC_PIN,para);
  analogWrite(MOTOR2_VELOC_PIN,para);

  delay(2000);

  digitalWrite(MOTOR2_DIRECCION_PIN,LOW);
  digitalWrite(MOTOR1_DIRECCION_PIN,LOW);

  analogWrite(MOTOR1_VELOC_PIN,velocidad);
  analogWrite(MOTOR2_VELOC_PIN,velocidad);

  delay(2000);

  analogWrite(MOTOR1_VELOC_PIN,para);
  analogWrite(MOTOR2_VELOC_PIN,para);

  delay(2000);

}
```

Código 4.11: Ejemplo completo de programación de las ruedas

4.1.11 Cámara Raspbrerry Pi

Para la programacion de la cámara, vamos a trabajar en el IDE Thonny Python, conectando está al puerto correspondiente de la Raspberry Pi 4B, como se indicó anteriormente.

Lo primero para poder manejar la cámara en Raspbian, es activarla desde el menú configuración del mismo, en el apartado de "interfaces". Tras ello ya podemos pasar a trabajar en el IDE. La idea será elaborar un programa en el que podamos realizar fotos y videos con esta cámara.

Para comenzar, debemos importar la librería para manejar la cámara de la Raspberry:

```
from picamera import PiCamera
```

Aparte de esta librería, debemos importar algunas más como la de tiempo "time", ya que vamos a utilizar pausas entre fotos.

Para almacenar las fotos en memoria debemos establecer un nombre y dirección en el sistema. Las librerías del sistema operativo y la de fecha, "os" y "datetime" respectivamente nos serán útiles para esto. Dicho destino y nombre de las fotos y videos lo estableceremos de la siguiente forma:

```
destino='/home/pi/Videos'
destino2='/home/pi/Pictures'

nombrefoto=os.path.join(destino2,dt.datetime.now().strftime('%Y-%m-
%d_%H:%M:%S.jpg'))
nombrevideo=os.path.join(destino,dt.datetime.now().strftime('%Y-%m-
%d_%H:%M:%S.h264'))
```

La foto la guardaremos en la carpeta "Pictures" del sistema y el video en la carpeta "Videos" del mismo. Para el nombre hemos elegido guardarlos con el nombre de la fecha actual en que se tomarán cada uno, lo cual hacemos, como se puede ver, a taves de la librería "datetime" (con "datetime.now()") y lo fusionamos con "os.path.join" con el destino establecido comentado. Es importante establecer un formato de guardado correcto: ".jpg" para la foto y ".h264" para el video en este caso.

Para iniciar la cámara, usando la librería "picamera" debemos hacer lo siguiente, donde la variable camararepresenta a la misma:

```
camara=PiCamera()
```

Para tomar una captura con ella, se utiliza la siguiente declaración:

`camara.capture()`, cuyo argumento entre paréntesis será el nombre y destino que le queramos dar.

Para grabar un video parar la grabación del mismo se utilizan las siguientes llamadas respectivamente:

```
camara.start_recording()
camara.stop_recording()
```

Sin embargo, antes de realizar cualquiera de estas acciones debemos iniciar la visualización de la cámara con la llamada:

```
camara.start_preview()
```

y tras realizar dichas foto imágenes, haremos lo contrario con:

```
camara.stop_recording()
```

Conocido esto, pasamos a elaborar un programa ejemplo en el que se grabará un video y se tomarán fotos durante la realización de este:

- Se tomarán 2 fotos concretamente, con 5 segundos de diferencia entre ellas y tras 5 segundos de iniciar el video la primera de ellas.
- Después de 5 segundos de tomar la segunda foto, se parará el video y con ello la visualización de la cámara.
- La acción de grabar se realizará mediante 2 funciones, una que iniciará la previsualización de la cámara junto con la grabación, y otra para lo contrario, cuyos nombres serán respectivamente: "grabo_video()" y "paro_video()"

Con ello el programa a simlar quedaría como sigue:

```
import os
import datetime as dt
from picamera import PiCamera
from time import sleep

destino='/home/pi/Videos'
destino2='/home/pi/Pictures'

camara=PiCamera()

nombrefoto=os.path.join(destino2,dt.datetime.now().strftime('%Y-%m-
%d_%H:%M:%S.jpg'))


def grabo_video():
        nombrevideo=os.path.join(destino,dt.datetime.now().strftime('%Y-%m-
%d_%H:%M:%S.h264'))
        camara.start_preview()
        camara.start_recording(nombrevideo)


def paro_video():
        camara.stop_recording()
        camara.stop_preview()

grabo_video()
sleep(5)
camara.capture(nombrefoto)
sleep(5)
nombrefoto=os.path.join(destino2,dt.datetime.now().strftime('%Y-%m-
%d_%H:%M:%S.jpg'))
camara.capture(nombrefoto)
sleep(5)
paro_video()
```

Código 4.12: Ejemplo completo de programación de la cámara Raspberrry

Tras la compilación y ejecución de este, comprobamos que funciona de manera correcta. Una de las fotos tomadas es la siguiente que, como observamos, es de gran calidad:

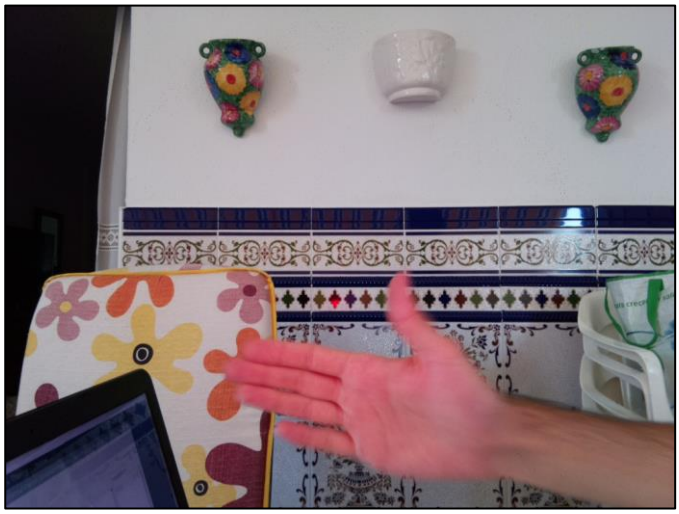

Figura 4.1: Foto de prueba con cámara Raspberry

4.2 Comunicación Arduino y Raspberry Pi por puerto serie

El último de los aspectos a tratar, va a ser la comunicación de Raspberry Pi y Arduino por puerto serie. Para ello conectaremos el USB tipo-B del Arduino con cualquiera de los USB tipo-A de la Raspberry Pi.

Por un lado, en cuanto a Arduino, para iniciar y trabajar con el puerto serie debemos utilizar la siguiente declaración:

```
Serial.begin(9600);
```

Donde el argumento indica la velocidad de transmisión de datos, 9600 bits por segundo en este caso. Dicho valor debe ser igual en ambos extremos para evitar problemas de comunicación. Habrá que tener, además, especial cuidado en leer y enviar el dato con el mismo tipo.

Para escribir datos en dicho puerto se usará:

```
Serial.println();
```

Cuyo argumento será aquello que queramos enviar. Se usaría `Serial.read();` para leer un dato.
Por otro lado, en cuanto a la Raspberry, en el IDE Thonny Python, para trabajar con el puerto serie, debemos importar la librería correspondiente:

```
import serial
```

Para configurar el inicio de dicho puerto en el programa, debemos hacer lo siguiente:

```
serial.Serial('/dev/ttyACM0',9600)
```

donde especificamos entre paréntesis el puerto donde está conectado el Arduino y la velocidad de transmisión de datos (9600 bits por segundos como en el otro extremo en el Arduino). Para conocer dicho puerto, podemos utilizar el comando siguiente en el terminal de Raspbian: *`dmesg | grep -v disconnect | grep -Eo "tty(ACM|USB)." | tail -1`*

Para leer un dato por dicho puerto serie, se usa el comando: `lectura=com.readline()`

Lo cual tendremos que decodificar en formato legible de la siguiente forma, a formato 'utf-8':

`ine=lectura.decode('utf-8').strip()` (con "strip()", se eliminan caracteres extras innecesarios)

Conviene, además, borrar todo aquello que haya quedado almacenado en el puerto serial y que no nos interese, antes de leer cualquier dato. Esto lo hacemos con la siguiente llamada:

```
com.flushInput()
```

Conocido todo esto a cerca del manejo del puerto serie tanto en el Arduino como en la Raspberry Pi, podemos pasar a elaborar un programa de ejemplo en el que comunicaremos ambos dispositivos.

La idea que se quiere mostrar, es utilizar el micrófono conectado al Arduino y cuando este reciba un valor alto del exterior, comunicarse por puerto serie con la Raspberry Pi y que esta utilice su cámara para hacer una foto. Para ello la Raspsberry Pi esperará en un bucle "While true" (bucle while infinito) a la espera de ese valor alto por el puerto serie.

El programa a elaborar en el extremo de la Raspberry es el siguiente entonces:

```
import serial
import os
import datetime as dt
from picamera import PiCamera
from time import sleep

#dmesg | grep -v disconnect | grep -Eo "tty(ACM|USB)." | tail -1
destino='/home/pi/Pictures/Obstaculos'
camara=PiCamera()

com=serial.Serial('/dev/ttyACM0',9600)
com.flushInput()

while True:
    lectura=com.readline()
    line=lectura.decode('utf-8').strip()
    print(line)

    if line=="1":
        nombrefoto=os.path.join(destino,dt.datetime.now().strftime('%Y-%m-
%d_%H:%M:%S.jpg'))
        camara.start_preview()
        sleep(1)
        camara.capture(nombrefoto)
        camara.stop_preview()
```

Código 4.13: Ejemplo de programación por puerto serie, extremo de la Raspberry

En el otro extremo de la comunicación el programa a implementar en el Arduino comentado debe quedar como sigue:

```
int MICRO=7;
int valormicro;

void setup() {

  pinMode(MICRO,INPUT);
  Serial.begin(9600);

}

void loop() {

valormicro=digitalRead(MICRO);
Serial.println(valormicro);
delay(1);

}
```

Código 4.14: Ejemplo de programación por puerto serie, extremo del Arduino

Al verificar y correr ambos programas, se comprueba una correcta comunicación de ambos dispositivos por puerto serie.

4.3 Elaboración programa esquiva-obstáculos

Una vez tratados cada uno de los componenetes de manera independiente, se va pasamos a elaborar el programa principal del animatrón, en el que se trabaje con todos ellos y, además, exista comunicación entre Raspberry Pi y Arduino.

Comenzaremos explicando la estructura del programa en Arduino, diferenciando bloque "setup" y bloque "loop", y posteriormente pasaremos describir el programam a desarrollar en la Raspberry.

En el bloque "setup" del sketch del programa en Arduino se pretende que en el animatrón se lleven a cabo las siguientes tareas:

- Tras la declaración yconiguracion de las variables y pines necesarios, se moverá las cejas hacia abajo, la cresta hacia detrás y el cuello hacia abajo, quedandose el currito en posición de espera a iniciarse. Dichas acciones las recogemos en la funcion "void finalizando()", que se mostrará posteriormente.
- En dicha posición, esperará la recepción de un valor alto por el micrófono lo suficientemente agudo (un chasquido de dedos o aplauso) para iniciarse.
- Una vez se reciba dicho valor, el Arduino se comunicará con la Raspberry y esta comenzará a grabar con la cámara.
- Después, encenderá el anillo LED y hará sonar una de las 3 pequeñas melodías elaboradas, a través del altavoz, de forma aleatoria cada vez que se inicie, las cuales son un de los temas de las bandas sonoras de las 3 películas: Star Wars, Piratas Del Caribe o Harry Potter. Dichas melodías se recojen en las 3 funciones siguientes: "void sw()", "void pdc()" y "void hp()" respectivamente.
- Tras ello, se ejecutará la función elaborada "void despierta()", la cual hará que el Currito, mire a su alrededor moviendo el servo de rotación, mueva la cabeza hacia arriba a través del servo del cuello, y mueva la cresta.

Por otro lado, en cuanto a lo que vamos a desarrollar en el bucle "loop" contnuamente tenemos:

- Calcularemos la distancia del objeto más cercano a través del sensor ultrasónico y en función de esta entraremos e uno de los 2 bloques "if" creados:
    - Si esa distancia es mayor de 20cm, encenderemos el anillo LED de color verde, llevaremos la cresta hacia detrás y las cejas hacia abajo, y comenzará a avanzar hacia adelante a través de los servos de las ruedas, ejecutándose para ello la funcion elaborada para ello: "void avanza()"
    - Si esa distancia es menor de 20cm (nos hemos encontrado un obstáculo), se parará el movimiento de las ruedas, llevaremos las cejas hacia detrás la cresta hacia delante y se pondrán los LEDs de color rojo. Tras ello, el robot retrocederá (funcion"void retrocede()") observará la zona mediante la rotación de su cuerpo y girará de forma aleatoria hacia uno de los 2 lados moviendo para ello sus 2 ruedas en sentido contrario (proceso recogido en las funciones "void giradcha()" y "void giraizqda()"). Además, se comunicará con la Raspberry Pi por puerto serie y esta, con el uso de su cámara tomara una foto del obstáculo.
- Durante la ejecución de la funcion "retrocede", los LEDs se encenderán de color azul. Cuando este girando hacia uno de los lados estos se iluminarán de color azul, pero solo la mitad hacia la que girará Currito.
- Antes de pararse o volverse a iniciar en funcion del resultado de la distancia del objeto más cercano, reproducirá por los altavoces una pequeña melodía (de tono grave para cuando vaya a parar y tono agudo para cuando vaya a ponerse en marcha).
- Si durante la ejecución de este bucle, el numero de fotos tomadas/obstáculos encontrados llegase a 5 o el tiempo desde el que se comenzóa a grabar con la cámara llegase a 80 segundos, será la Raspberry Pi la que se comunique con el Arduino y esteejecutará la funcion "void finalizando()" de manera que quede en posición de espera a que vuelva a recibir un valor alto por el micrófono y que vuelva a

realizar las mismas operaciones, tanto el programa Arduino como el de la Raspberry Pi.

Se han incluido en el programa varias variables banderas o "flags" que hacen más eficiente el mismo:

- Flag1: evita que, mientras la distancia del objeto más cercano sea mayor de 20cm, no esté entrando continuament en el bloque "if" correspondiente y ejecutando las mismas tareas, solo active las ruedas y luces de color verde una sola vez.
- Flag2: evita que, como al terminar el bloque "setup", la posición de los servos de la cresta y cejas queda igual que cuando la orden es moverse hacia adelante, no haga esto 2 veces seguidas si al iniciar al Currito este no tiene ningún obstáculo delante.

Cabe destacar un último aspecto en relación a las librerías utilizadas. En este caso, y a diferencia de los ejemplos en los que se trabajaba con servomotores, no vamos a usar la librería "servo.h". esto será así, porque al utilizar dicha librería se desactia la función PWM de los pines digitales 9 y 10, y en el pin 10 necesitamos dicha función (analogWrite() para la velocidad de las ruedas). Por lo tanto, el movimiento a un cierto ángulo de cada uno de los servomotores se realizará de forma manual con un bucle for, como se muestra a continuación:

```
   for (int Hz =0; Hz < 50 ;Hz++){       // repetimos la instruccion 50 veces
     digitalWrite (6,HIGH);
     delayMicroseconds(PULSOMAXCTA);                 // llevamos a 0°la cresta
(hacia detras)
     digitalWrite (6,LOW);
     delay(10);
 }
```

Es decir, enviamos 50 pulsos al servomotor (trabaja a 50Hz), de longitud igual al especificado, en microsegundos, entre las declaraciones de pulso alto y pulso bajo, con envio de dichos pulsos cada 10 milisegundos.

En relación a la Raspberry y, como se ha podido deducir a partir de lo comentado más arriba, se pretende realiza un programa que haga lo siguiente:

- En un bucle "while True", espere hasta recibir la orden grabación por parte del Arduino, a través del puerto serie, y en ese instante comenzar la grabación del video del recorrido del Currito.
- Tras ello saldrá de tal bucle, y en otro bucle while True, esperaráa que el valor calculado por el sensor ultrasónico (enviado también por puerto serie), sea menor de 20, y entonces tomará una foto.
- Cuando el nuemro de fotos llege a 5, o el timepo desde que se comenzó la grabación sea mayor de 80 segundos, sladrá de este segundo bucle while true, notificando antes al Arduino para que este ejecute la funcion "finalizando".
- Entonces, volveremos al primero de los bucles while, esperando de nuevo la notificación por parte del Arduino para comenzar a grabar de nuevo.

Una vez comentado todo esto, pasamos a la elaboración de estos tanto en el IDE de Arduino como en el de Thonny Python. Por su gran extensión, la elaboración final de estos se puede ver en el Anexo III de esta memoria.

Ejecutando ambos programas, el funcionamiento del currito es correcto, realizando las fotos y videos de forma adecuada, esquivando los obstáculos con gran acierto, y con una comunicación entre la Raspberry Pi y el Arduino Uno correcta.

Con la elaboracion de este programa principal, damos por concluido el trabajo de programación en este proyecto.

5 CONCLUSIONES Y VALORACIONES

"El tiempo es el mejor autor:
siempre encuentra un final perfecto.".
- Charles Chaplin -

Durante el desarrollo de este proyecto, se han adquirido ciertos conocimientos, y se han logrado una serie de objetivos, entre los que caben destacar los siguientes:

- Se ha logrado, como concepto general, que el diseño de Currito, interactue y se comunique con el entorno lográndose cada uno de los objetivos concretos que se pretendían.
- Se ha conseguido poner en funcionamiento y controlar cada uno de los dispositivos electrónicos y mecánicos que contiene el robot-animatrón, logrando un profundo aprendizaje de los entornos Arduino y Raspbian y de los lenguajes utilizados en esos entornos.
- Se ha logrado con éxito la conexión mediante puerto serie de los entornos Arduino y Raspberry Pi, obteniéndose, como consecuencia, un mayor acercamiento de Currito al entorno que le rodea.

Para un mejor comportamiento del robot en el entorno, y en funcion de los inconvenientes observados durante el desarrollo de este trabajo, convendría incluir un mayor número de sensores de distancia alrededor del cuerpo de este, al ser un artefacto destinado a moverse por el ambiente que le rodea y ello, aportaría ma´s infrmacion de este. Como consecuencia se podrían desarrollar tareas de SLAM o mapeo del entorno con mayor facilidad como futuros proyectos.

Otra característica a destacar, para un mejor funcionamiento, sería la superficie de contacto de Currito con el suelo, medio por el que se moverá a través de las ruedas. Al tratarse de una superficie de plástico, en suelos algo deslizantes no se desenvuelve tan fácilmente como en otros que no lo sean.

Para tratar esto, se han incluido unas finas cintas de goma en la superficie que permiten mayor movilidad en dichos terrenos, con mayor agarre, y cuya inclusión no supone de forma notable aumentos de peso o problemas de espacio en el animatrón.

En definitiva, se trata de un proyecto que abarca diversas materias (robótica, electrónica, informática…) y permite multiples apmpliaciones de futuro al contener 2 contoladores (Arduino y Raspberry) que permiten gran variedad de tareas con los dispositivos que equipa.

6 ANEXOS

En esta sección se van a incluir una serie de anexos a este trabajo que aportan información adicional relevante y amplian la materia tratada en el mismo.

6.1 Anexo I: Puesta en marcha de Raspberry Pi 4B y conexión remota

En este primer anexo, se va a tratar todo aquello necesario para poner en marcha y trabajar con la Raspberry Pi 4B, además de como usarla de forma remota.

6.1.1 Intalacion Sistema operativo

En la Raspberry Pi se pueden instalar diferentes sistemas operativos (Linux, Ubuntu, Raspbian…). En nuestro caso, se ha elegido Raspbian al ser el oficial del fabricante el cual podremos descargar en su página web [21].

Para la instalacion del sistema operativo necesitamos una tarjeta SD formateada, de capacidad suficiente según el proyecto, en nuestro caso con 16GB (la versión completa del sistema operativo ocupa unos 8GB).

Necesitaremos un software para instalar el sistema operativo en la Raspberry Pi, el cual también se puede encontrar en la pagina web oficial del fabricante.

Una vez instalado en la tarjeta SD mediante dicha aplicación, ya solo tendremos que introducirla en la ranura para tarjetas SD, en la parte posterior de la placa y ya podremos usarla.

Para manejarnos por dicho SO, necesitaremos mostrar el interfaz por algún monitor, para lo cual necesitaremos conectar la Rasoberry Pi a alguno mediante un cable micro HDMI. También podremos manejarla de forma remota, para lo que necesitaremos unicamene darle alimentación y conexión a internet.

6.1.2 Conexión remota.

Esto nos permitirá su manejo dede otro ordenador. Para ello, una vez encendida y conectada internet, necesitaremos un software extra según queramos conectarla por un método u otro.

Vamos a tratar 2 posibles métodos, aunque existen más: SSH y VNC.

Para ambos métodos debemos activar las correspondientes funciones SSH y VNC desde el apartado interfaces de la configuración de la Raspberry Pi.

6.1.2.1 SSH

Este tipo de conexion remota, Secure Shell, nos permite manejar remotamente el terminal de Raspbian, a través del cual podremos manejar todas las funciones de la placa.

Para ello necesitamos descargar el software “Putty”, a través del cual podremos conectarnos mediante tal método conociendo la dirección IP de nuestra Raspberry Pi y la contraseña y usuario de la misma [22].

6.1.2.2 VNC

Este método, Virtual Network Computing, nos permite manejar en su totalidad, la interfaz de la Raspberry Pi, como si la tuviésemos conectada a un monitor.

Para ello necesitaremos lo mismo que en el método anterior, conocer la IP de la Raspberry Pi, el usuario y contraseñas de la misma y un software extra, en este caso: "realVNC" [23].

6.2 Anexo II: Presupuesto del material utilizado

En este segundo anexo se va a detallar los presupuestos de cada uno de los componentes del animatrón, tanto mecánicos como electrónicos. A su vez se diferenciarán en estos, cuales correspondena cada una de las partes del Currito (cabeza, cuello y cuerpo).

6.2.1 Componentes electrónicos:

Elemento	Descripción	Precio/Ud. (euros)	Número de Uds.	Precio total (euros)
Cabeza				***30.14***
Anillo LED	Anillo Sparkfun de x24 neopixel LED RGB	13	1	13.00
Cámara Raspberry	Módulo de cámara raspberry Pi V2 8MP	27.14	1	27.14
Cuerpo				***99,88***
Arduino Uno	Placa para control de sensores y actuadores	22.00	1	22.00
Interruptor	Interruptor Basculante 3A 250V On/Off 2 pines	2.00	1	2.00
Micrófono	Módulo amplificador ajustable modeo KY-038	4.49	1	4.49
Mini altavoz mp3	Speaker de 8 Ohm, 40mm de diámetro y 2 W	2.00	2	4.00
Motor Shield L298P	Módulo controlador para manejo de motores	8.00	1	8.00
PIR proximidad	Sensor ultrasónico HC-SR04	1.50	1	1.50
Raspberry Pi 4B	Mini ordenador de 1.5GHz quad-core y 2GB RAM	55.00	1	55.00
Sensor Shield V5	Escudo para conexión de múltiples sensores	2.89	1	2.89
Total				***130,02***

6.2.2 Componentes mecánicos:

Elemento	Descripción	Precio/Ud. (euros)	Número de Uds.	Precio total (euros)
Cabeza				***36.91***
Eje acero templado	Eje de acero templado de 8mm de diámetro	1.75	1	1.75
Rodamiento 608ZZ	Rodamientos de la cara	1.10	4	4.40
Rotula M3	Rotula M3 con agujero de 3mm	1.00	4	4.00
Servo Futaba 3003	Servomotor para la cresta y la boca	6.98	2	13.96
Servo Turnigy TGY 50090	Servomotor analogico para ambas cejas	5.40	2	10.80
Tornillo M3x10	Tornillos allen de cabeza avellanada	0.20	6	1.20
Tornillo M3x16	Tornillos allen de cabeza avellanada	0.20	4	0.80
Cuello				***20.20***
Eje acero templado	Eje para rodamientos del cuello, de 44mm	1.00	2	2.00
Eje acero templado	Eje para rodamientos del cuello, de 14.5mm	1.00	2	2.00
Rodamiento 608ZZ	Rodamientos para el movimiento del cuello	1.10	6	6.60
Servo HXT12K M 11kg	Servomotor para el movimiento del cuello	9.00	1	9.00
Tornillo M3x14	Tornillos allen de cabeza avellanada	0.20	3	0.60
Cuerpo				***36.20***
Eje hacer templado	Eje para el giro del cuerpo, de 20mm	1.00	1	1.00
Insertos de metal M3	Insertos metálicos para roscado en plástico ABS	0.10	14	1.40
Rodamiento 608ZZ	Rodamientos para la rotación del cuerpo	1.10	2	2.20
Rueda loca	Rueda trasera de 30mm de diámetro	1.00	1	1.00
Separadores hexagonales	Separadores M3 50mm para sujección Arduino	0.30	4	1.20
Servo HXT12K M 11kg	Servomotor para rotación del cuerpo y ruedas	9.00	3	27.00
Tornillo M3x12	Tornillos allen de cabeza avellanada	0.20	3	0.60
Tornillo M3x16	Tornillos allen para sujección de Raspberry Pi	0.20	4	0.80
Tornillo M3x18	Tornillos allen de cabeza avellanada	0.20	3	0.60
Tornillo M3x20	Tornillos allen de cabeza avellanada	0.20	2	0.40
Total				***93.31***

6.3 Anexo III: Código completo programa esquiva-obstáculos

6.3.1 Parte I: Arduino

```
#include <Adafruit_NeoPixel.h>
Adafruit_NeoPixel tira= Adafruit_NeoPixel(24,9,NEO_GRB + NEO_KHZ800);

//definicion variables necesarias
int PULSOMAXROT=2400;
int PULSOMINROT=600;

int PULSOMAXCEJAD=1800;
int PULSOMINCEJAD=1200;

int PULSOMAXCEJAI=1800;
int PULSOMINCEJAI=1000;

int PULSOMAXCLL=2500;
int PULSOMINCLL=500;

int PULSOMAXCTA=2300;
int PULSOMINCTA=600;

int MOTOR1_DIRECCION_PIN = 12;
int MOTOR2_DIRECCION_PIN = 13;

int MOTOR1_VELOC_PIN = 10;
int MOTOR2_VELOC_PIN = 11;

int velocidad=110;
int velocidad2=100;
int girorandom;

int TRIGGER=2;
int ECHO=3;
int DURACION;
int DISTANCIA;

int MICRO=7;
int VALORMICRO;
int ALTAVOZ=8;
int melodiarandom;

int x=2;
int flag1=0;
int flag2=0;

void setup() {

  //inicializacion de pines y puerto serie
  Serial.begin(9600);

  pinMode(MOTOR1_DIRECCION_PIN, OUTPUT);
  pinMode(MOTOR2_DIRECCION_PIN, OUTPUT);
  pinMode(MOTOR1_VELOC_PIN, OUTPUT);
  pinMode(MOTOR2_VELOC_PIN, OUTPUT);

  pinMode (6, OUTPUT);
  pinMode (5, OUTPUT);
  pinMode (A0, OUTPUT);
  pinMode (A1, OUTPUT);
  pinMode (A5, OUTPUT);
```

```
  pinMode(TRIGGER, OUTPUT);
  pinMode(ECHO, INPUT);
  tira.begin();
  tira.show();
  pinMode(ALTAVOZ, OUTPUT); // Definimos el pin 8 como salida.
  pinMode(MICRO, INPUT);


finalizando();

//espero a recibir un valor alto por el microfono para iniciar el programa
while(x!=1){
  VALORMICRO=digitalRead(MICRO);
  if(VALORMICRO==HIGH){
    Serial.println("inicio");
    Serial.println("grabacion");
    x=1;
  }
}



tira.setBrightness(70);
 for (int i=0;i<24;i++){
tira.setPixelColor(i,0,0,255);
tira.show();
delay(40);
tira.setPixelColor(i,0,0,0);
tira.show();
  }



randomSeed(analogRead(A2)); //valor aleatorio para reproducir cualquiera de
las 3 melodias
melodiarandom=random(1,4);

if(melodiarandom==1){
  for (int i=23;i>=0;i--){
tira.setPixelColor(i,0,255,0); //color verde
tira.show();
delay(70);
  }
sw(); //melodia star wars

}

if(melodiarandom==2){
    for (int i=23;i>=0;i--){
tira.setPixelColor(i,90,30,0); //color amarillo-marron
tira.show();
delay(70);
  }
pdc(); //melodia piratas del caribe

}
if(melodiarandom==3){
    for (int i=23;i>=0;i--){
tira.setPixelColor(i,75,0,75); //color morado-rosa
tira.show();
delay(70);
```

```
  }

hp(); //melodia de harry potter

}

despierta();

}
void loop() {

 digitalWrite(TRIGGER, HIGH);
 delay(1);
 digitalWrite(TRIGGER, LOW);
 DURACION= pulseIn(ECHO, HIGH);
 DISTANCIA=DURACION/58.2;
 delay(1);
 Serial.println(DISTANCIA);
 if(DISTANCIA<=20 && DISTANCIA>=0){
   flag2=1;
   flag1=0;
   para();
   retrocede();
   delay(1000);
   para2();
   delay(500);
   observazona();
   girorandom=random(1,3);
   delay(1);
   if (girorandom==1){
      giraizqda();
      delay(1500);
    }
   else{
      giradcha();
      delay(1500);
    }
    para2();
  }

if(Serial.available()>0){
  String recibo = Serial.readStringUntil('\n');
  if (recibo=="finaliza")
  x=2;
  flag1=0;
  flag2=0;
  para2();
  finalizando();
  while(x!=1){
  VALORMICRO=digitalRead(MICRO);
  if(VALORMICRO==HIGH){
    Serial.println("inicio");
    Serial.println("grabacion");
    x=1;
  }
}
 for (int i=0;i<24;i++){
tira.setPixelColor(i,0,0,255);
tira.show();
delay(40);
tira.setPixelColor(i,0,0,0);
tira.show();
```

```
  }
despierta();

}


    if(flag1==0){                    //el flag1 para que continuamente no
este moviendo las cejas hacia delante (innecesario)
     if(flag2==1){

    for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50 veces
    digitalWrite (A1,HIGH);
    delayMicroseconds(PULSOMINCEJAD);
    digitalWrite (A1,LOW);           // llevamos a 0° cejad

    digitalWrite (5,HIGH);
    delayMicroseconds(PULSOMAXCEJAI);
    digitalWrite (5,LOW);          // llevamos a 180° cejai(delante)
    delay(10);
 }
    for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50 veces
     digitalWrite (6,HIGH);
     delayMicroseconds(PULSOMAXCTA);                // llevamos a 0°la cresta
(hacia detras)
     digitalWrite (6,LOW);
     delay(10);
 }
     }

   for (int i=23;i>=0;i--){
    tira.setPixelColor(i,0,255,0);
    tira.show();
    }

    for(int i=0;i<18;i++){
tone(ALTAVOZ,100*i,100);
delay(15+i);
}
noTone(ALTAVOZ);

avanza();
    flag1=1;
    delay(10);


}

}

void avanza()
{

  digitalWrite(MOTOR1_DIRECCION_PIN,HIGH); //direccion hacia delante con HIGH
  digitalWrite(MOTOR2_DIRECCION_PIN,HIGH);
  analogWrite(MOTOR1_VELOC_PIN,velocidad);
  analogWrite(MOTOR2_VELOC_PIN,velocidad);

}

void giraizqda()
{
  for (int i=0;i<24;i++){
```

```
    tira.setPixelColor(i,0,0,0);
    tira.show();
    }

  for (int i=4;i<16;i++){
    tira.setPixelColor(i,0,0,255);
    tira.show();
    }
  digitalWrite(MOTOR1_DIRECCION_PIN,HIGH);
  digitalWrite(MOTOR2_DIRECCION_PIN,LOW);
  analogWrite(MOTOR1_VELOC_PIN,velocidad2);
  analogWrite(MOTOR2_VELOC_PIN,velocidad2);
}

void giradcha()
{
  for (int i=0;i<24;i++){
    tira.setPixelColor(i,0,0,0);
    tira.show();
    }

 for (int i=16;i<24;i++){
  tira.setPixelColor(i,0,0,255);
  tira.show();
  }
  for (int i=0;i<4;i++){
    tira.setPixelColor(i,0,0,255);
    tira.show();
    }
  digitalWrite(MOTOR1_DIRECCION_PIN,LOW);
  digitalWrite(MOTOR2_DIRECCION_PIN,HIGH);
  analogWrite(MOTOR1_VELOC_PIN,velocidad2);
  analogWrite(MOTOR2_VELOC_PIN,velocidad2);
}

void retrocede()
{
      for (int i=23;i>=0;i--){
   tira.setPixelColor(i,0,0,255);
   tira.show();
   }
  digitalWrite(MOTOR1_DIRECCION_PIN,LOW); //direccion hacia detras
  digitalWrite(MOTOR2_DIRECCION_PIN,LOW);
  analogWrite(MOTOR1_VELOC_PIN,velocidad);
  analogWrite(MOTOR2_VELOC_PIN,velocidad);
}

void para()
{
  analogWrite(MOTOR1_VELOC_PIN,0);
   analogWrite(MOTOR2_VELOC_PIN,0);

    for (int i=23;i>=0;i--){
   tira.setPixelColor(i,255,0,0);
   tira.show();
   }
   tone(ALTAVOZ,220,600);
   delay(120);
   tone(ALTAVOZ,40,500);
      for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50
veces
    digitalWrite (A1,HIGH);
```

```
    delayMicroseconds(PULSOMAXCEJAD);
    digitalWrite (A1,LOW);  // llevamos a 180° cejad (atras)

    digitalWrite (5,HIGH);
    delayMicroseconds(PULSOMINCEJAI);
    digitalWrite (5,LOW);  // llevamos a 0° cejai
    delay(10);
   }


   for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50 veces
    digitalWrite (6,HIGH);
    delayMicroseconds(PULSOMINCTA);                // llevamos a 180° (hacia
arriba la cresta)
    digitalWrite (6,LOW);
    delay(10);
 }
}


void para2()
{

  analogWrite(MOTOR1_VELOC_PIN,0);
  analogWrite(MOTOR2_VELOC_PIN,0);
}

void observazona(){
      for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50
veces
  digitalWrite (A0,HIGH);
  delayMicroseconds(800); //rotacion cuerpo a 0°
  digitalWrite (A0,LOW);

  delay(10);
   }

         for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50
veces
  digitalWrite (A0,HIGH);
  delayMicroseconds(1500); //rotacion cuerpo a 90°
  digitalWrite (A0,LOW);

  delay(10);
   }

         for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50
veces
  digitalWrite (A0,HIGH);
  delayMicroseconds(2200); // rotacion cuerpo a 180°
  digitalWrite (A0,LOW);

  delay(10);
   }

         for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50
veces
  digitalWrite (A0,HIGH);
  delayMicroseconds(1500); //rotacion cuerpo a 90°
  digitalWrite (A0,LOW);

  delay(10);
```

```
   }
}

void despierta(){
   for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50 veces
digitalWrite (A5,HIGH);
delayMicroseconds(PULSOMAXCLL);                 // llevamos a 180° (cabeza
hacia arriba)
digitalWrite (A5,LOW);
delay(10);
 }

      for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50
veces
  digitalWrite (A0,HIGH);
  delayMicroseconds(800); //rotacion cuerpo a 0°
  digitalWrite (A0,LOW);

  delay(10);
   }

         for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50
veces
  digitalWrite (A0,HIGH);
  delayMicroseconds(1500); //rotacion cuerpo a 90°
  digitalWrite (A0,LOW);

  delay(10);
   }

         for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50
veces
  digitalWrite (A0,HIGH);
  delayMicroseconds(2200); // rotacion cuerpo a 180°
  digitalWrite (A0,LOW);

  delay(10);
   }

         for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50
veces
  digitalWrite (A0,HIGH);
  delayMicroseconds(1500); //rotacion cuerpo a 90°
  digitalWrite (A0,LOW);

  delay(10);
   }
 for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50 veces
digitalWrite (6,HIGH);
delayMicroseconds(PULSOMINCTA);                 // llevamos a 0° (hacia
delante la cresta)
digitalWrite (6,LOW);
delay(10);
 }


 for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50 veces
digitalWrite (6,HIGH);
delayMicroseconds(PULSOMAXCTA);                 // llevamos a 180° (hacia
detras la cresta)
digitalWrite (6,LOW);
delay(10);
```

```
 }
}

void pdc()
{
tone(ALTAVOZ,294,100);
  tira.setBrightness(20);
  tira.show();
delay(100);
tone(ALTAVOZ,440,100);
  tira.setBrightness(70);
  tira.show();
delay(100);
tone(ALTAVOZ,523,100);
  tira.setBrightness(20);
  tira.show();
delay(100);
tone(ALTAVOZ,587,100);
  tira.setBrightness(70);
  tira.show();
delay(200);
tone(ALTAVOZ,587,100);
  tira.setBrightness(20);
  tira.show();
delay(200);
tone(ALTAVOZ,587,100);
  tira.setBrightness(70);
  tira.show();
delay(100);
tone(ALTAVOZ,659,100);
  tira.setBrightness(20);
  tira.show();
delay(100);
tone(ALTAVOZ,698,100);
  tira.setBrightness(70);
  tira.show();
delay(200);
tone(ALTAVOZ,698,100);
  tira.setBrightness(20);
  tira.show();
delay(200);
tone(ALTAVOZ,698,100);
  tira.setBrightness(70);
  tira.show();
delay(100);
tone(ALTAVOZ,783,100);
  tira.setBrightness(20);
  tira.show();
delay(100);
tone(ALTAVOZ,659,100);
  tira.setBrightness(70);
  tira.show();
delay(200);
tone(ALTAVOZ,659,100);
  tira.setBrightness(20);
  tira.show();
delay(200);
tone(ALTAVOZ,587,100);
  tira.setBrightness(70);
  tira.show();
delay(100);
tone(ALTAVOZ,523,100);
```

```
  tira.setBrightness(20);
  tira.show();
delay(100);
tone(ALTAVOZ,523,100);
  tira.setBrightness(70);
  tira.show();
delay(100);
tone(ALTAVOZ,587,100);
  tira.setBrightness(20);
  tira.show();
delay(300);

}

void sw()
{
  delay(100);
  tone(ALTAVOZ,440);
  tira.setBrightness(20);
  tira.show();
  delay(250);
  noTone(ALTAVOZ);
  tone(ALTAVOZ,440);
  tira.setBrightness(70);
  tira.show();
  delay(250);
  noTone(ALTAVOZ);
  tone(ALTAVOZ,440);
  tira.setBrightness(20);
  tira.show();
  delay(250);
  tone(ALTAVOZ,349);
  tira.setBrightness(70);
  tira.show();
  delay(175);
  tone(ALTAVOZ,523);
  tira.setBrightness(20);
  tira.show();
  delay(75);
  tone(ALTAVOZ,440);
  tira.setBrightness(70);
  tira.show();
  delay(250);
  tone(ALTAVOZ,349);
  tira.setBrightness(20);
  tira.show();
  delay(175);
  tone(ALTAVOZ,523);
  tira.setBrightness(70);
  tira.show();
  delay(75);
  tone(ALTAVOZ,440);
  tira.setBrightness(20);
  tira.show();
  delay(400);
  noTone(ALTAVOZ);

}

void hp()
{
    tone(ALTAVOZ,617,150);
```

```
      tira.setBrightness(20);
  tira.show();
  delay(150);
  tone(ALTAVOZ,824,150);
  delay(200);
  tone(ALTAVOZ,980,150);
    tira.setBrightness(70);
  tira.show();
  delay(100);
  tone(ALTAVOZ,873,200);
    tira.setBrightness(20);
  tira.show();
  delay(150);
  tone(ALTAVOZ,824,300);
    tira.setBrightness(70);
  tira.show();
  delay(250);
  tone(ALTAVOZ,1234,200);
    tira.setBrightness(20);
  tira.show();
  delay(200);
  tone(ALTAVOZ,1100,350);
    tira.setBrightness(70);
  tira.show();
  delay(300);
  tone(ALTAVOZ,925,350);
    tira.setBrightness(20);
  tira.show();
  delay(300);
  tone(ALTAVOZ,824,250);
    tira.setBrightness(70);
  tira.show();
  delay(200);
  tone(ALTAVOZ,980,150);
    tira.setBrightness(20);
  tira.show();
  delay(100);
  tone(ALTAVOZ,873,200);
    tira.setBrightness(70);
  tira.show();
  delay(150);
  tone(ALTAVOZ,777,200);
    tira.setBrightness(20);
  tira.show();
  delay(150);
  tone(ALTAVOZ,873,200);
    tira.setBrightness(70);
  tira.show();
  delay(150);
  tone(ALTAVOZ,617,350);
    tira.setBrightness(20);
  tira.show();
  delay(300);
}

void finalizando()
{
    for (int i=0;i<24;i++){
    tira.setPixelColor(i,0,0,0);
    tira.show();
    }
```

```
   for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50 veces
  digitalWrite (A5,HIGH);
  delayMicroseconds(PULSOMINCLL);                // llevamos a 0° al cuello
(cabeza hacia abajo)
  digitalWrite (A5,LOW);
  delay(10);
 }

   for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50 veces
digitalWrite (6,HIGH);
delayMicroseconds(PULSOMAXCTA);                // llevamos a 180° (hacia
detras la cresta)
digitalWrite (6,LOW);
delay(10);
 }

    for (int Hz =0; Hz < 50 ;Hz++){      // repetimos la instruccion 50 veces
    digitalWrite (A1,HIGH);
    delayMicroseconds(PULSOMINCEJAD);
    digitalWrite (A1,LOW);          // llevamos a 0° cejad

    digitalWrite (5,HIGH);
    delayMicroseconds(PULSOMAXCEJAI);
    digitalWrite (5,LOW);          // llevamos a 180° cejai(delante)
    delay(10);
 }
}
```

6.3.2 Parte II: Raspberry Pi

```
import serial
import os
import datetime as dt
from picamera import PiCamera
import time
#dmesg | grep -v disconnect | grep -Eo "tty(ACM|USB)." | tail -1
destino='/home/pi/Videos/Prueba1'
destino2='/home/pi/Pictures/Obstaculos'
camara=PiCamera()
com=serial.Serial('/dev/ttyACM0',9600)


def grabo_video():
        nombrevideo=os.path.join(destino,dt.datetime.now().strftime('%Y-%m-
%d_%H:%M:%S.h264'))
        camara.start_preview()
        camara.start_recording(nombrevideo)


def paro_video():
        camara.stop_recording()
        camara.stop_preview()

while True:
    com.flushInput()
    cont=0;
    print("el valor de cont ahora es:"+str(cont))

    while True:
```

```
        lectura=com.readline()
        line=lectura.decode('utf-8').strip()
        print(line)

        if line=="grabacion":
            grabo_video()
            inicio=time.time()
            com.flushInput()
            break

    while True:
        lectura=com.readline()
        line=lectura.decode('utf-8').strip()
        print(line)
        fin=time.time()

        if int(line)<= 20:
            nombrefoto=os.path.join(destino2,dt.datetime.now().strftime('%Y-
%m-%d_%H:%M:%S.jpg'))
            camara.start_preview()
            time.sleep(1)
            camara.capture(nombrefoto)
            camara.stop_preview()
            cont=cont+1
            print("valor del contador:"+str(cont))

        if cont==5 or fin-inicio>=80:
            print("finalizando")
            mensaje="finaliza"
            envio=mensaje.encode('latin-1')
            com.write(envio)
            time.sleep(7)
            paro_video()
            break
```

7 BIBLIOGRAFIA Y REFERENCIAS

A continuación se exponen aquellos documentos y fuentes consultadas y que han servido de apoyo a este trabajo, además de lugares de interés relacionados con la materia tratada en este.

[1] Arduino Uno: https://www.hwlibre.com/arduino-uno/.

[2] Motor Shield L298P: https://www.mantech.co.za/Datasheets/Products/EX029.pdf.

[3] Sensor Shield V5: https://solectroshop.com/es/shields-arduino/19-sensor-shield-v50-apc220-modulo-bluetooth-5v-servo-para-arduino.html.

[4] Raspberry Pi 4B: https://www.raspipc.es/index.php?ver=tienda&accion=verArticulo&idProducto=1751&src=raspberrypi.

[5] Cámara Raspberry: https://es.rs-online.com/web/p/camaras-para-raspberry-pi/9132664/?cm_mmc=ES-PLA-DS3A-_-google-_-PLA_ES_ES_Raspberry_Pi_%26_Arduino_y_M%C3%B3dulos_de_Desarrollo_Whoop-_-(ES:Whoop!)+C%C3%A1maras+para+Raspberry+Pi-_-9132664&matchtype=&pla-335061582843&gclid=.

[6] Anillo de 24 LEDs: https://sandorobotics.com/producto/com-12665/.

[7] Acerca del chip LED del anillo de LEDs: https://akirasan.net/como-funciona-un-led-ws2812b/.

[8] Sensor ultrasónico: https://electronperdido.com/shop/sensores/distancia/sensor-distancia-ultrasonidos-hc-sr04/.

[9] Micrófono KY-038: https://uelectronics.com/producto/modulo-ky-038-sensor-microfono/.

[10] Mini altavoz mp3: https://solectroshop.com/es/sonido-y-acustica/5098-mini-altavoz-40mm-3w-impedancia-4-ohm-mp3.html?gclid=Cj0KCQjw7MGJBhD-ARIsAMZ0eesjLAr_zhHZJGbNR0N4t66UpffB0pOoMs4FCOuUFGHxAFy1ijlX0jUaAsowEALw_wcB.

[11] Servomotor Turnigy TGY50090: https://www.amazon.com/-/es/Turnigy-tgy-50090-metal-Analog-Servo/dp/B00USQXHZG.

[12] Servomotor Futaba 3003s:: https://store.prometec.net/producto/servo-futaba-s3003-180o/.

[13] Servomotor HXT12K M 11kg: https://hobbyking.com/es_es/hxt-10kg-servo-metal-gear-10kg-0-16sec-55g.html.

[14] Batería liPo: https://es.aliexpress.com/item/4000186574271.html?srcSns=sns_WhatsApp&spreadType=socialShare&bizType=ProductDetail&social_params=60091610549&aff_fcid=8136417042a0426f9ad90b452d324832-1630506126023-09744-_vtd7X8&tt=MG&aff_fsk=_vtd7X8&aff_platform=default&s.

[15] interruptor basculante: https://solectroshop.com/es/pulsadores-e-interruptores/971-2x-interruptor-basculante-3a-250v-on-off.html.

[16] A cerca de la conexión de la cámara de la Raspberry Pi: https://parzibyte.me/blog/2021/02/15/conectar-instalar-camara-raspberry-pi-4/.

[17] Página de descarga del Arduino IDE: https://www.arduino.cc/en/software.

[18] Página de informacion de manejo Arduino IDE: https://internetpasoapaso.com/arduino-ide/.

[19] Informacion a cerca del Thonny Python IDE: https://en.wikipedia.org/wiki/Thonny.

[20] Página oficial Thonny Python IDE: https://thonny.org/.

[21] Página oficial de Raspberry para sistema operativo Raspbian://www.raspberrypi.org/software/operating-systems/.

[22] Página de descarga de Putty, conexión SSH para Raspberry Pi: https://www.putty.org/.

[23] Página de descarga de realVNC, conexión VNC para Raspberry Pi: https://www.realvnc.com/es/connect/download/viewer/.

[24] Información a cerca de la conexión serial entre Arduino y Raspberry Pi: https://www.youtube.com/watch?v=Z-KoMbs7zsY