

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

PUESTA EN MARCHA Y PROGRAMACIÓN DE UN ROBOT ANIMATRÓNICO

Autor: Jaime Haldón Vallellano

Tutores: Ignacio Alvarado
Jose María Maestre Torreblanca

Titulación: Grado en Ingeniería de las
Tecnologías Industriales

Dpto. de Ingeniería de sistemas y automática



Sevilla, Septiembre de 2021

ÍNDICE

1. INTRODUCCIÓN
2. APARIENCIA FÍSICA
3. COMPONENTES ELECTRÓNICOS
4. SOFTWARE EMPLEADO
5. PROGRAMACIÓN DE CURRITO
6. CONCLUSIONES Y VALORACIONES

ÍNDICE

1. **INTRODUCCIÓN**
2. APARIENCIA FÍSICA
3. COMPONENTES ELECTRÓNICOS
4. SOFTWARE EMPLEADO
5. PROGRAMACIÓN DE CURRITO
6. CONCLUSIONES Y VALORACIONES

Antecedentes

- Se dispone del diseño del robot animatrónico.
- Se realiza como TFG la puesta en marcha del mismo.

Objetivos

- Analizar y programar cada uno de los componentes.
 - Coordinar de forma idónea cada uno de ellos.
-
- Lograr una correcta interacción de Currito con el entorno que le rodea.

ÍNDICE

1. INTRODUCCIÓN
2. **APARIENCIA FÍSICA**
 - Características
 - Exterior
 - Interior
3. COMPONENTES ELECTRÓNICOS
4. SOFTWARE EMPLEADO
5. PROGRAMACIÓN DE CURRITO
6. CONCLUSIONES Y VALORACIONES

Características

➤ Basado en la mascota Curro de la Expo 92

○ Material: ABS

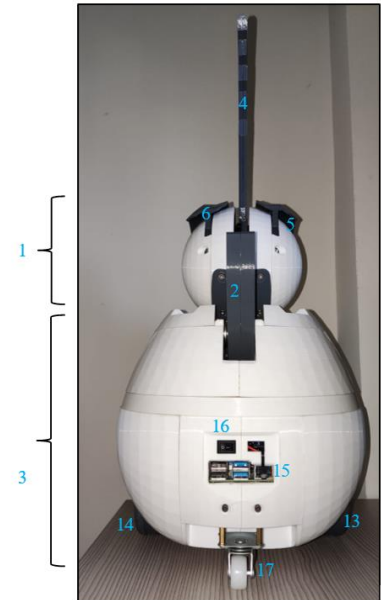
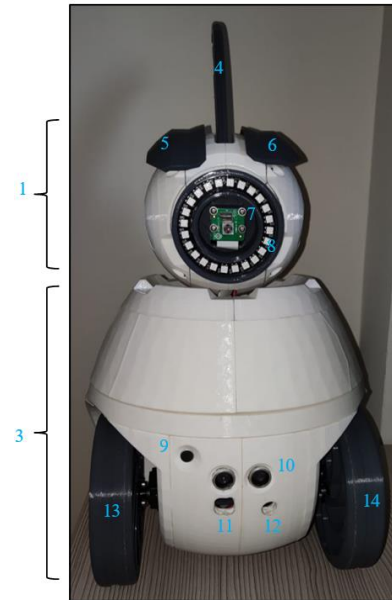
- Acrylonitrile Butadiene Styrene
- Impresión 3D
- Buena rigidez y ligereza
- Gran resistencia a impactos y temperatura

○ Dimensiones y peso

Alto	37.8 – 46.0 cm
Ancho	23.8 cm
Largo	24.3 cm
Peso	2.72 Kg

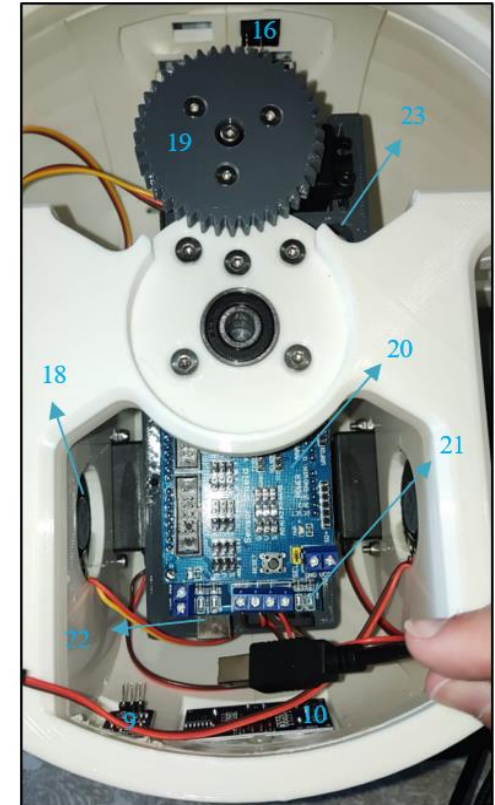
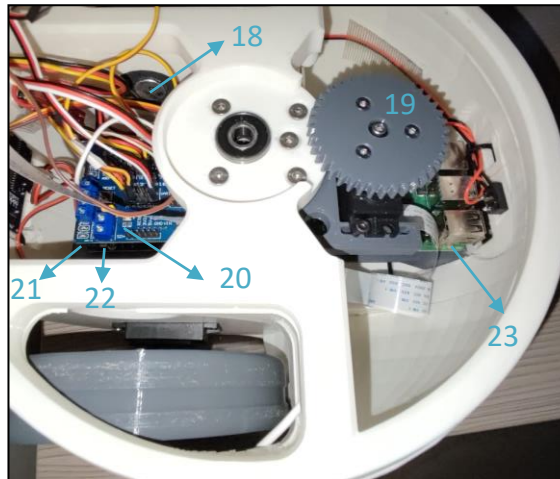
Exterior

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. Abertura 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



Interior

- 19. Engranaje para rotación del cuerpo
- 20. Sensor Shield V5
- 21. Motor Shield L298P
- 22. Arduino Uno
- 23. Raspberry Pi 4B



ÍNDICE

1. INTRODUCCIÓN

2. APARIENCIA FÍSICA

3. COMPONENTES ELECTRÓNICOS

- Arduino Uno
- Raspberry Pi 4B
- Motor Shield L298P
- Sensor Shield V5
- Cámara Raspberry Pi V2
- Anillo LED
- Sensor Ultrasónico HC-SR04
- Micrófono KY-038
- Altavoz mp3
- Servomotor Turnigy TGY50090
- Servomotor Futaba 3003
- Servomotor HXT12K M11kg
- Batería LiPo
- Interruptor basculante

4. SOFTWARE EMPLEADO

5. PROGRAMACIÓN DE CURRITO

6. CONCLUSIONES Y VALORACIONES

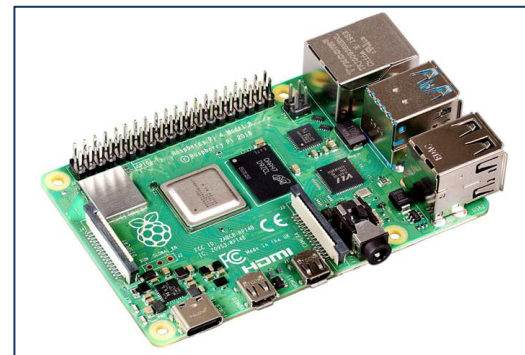
Arduino Uno

- Microcontrolador: Atmel ATmega328 a 16 Mhz
- Tensión de alimentación $\left\{ \begin{array}{l} - 6 \text{ a } 20\text{V} \text{ límites} \\ - 7 \text{ a } 12\text{V} \text{ recomendado} \end{array} \right.$
- Intensidad de corriente E/S: 40mA
- 14 pines digitales (6 pwm)
- 6 pines analógicos



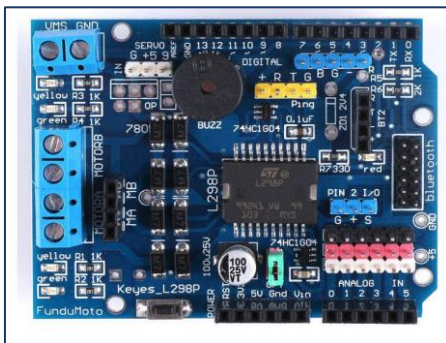
Raspberry Pi 4B

- Procesador: Broadcom BCM2711, quad-core Cortex-A72 a 1,5GHz
- Memoria RAM: 2GB LPDDR4
- Alimentación: 5V a 3A
- Avanzadas opciones de conectividad y multimedia



Motor shield L298P

- Placa distribuidora
- Chip controlador de motores: L298P
- Alimentación: 6.5 a 12V por VIN
- Intensidad máxima de salida a motores: 2A
- Se utilizará para mover las ruedas



Sensor shield V5

- Placa distribuidora
- Facilita la conexión de múltiples dispositivos
- Habilita GND y VCC independientes para cada pin



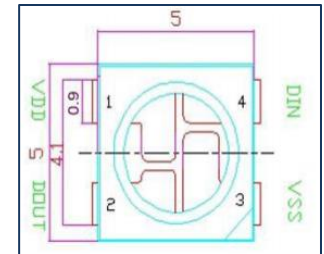
Cámara Raspberry Pi V2

- Sensor: Sony IMX 219 PQ CMOS de 8Mp
- Resolución de imagen estática: 8 Mp
- Velocidad cuadro de video: 1080p-30fps
- Múltiples modos y filtros
- Conectada a Raspberry Pi



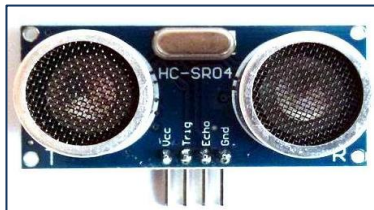
Anillo LED

- 24 neopixeles de tipo WS2812B
- Tensión de alimentación: 5V
- Intensidad máxima de consumo: 60mA
- 3 pines: VCC, GND y señal
- Conectado a pin digital 9

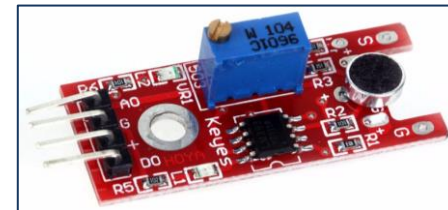


Sensor ultrasónico HC-SR04 Micrófono KY-038

- Rango de distancia: 2cm a 4m
- Tensión de alimentación: 5V
- Intensidad de consumo: 15mA
- 4 pines: VCC, GND, TRIG y ECHO
- Conectado a pines digitales 2 y 3



- Chip: LM393
- Tensión de alimentación: 5V
- Gama de frecuencia: 10 a 10000 Hz
- 4 pines: VCC, GND, A0 y D0
- Conectado a pin digital 7



Altavoz mp3

- Potencia: 2W
- 8 Ohmios
- Tensión de alimentación: 5V
- 2 pines: VCC y GND
- Conectado a pin digital 8



Servomotor Turnigy 50090

- 2 servos Turnigy TGY50090
- Torque/voltaje: 1,6 a 2 Kg/cm
- Velocidad/voltaje: 0.08 a 0.07 sec/60º
- Tensión de alimentación: 4.8 - 6 V
- Rotación de 180º
- Servo ceja derecha a pin analógico A0
- Servo ceja izquierda a pin digital 5



Servomotor Futaba 3003

- 2 servos Futaba 3003
 - Torque/voltaje: 3,17 a 4,10 Kg/cm
 - Velocidad/voltaje: 0.23 a 0.19 sec/60º
 - Tensión de alimentación: 4.8 - 6 V
 - Rotación de 180º
-
- Servo cresta a pin digital 6
 - Servo boca a pin digital 4



Servomotor HXT12K M 11Kg

- 4 servos HXT12K M 11Kg
 - Torque/voltaje: 9,4 a 11 Kg/cm
 - Velocidad/voltaje: 0.20 a 0.16 sec/60º
 - Tensión de alimentación: 4.8 - 6 V
 - Rotación de 180º o 360º
-
- Servo rotación a pin analógico A0
 - Servo cuello a pin analógico A5
 - Servo rueda derecha a pines digitales 10 y 12
 - Servo rueda izquierda a pines digitales 11 y 13



Batería LiPo

- Capacidad: 1500mAh
- Voltaje: 7,4V
- Peso: 68g

➤ Conectado a interruptor y a motor shield



Interruptor basculante

- Tensión máxima soportada: 250V
- Intensidad de corriente máxima: 3A

➤ Conectado a batería LiPo



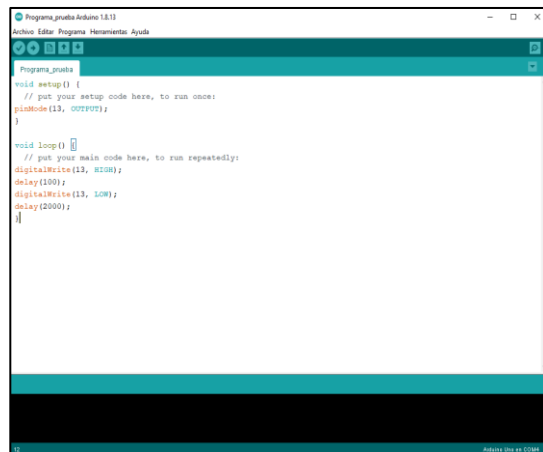
ÍNDICE

1. INTRODUCCIÓN
2. APARIENCIA FÍSICA
3. COMPONENTES ELECTRÓNICOS
4. **SOFTWARE EMPLEADO**
 - Arduino IDE
 - Thonny Python IDE
4. PROGRAMACIÓN DE CURRITO
5. CONCLUSIONES Y VALORACIONES

Arduino IDE



- Software libre de Arduino
- Programa de aplicación
- Lenguaje de programación similar a C++
- Múltiples librerías disponibles
- Estructura de 2 bloques obligatorios: “void setup” y “void loop”



```

Programa_guata Arduino 1.8.15
Archivo Editor Programa Herramientas Ayuda

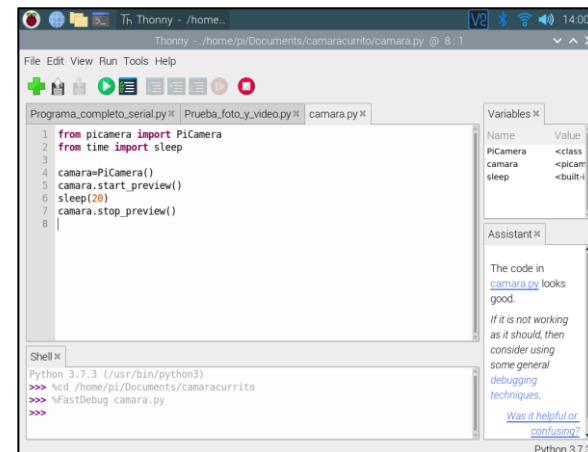
Programa_guata
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);
  delay(2000);
}
  
```

Thonny Python IDE



- Lenguaje Python
- Lenguaje dinámico
- Múltiples librerías disponibles
- Gran sencillez y fácil manejo
- Incluida en Raspbian desde 2017



```

Thonny - /home...
Thonny - /home/pi/Documents/camaracurrito/camara.py @ 8: 1

File Edit View Run Tools Help

Programa_completo_serial.py Prueba_foto_y_video.py camara.py
1 from picamera import PiCamera
2 from time import sleep
3
4 camara=PiCamera()
5 camara.start_preview()
6 sleep(20)
7 camara.stop_preview()
8

Variables
Name Value
PiCamera <class
camara <picam
sleep <built-

Assistant
The code in
camara.py looks
good.
If it is not working
as it should, then
consider using
some general
debugging
techniques.
Was it helpful or
confusing?
Python 3.7.3
  
```

ÍNDICE

1. INTRODUCCIÓN

2. APARIENCIA FÍSICA

3. COMPONENTES ELECTRÓNICOS

4. SOFTWARE EMPLEADO

5. PROGRAMACIÓN DE CURRITO

- Servomotores
- Anillo LED
- Sensor ultrasónico
- Micrófono
- Altavoz
- Ruedas
- Cámara
- Comunicación Arduino-Raspberry Pi
- Programa completo esquivo-obstáculos

6. CONCLUSIONES Y VALORACIONES

Servomotores

- **Incluir librería servo:** `#include <Servo.h>`
- **Definir variable tipo:** `Servo servol;`
- **Vincular servo a pin:** `servol.attach(PIN, PULSOMIN, PULSOMAX);`
- **Desvincular servo:** `servol.detach(PIN);`
- **Movimiento del servo:** `servol.write(GRADOS);`
- **Pulsos de los servos:**

<i>Servomotor</i>	<i>PULSOMIN</i>	<i>PULSOMAX</i>
Ceja derecha	1200	1800
Ceja izquierda	1000	1800
Cresta	700	1300
Cuello	500	2500
Rotación	2400	600

Anillo LED

- **Incluir librería :** `#include <Adafruit_NeoPixel.h>`
- **Definir variable tipo:** `Adafruit_NeoPixel tira=Adafruit_NeoPixel(24,9,NEO_GRB+NEO_KHZ800);`
- **Inicialización:** `tira.begin();`
- **Establecer color:**
`tira.setPixelColor(7,90,30,0);`
- **Modificar brillo:** `tira.setBrightness();`
- **Encender LEDs:** `tira.show();`

Sensor ultrasónico

- **Definición de pines:**
`pinMode(TRIGGER, OUTPUT);`
`pinMode(ECHO, INPUT);`
- **Envío de onda:** `digitalWrite(TRIGGER, HIGH);`
- **Recepción de onda:** `pulseIn(ECHO, HIGH);`
- **Cálculo de distancia:**
`digitalWrite(TRIGGER, HIGH);`
`delay(1);`
`digitalWrite(TRIGGER, LOW);`
`DURACION= pulseIn(ECHO, HIGH);`
`DISTANCIA=DURACION/58.8;`

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

De manera que:

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

Micrófono

- Ajuste previo de ganancia
- Definición de pin: `pinMode(MICRO, INPUT);`
- Lectura de ruido: `digitalRead(MICRO);`

Altavoz

- Definición de pin: `pinMode(ALTAVOZ, OUTPUT);`
- Producción de sonido:
`tone(ALTAVOZ, FRECUANCIA, DURACION);`

Ruedas

- **Definición de pines:**

```
pinMode(MOTOR1_DIRECCION_PIN, OUTPUT);
```

```
pinMode(MOTOR2_DIRECCION_PIN, OUTPUT);
```

```
pinMode(MOTOR1_VELOC_PIN, OUTPUT);
```

```
pinMode(MOTOR2_VELOC_PIN, OUTPUT);
```

- **Movimiento de ruedas:**

```
analogWrite(MOTOR1_VELOC_PIN, velocidad);
```

```
analogWrite(MOTOR2_VELOC_PIN, velocidad);
```

- **Dirección de ruedas:**

```
digitalWrite(MOTOR1_DIRECCION_PIN, HIGH/LOW);
```

```
digitalWrite(MOTOR2_DIRECCION_PIN, HIGH/LOW);
```

Cámara

- **Activar la interfaz previamente**

- **Inclusión de librerías:** `from picamera import PiCamera`

- **Vincular variable de tipo cámara:**
`camara=PiCamera()`

- **Previsualización:**
`camara.start_preview()`
`camara.stop_preview()`

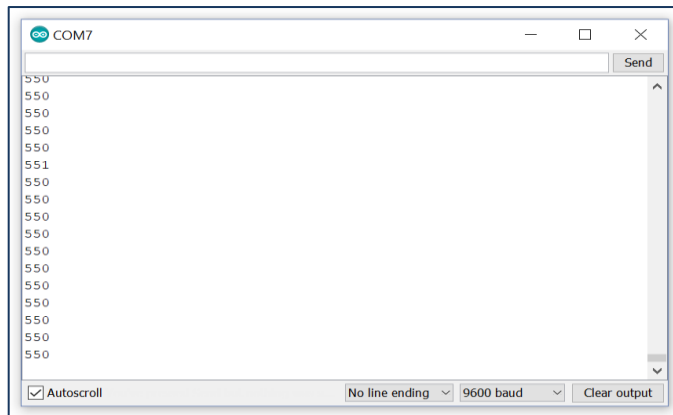
- **Captura de foto:**
`camara.capture(destinofoto)`

- **Captura de vídeo:**
`camara.start_recording(destinovideo)`
`camara.stop_recording(destinovideo)`
➤ `destinovideo='/home/pi/Videos/prueba1'`

Comunicación Arduino y Raspberry Pi

○ Arduino:

- **Monitor serial:** `Serial.begin(9600);`
- **Escribir:** `Serial.println();`
- **Leer:** `Serial.read();`
- **Otras:** `Serial.readStringUntil('\n')`
`Serial.flush();`
`Serial.available();`



○ Raspberry:

- **Librería:** `import serial`
- **Configurar puerto serial:**
`com=serial.Serial('/dev/ttyACM0',9600)`
- **Leer:** `com.readline()`
- **Escribir:** `com.write()`
- **Otra:** `com.flushInput()`

Programa completo esquiv-obstáculos

○ **Arduino:**

- ☐ Posición de espera
- ☐ Un valor alto en el micrófono lo inicia
- ☐ Se comunica con Raspberry para su inicio
- ☐ Reproduce aleatoriamente 1 de 3 melodías y enciende anillo LED
- ☐ Observa la zona
- ☐ Calcula distancia de objeto más próximo hasta aviso de Raspberry:
 - <20cm: cambia de dirección
 - >20cm: avanza
- ☐ Repite el proceso

❖ **No se usará librería “Servo.h”:**

- Desactiva funcionalidad PWM

○ **Raspberry:**

- ☐ Espera inicio de Curríto
- ☐ Comienza grabación y temporizador
- ☐ Toma fotos de obstáculos
- ☐ Si toma 5 fotos o ha pasado 1,5 minutos: ordena a Arduino que pare.
- ☐ Repite el proceso

```
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);
}
```

ÍNDICE

1. INTRODUCCIÓN
2. APARIENCIA FÍSICA
3. COMPONENTES ELECTRÓNICOS
4. SOFTWARE EMPLEADO
5. PROGRAMACIÓN DE CURRITO
6. **CONCLUSIONES Y VALORACIONES**

Conclusiones y valoraciones

- **Exitosa interacción de Curríto con el entorno**
 - **Correcto funcionamiento de los componentes utilizados**
 - **Conexión de Arduino y Raspberry Pi por puerto serie**

- **Enfoque práctico de la formación académica adquirida**
 - **Análisis de información técnica de componentes**
 - **Manejo de software para programación**
 - **Profundo aprendizaje de los entornos Arduino y Raspbian**

- **Múltiples ampliaciones de futuro**

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

PUESTA EN MARCHA Y PROGRAMACIÓN DE UN ROBOT ANIMATRÓNICO

Autor: Jaime Haldón Vallellano

Tutores: Ignacio Alvarado
Jose María Maestre Torreblanca

Titulación: Grado en Ingeniería de las
Tecnologías Industriales

Dpto. de Ingeniería de sistemas y automática



Sevilla, Septiembre de 2021