Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

SISTEMA ALTERNATIVO DE MOVILIDAD PARA PERSONAS CON DISCAPACIDAD DE MOVIMIENTO POR MEDIO DE UNA SILLA DE RUEDAS CONTROLADA POR EL MOVIMIENTO DEL OJO

Andrés Estuardo Tejeda Girón
Asesorado por el Ing. Jorge Mario Hernández Rivas

Guatemala, noviembre de 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

SISTEMA ALTERNATIVO DE MOVILIDAD PARA PERSONAS CON DISCAPACIDAD DE MOVIMIENTO POR MEDIO DE UNA SILLA DE RUEDAS CONTROLADA POR EL MOVIMIENTO DEL OJO

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

ANDRÉS ESTUARDO TEJEDA GIRÓN
ASESORADO POR EL ING. JORGE MARIO HERNÁNDEZ RIVAS

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO ELECTRÓNICO

GUATEMALA, NOVIEMBRE DE 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Oscar Humberto Galicia Nuñez
VOCAL V	Br. Carlos Enrique Gómez Donis
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. José Anibal Silva de los Ángeles
EXAMINADOR	Ing. Carlos Eduardo Guzmán Salazar
EXAMINADOR	Ing. Byron Odilio Arrivillaga Méndez
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

SISTEMA ALTERNATIVO DE MOVILIDAD PARA PERSONAS CON DISCAPACIDAD DE MOVIMIENTO POR MEDIO DE UNA SILLA DE RUEDAS CONTROLADA POR EL MOVIMIENTO DEL OJO

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 14 de marzo de 2018.

Andrés Estuardo Tejeda Girón

ACTO QUE DEDICO A:

Dios	Por darme esta oportunidad y llenarme de bendiciones durante mis estudios.
Mis padres	Mynor Tejeda e Ingrid Girón, por su amor y apoyo.
Mis abuelas	Por creer en mí.
Mis hermanas	Kathya y Melissa Tejeda Girón, por estar siempre para mí.
Mi familia	Por su apoyo durante este trayecto.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser la casa de estudios que me dio la oportunidad de aprender, crecer y desenvolverme.
Facultad de Ingeniería	Por el conocimiento adquirido a lo largo de la carrera y el apoyo brindado durante este tiempo.
Mis amigos	Por su motivación y apoyo incondicional.
IEEE	Por las oportunidades que he tenido al pertenecer a la institución.
Mi asesor	Ing. Jorge Mario Hernández, por brindarme su apoyo durante el desarrollo de este trabajo de graduación.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

LISTA DE SÍMBOLOS

Símbolo	Significado
A	Amperios
cm	Centímetro
p	Escaneo progresivo
GB	Gigabyte
°C	Grados Celsius
Km/h	Kilómetros por hora
MB	Megabytes
MP	Megapíxeles
m	Metro
s	Segundos
TB	Terabytes
V	Voltios
W	Watts

GLOSARIO

ADC	Conversión análoga a digital, por sus siglas en inglés.
Ánodo	Electrodo positivo de una celda electrolítica.
ARM	Tipo de arquitectura utilizada en la mayoría de procesadores de sistemas embebidos.
Arreglos	Un conjunto de datos ordenados en forma de vectores o matrices.
Bit	Dígito binario. Se utiliza para medir la capacidad de almacenamiento o cantidad de información digital.
Booleano	Variable que solo puede tomar dos valores.
Cátodo	Electrodo negativo de una celda electrolítica.
CPU	Unidad central de procesamiento, por sus siglas en inglés.
DC	Corriente continua.
EEPROM	Tipo de memoria ROM que puede ser programada y borrada eléctricamente.

Electrólisis	Proceso químico en el cual se descomponen sustancias al aplicar corriente continua.
Electrolito	Sustancia que se descompone en la electrólisis.
Gigabit Ethernet	Protocolo de transmisión de datos basada en el protocolo Ethernet.
Hardware	Partes físicas que componen un dispositivo o sistema electrónico.
HDMI	Interfaz multimedia de alta definición, por sus siglas en inglés, es una interfaz de transmisión de audio y video.
Iris	La parte circular coloreada del ojo.
Joystick	Es una palanca de control. Es un periférico de entrada.
LAN	Red de área local, por sus siglas en inglés.
Machine learning	Campo en la computación que permite a un sistema computarizado mejorar progresivamente
Par motor	Es el momento de fuerza que ejerce un motor sobre un eje.

Píxel	Elemento más pequeño que forma parte de una imagen digital.
PWM	Modulación por anche de pulso, por sus siglas en inglés.
RAM	Memoria de acceso aleatorio, por sus siglas en inglés. Es la memoria principal de las computadoras.
Realidad aumentada	Término que se utiliza para describir el uso de dispositivos tecnológicos para añadir información visual al entorno del mundo real
SD	Formato de tarjeta de memoria utilizada en dispositivos portátiles.
Texas Instruments	Compañía que se dedica a la producción y distribución de componentes electrónicos.
Tupla	Lista ordenada de elementos
.	
USB	Bus serial universal, por sus siglas en inglés. Es un protocolo de conexión que permite enlazar físicamente distintos dispositivos.

RESUMEN

Para la realización de este proyecto es importante conocer las herramientas y materiales que hacen al sistema funcional y autónomo. También es necesario identificar las limitantes, tanto del usuario como del hardware del sistema.

Una vez hecho esto, se procede al desarrollo del sistema. Una cámara obtiene las imágenes de los movimientos oculares para luego ser analizadas en una Raspberry Pi. Por medio Python, se realiza el procesamiento de las imágenes para rastrear el ojo y obtener la posición del iris y, así enviar las instrucciones adecuadas a un microcontrolador para el control de los motores que impulsan la silla de ruedas.

Con las pruebas de la identificación del iris exitosas, se procede a realizar la instalación del sistema a una silla de ruedas basada en la información de este trabajo.

OBJETIVOS

General

Desarrollar un sistema de movimiento auxiliar para personas que se movilizan en silla de ruedas.

Específicos

1. Explicar los conceptos básicos necesarios para el desarrollo del proyecto.

2. Detallar los bloques que componen el sistema de movimiento auxiliar.

3. Desarrollar un programa capaz de rastrear el movimiento del ojo.

4. Promover el uso de la tecnología para solucionar problemas en el ámbito médico en Guatemala.

INTRODUCCIÓN

Con el constante crecimiento de la población, también incrementa la cantidad de personas que padecen de alguna discapacidad física que limita su locomoción y se ven en la necesidad de utilizar una silla de ruedas para movilizarse.

Existen diferentes tipos de sillas de ruedas, desde las que se conocen comúnmente hasta sillas eléctricas controladas por *joystick*, por movimientos o gestos faciales e incluso por comandos de voz, pero estas soluciones no son prácticas para todas las personas. Otra alternativa, es el control de la silla a través del rastreo de la posición del iris o la pupila del ojo humano.

Este proyecto está enfocado en la última alternativa para auxiliar a personas con necesidad de trasladarse en una silla de ruedas, pero no cuentan con ayuda de alguien para movilizarse.

1. CONCEPTOS FUNDAMENTLES

1.1. Ojo humano

Los ojos son los órganos encargados de reunir información en forma de luz de un campo visual para que el cerebro la interprete en forma de imágenes, la pupila captura la luz proveniente del exterior y su tamaño es controlado por el iris. Estos movimientos regulan la intensidad de la luz que entra por la pupila.

1.1.1. Campo visual del ojo humano

El campo visual de cada ojo es la parte del mundo exterior que percibe cada ojo. En teoría, debe ser circular, pero realmente se divide al medio por la nariz y, en su parte superior, por el techo de la órbita[1].

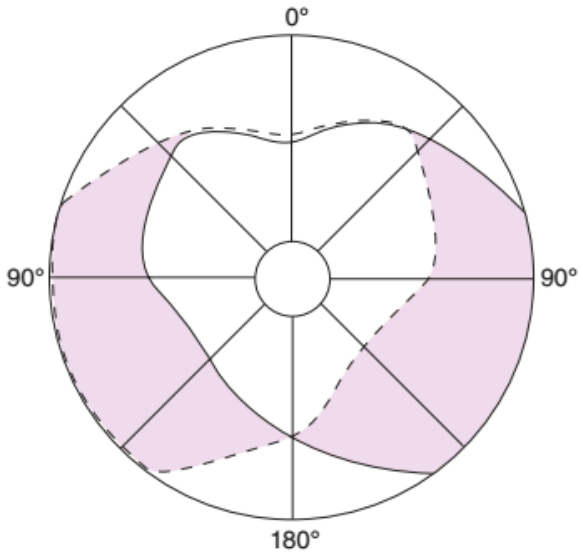
Figura 1. **Campo visual de los ojos**

Fuente: Campo visual de los ojos http://www.tuoptometrista.com/deteccion/alteraciones-del-campo-visual/ Consulta: 13 de abril de 2018.

[1] BARRETT, Kim y otros. *Ganong. Fisiología Médica*. p. 195.

En la figura 2 está representado el campo visual monocular y binocular. La línea punteada encierra el campo visual del ojo izquierdo; la línea sólida, la del ojo derecho. El área común, con forma de corazón, se observa con visión binocular, es decir, la zona donde ambos campos visuales coinciden, y las áreas de color representan la visión monocular[2].

Figura 2. **Campos visuales monocular y binocular**



Fuente: GANONG Fisiología Médica. p 195.

1.1.2. Movimientos oculares

Existen cuatro tipos de movimientos oculares: los movimientos sacádicos, que se producen cuando la mirada cambia de un objeto a otro. Los movimientos de persecución visual, que son movimientos de rastreo; se producen cuando los ojos siguen un objeto que se mueve. Los movimientos

[2] BARRETT, Kim y otros. *Ganong. Fisiología Médica*. p. 195.

vestibulares, que ocurren como respuesta cuando se conserva la vista fija mientras la cabeza se mueve. Y los movimientos de convergencia, que acercan los ejes visuales al centrar la atención en un objeto cercano[3].

1.2. Procesamiento digital de imágenes

El procesamiento digital de imágenes tiene como objetivo analizar aspectos de interés en objetos o individuos, ya sean estáticos o móviles, como un rostro o una placa automovilística, para conseguir cierto tipo de información sobre ellos y así trabajar con estos datos. Las imágenes pueden ser obtenidas de muchas maneras, por ejemplo: fotográficamente o electrónicamente. Para examinar, categorizar y cuantificar el contenido de estas imágenes se necesita de algo llamado visión artificial.

1.2.1. Visión artificial

La visión artificial permite la obtención, procesamiento y análisis de imágenes digitales, ya sea en tiempo real o no. Una cámara captura la información requerida para luego ser procesada y analizada, y ejecutar una acción según los resultados obtenidos.

Entre las tareas más básicas de la visión artificial se encuentran: determinación de posición y coordenadas de objetos, identificación e inspección, realización de mediciones, entre otras. Al aplicar estas tareas es posible: realizar inspecciones de objetos sin contacto físico, llevar control de calidad en líneas de producción sin error humano, reducir el tiempo en procesos automatizados, rastrear objetos en movimiento, identificar placas de automóviles, entre otros.

[3] BARRETT, Kim y otros. *Ganong. Fisiología Médica*. p. 196.

1.3. Raspberry Pi

Es una computadora del tamaño de una tarjeta de crédito diseñada para que las personas desarrollen habilidades de programación y diseño de hardware. Funciona ejecutando un sistema operativo desde una memoria SD y alimentada por un cargador de teléfono celular USB. Es posible conectarle un monitor, mouse, teclado. Gracias a su tamaño, programabilidad, conectividad y bajo costo, fácilmente se pueden desarrollar aplicaciones científicas, médicas, musicales, meteorológicas, de automatización, ocio, entre otras.

1.4. Sistema operativo

Un sistema operativo se encarga de gestionar todos los recursos de un dispositivo electrónico. Generalmente, cuenta con interfaz gráfica para que el usuario pueda utilizar el conjunto básico de programas incluidos en los diferentes tipos de sistemas existentes. Entre los sistemas operativos más conocidos se encuentran: Microsoft Windows, Linux, Mac OS, Android, IOS, Windows Phone, entre otros.

1.4.1. Sistemas operativos para Raspberry Pi

Existen diferentes versiones de sistemas operativos, tanto oficiales como no oficiales, para Raspberry Pi. Entre ellos se encuentran: Raspbian, Pidora, OSMC, LibreELEC, Snappy Ubuntu Core, Ubuntu MATE, RISC, etc. El sistema operativo recomendado para este proyecto es Raspbian.

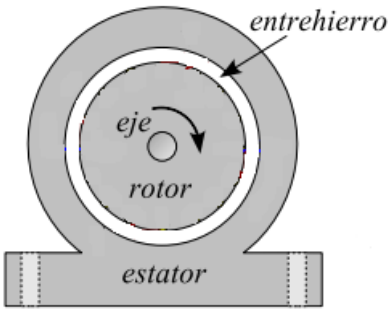
1.5. Microcontroladores

Son circuitos integrados programables que pueden ejecutar órdenes grabadas en su memoria. Están compuestos por varios bloques funcionales (CPU, memorias, entradas y salidas de propósito general y periféricos), que cumplen con una tarea específica[4]. El microcontrolador recomendado para este proyecto, por ser de bajo costo y tener excelentes capacidades de procesamiento, es el TM4C123GH6PM de la compañía Texas Instruments.

1.6. Máquina eléctrica rotativa

Es una máquina a la que se le suministra energía eléctrica y la transforma en energía mecánica. Esta acción genera un par motor que lo hace girar. Usualmente son de forma cilíndrica. Poseen un eje mecánico, a través del cual se realiza el intercambio de energía, una pieza estática llamada estator y una pieza móvil denominada rotor.

Figura 3. **Estructura básica de una máquina eléctrica rotativa**



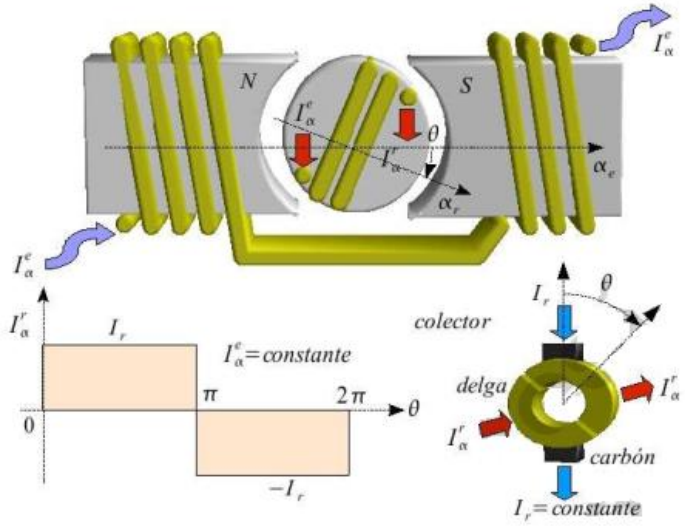
Fuente: ALLER, José Manuel. *Máquinas eléctricas rotativas: Introducción a la teoría general*. p. 104.

[4] BARRIENTOS, David. *Introducción al diseño de sistemas embebidos*. p. 23.

1.6.1. Máquina de corriente continua

Es una máquina eléctrica rotativa que posee un devanado en el estator, una armadura en el rotor y un colector que permite la inversión de corrientes en la armadura. Cuando se inyecta corriente continua al embobinado del estator y al rotor, estas corrientes producen fuerzas electromagnéticas que intentan alinearse, como consecuencia se produce el giro del rotor.

Figura 4. **Diagrama esquemático de una máquina elemental de corriente continua**

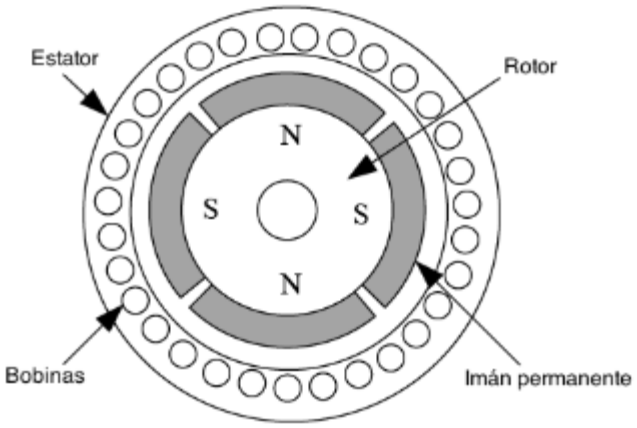


Fuente: ALLER, José Manuel. *Máquinas eléctricas rotativas: Introducción a la teoría general*. p. 117.

1.6.1.1. Motor eléctrico de corriente continua sin escobillas

Es una máquina eléctrica que tiene de 2 a 8 imanes permanentes en el rotor y bobinas en el estator. Posee un circuito electrónico que controla en secuencia la energización de las bobinas del estator. Las bobinas energizadas generan un campo magnético giratorio que atrae y repele los polos del imán del rotor, haciéndolo girar. A diferencia de los motores con escobillas, estos no generan desgaste, calor, ruido y chispas. Debido a estas características, los motores sin escobillas son más eficientes y permiten que el tamaño de los mismos sea más pequeño sin afectar su potencia de salida.

Figura 5. **Estructura básica de un motor de corriente continua sin escobillas**



Fuente: Estructura básica de un motor de corriente continua sin escobillas http://www.labc.usb.ve/paginas/EC5136/MaquinasDC.pdf. Consulta: 13 de abril de 2018.

1.7. Batería de ácido plomo

Este tipo de batería está conformada por placas negativas, positivas, aislante y un electrolito. Las placas negativas están hechas de plomo y realizan la función de cátodos, mientras que las positivas están hechas de plomo recubierto de dióxido de plomo y se comportan como ánodos. El electrolito es una disolución de ácido sulfúrico. Las placas se sumergen en un contenedor con el electrolito y se separan los ánodos de los cátodos con un aislante para evitar un cortocircuito.

Cuando el plomo se combina con el ácido sulfúrico, se produce sulfato de plomo. Al aplicar una corriente eléctrica a las placas ocurre la electrólisis del agua, de modo que se libera hidrógeno en la placa negativa y oxígeno en la placa positiva. Al mismo tiempo, se genera dióxido de plomo en la placa positiva (haciéndola una carga positiva), y en la placa negativa se tiene una pérdida de electrones (haciéndola una carga negativa)[5].

1.8. Transformador de voltaje DC-DC

Es un sistema que transforma una tensión de voltaje a otra. Existen tres tipos: elevadores, reductores y elevadores-reductores.

[5] *Estudio y fabricación de una batería ácido plomo. Temas de Ciencia y Tecnología* [en línea]. 2017, enero-abril. p. 23-28. Disponible en: http://www.utm.mx/edi_anteriores/temas61/T61_1E3_Estudio_y_fabricacion_ bateria.pdf. Consulta: 13 abril 2018

1.9. MOSFET

El transistor de efecto de campo semiconductor de óxido metálico, por sus siglas en inglés, es un semiconductor que se utiliza en tareas que se requiera conmutar o amplificar señales. Tienen 3 terminales: compuerta, fuente y drenaje, y están divididos en 2 tipos: canal N y canal P.

Este tipo de transistor trabaja similarmente que el transistor de efecto de campo (JFET), con la diferencia que los MOSFET tienen la compuerta aislada al canal e internamente tienen un diodo para evitar las corrientes de rebote.

Cuando la tensión en compuerta es 0, no existe corriente entre la fuente y el drenaje, lo que quiere decir que se encuentra en corte. Cuando una tensión lo suficiente mente grande alcanza el umbral de la compuerta del transistor, este entra en estado de saturación y se crea un flujo de corriente entre la fuente y el drenaje.

2. RECURSOS Y DIAGRAMA DE BLOQUES

2.1. Python

Es un lenguaje de programación orientado a objetos de alto nivel y de código abierto con sintaxis clara y concisa, esto hace que Python sea un lenguaje muy productivo y fácil de aprender. Soporta diferentes tipos de librerías y paquetes libres que permiten el desarrollo de diferentes tipos de aplicaciones. Es de rápida depuración, edición y corrección debido a que no cuenta con un proceso de compilación. La versión de Python recomendada es la 3, siendo la 3.7 la última actualización hasta la fecha.

2.1.1. Numpy

Es una librería libre para Python que maneja operaciones con arreglos multidimensionales. Permite representar imágenes en matrices para analizarlas con mayor facilidad y rapidez.

2.1.2. PySerial

Esta librería permite el acceso y configuración de puertos seriales a través de Python.

2.1.3. OpenCV

Es una librería de visión artificial y *machine learning* de código abierto compatible con C++, Pyhton, Java y MATLAB, con soporte para Windows,

Linux, Mac OS, iOS y Android. Diseñada para la eficiencia computacional y con un fuerte enfoque para aplicaciones basadas en tiempo real, provee una infraestructura común para aplicaciones de visión artificial.

La librería posee algoritmos optimizados que son usados para seguir el movimiento de los ojos, identificar objetos, rastrear movimientos de cámaras, extraer modelos 3D de objetos, detectar y reconocer rostros, clasificar acciones humanas en videos, encontrar imágenes similares en una base de datos de imágenes, rastrear objetos en movimiento, compilar imágenes para generar una imagen de un escenario completo en alta resolución, remover los ojos rojos de imágenes tomadas con "*flash*", seguir los movimientos de los ojos, reconocer espacios y establecer marcadores en aplicaciones de realidad aumentada, entre otros.

2.2. Raspberry Pi 3 modelo B+

Este modelo es el más reciente de la línea de Raspberry. Tiene un procesador ARM Cortex-A53 de 64 bits, 1GB de memoria RAM, red LAN inalámbrica, Bluetooth 4,2, 40 pines de uso de propósito general, 4 puertos USB 2,0, 1 puerto Gigabit Ethernet, 1 puerto HDMI, espacio para memoria micro SD, y se alimenta con 5V y 2,5A vía un puerto micro USB.

2.3. Memoria micro SD

Es la memoria *flash* más pequeña del formato SD usada comúnmente en dispositivos electrónicos portátiles. Cuenta con diferentes velocidades de transferencia de datos y capacidad de almacenamiento de hasta 2TB. Se recomienda usar una memoria de al menos 8GB y de clase 10, es decir, que transfiere datos a una velocidad mínima de 10MB/s.

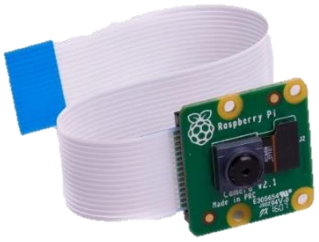
Figura 6. **Raspberry Pi 3 modelo B+**

Fuente: Raspberry Pi 3 modelo B+ https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. Consulta: 10 de marzo de 2018.

2.4. Módulo de cámara versión 2

Utiliza un sensor Sony IMX219 de 8 megapíxeles para tomar videos e imágenes en alta definición. Posee una buena calidad de imagen, fidelidad a los colores y un buen rendimiento en poca iluminación. Soporta formatos de video de 1080p30, 720p60 y VGA90. El módulo es compatible con todos los modelos de Raspberry Pi 1, 2 y 3, y se conecta con un cable cinta de 15cm.

Figura 7. **Módulo de cámara**



Fuente: Módulo de cámara https://www.raspberrypi.org/products/camera-module-v2/. Consulta: 10 de marzo de 2018.

2.5. Raspbian

Raspbian es el sistema operativo basado en Debian Linux, oficial con soporte para la Raspberry Pi. Consta con un set de programas básicos y utilidades que hacen funcionar una Raspberry Pi y viene con una gran variedad de paquetes en un software preconfigurado de fácil de instalación. El tamaño de la última versión, Raspbian Jessie, es de 4GB.

2.6. TM4C123G *LaunchPad Evaluation Kit* (Tiva C)

Una tarjeta de desarrollo capaz de depurar proyectos sin necesidad de herramientas adicionales. Ideal para proyectos de conexión, comunicación o control. Tiene una arquitectura de 32 bitsARM Cortex M4F. Trabaja a una capacidad máxima de 80 MHz, tiene una memoria *flash* de 256KB, una memoria RAM de 32 KB, una memoria EEPROM de 2KB, ADC de 12 bits, canales PWM, temporizadores, comparadores digitales y análogos y 43 pines de uso de propósito general.

Figura 8. **EK-TM4C123GXL**

Fuente: EK-TM4C123GXL http://www.ti.com/tool/ek-tm4c123gxl. Consulta: 10 de marzo de 2018.

2.7. *Code Composer Studio*

Es un entorno integrado de desarrollo -IDE-, por sus siglas en inglés- de Texas Instruments que provee soporte para los microcontroladores y sistemas embebidos de esta compañía. Incluye un compilador de C/C++ optimizado, editor y depurador de código y otras características que lo hacen intuitivo para el usuario.

2.8. Sensor de distancia ultrasónico HC-SR04

Un sensor capaz de calcular distancias que se encuentran en un rango de 2cm a 450cm. Conformado por dos pequeños cilindros, uno emite una señal ultrasónica y otro la recibe. Este sensor cuenta el tiempo que transcurre entre la emisión y recepción de la señal y produce un voltaje de salida proporcional a la medición. Útil para identificar obstáculos en una trayectoria.

Figura 9. **Sensor HC-SR04**

Fuente: Sensor HC-SR04 https://http2.mlstatic.com/sensor-ultrasonico-hc-sr04-medir-distancias-detectar-objetos-D_NQ_NP_787611-MLA20604543581_022016-F.jpg. Consulta: 10 de marzo de 2018.

2.9. Cámara web *Logitech* c270

Es una cámara digital que se conecta a una computadora a través de un puerto USB. Captura imágenes con resolución de hasta 3MP, captura de video en alta definición con 720p de resolución, autoenfoque y corrección automática de luz. Posee un cable de 1,5m de longitud.

Figura 10. ***Logitech* c270**

Fuente: *Logitech* c270 https://assets.logitech.com/assets/55372/webcam-c270-gallery.png. Consulta: 13 de abril de 2018.

2.10. Módulo LM2577

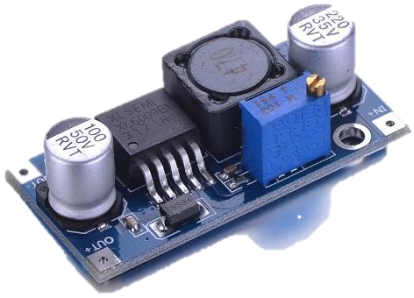
Es un transformador de voltaje DC-DC variable que puede elevar y reducir el voltaje que se le suministra. El voltaje de entrada puede variar entre 3,5V y 40V y su salida puede ser ajustada entre 1,5V y 35V y 2A.

2.11. MOSFET IRFZ44M

Este MOSFET de potencia tipo N es de ultra baja resistencia y de alta conmutación, lo cual lo hace viable y eficiente para una amplia variedad de aplicaciones. El empaquetado de este componente es el preferido ya que puede

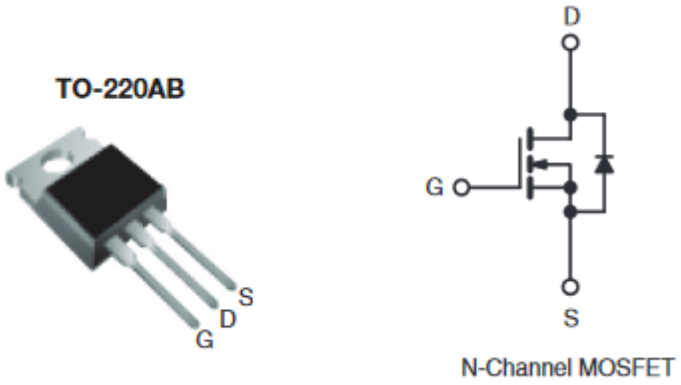
disipar muy bien potencia de hasta 50W. Puede manejar corrientes de drenaje de un máximo de 49A y soportar hasta temperaturas de 175 °C. Este transistor entra en estado de saturación con una tensión entre 2V y 4V, ideal para trabajar con el voltaje de salida de los pines del microcontrolador.

Figura 11. **Módulo LM2577**



Fuente: Módulo LM2577 http://kimtronix.com/admin/component_images/528821.jpg. Consulta: 13 de abril de 2018.

Figura 12. **IRFZ44M**

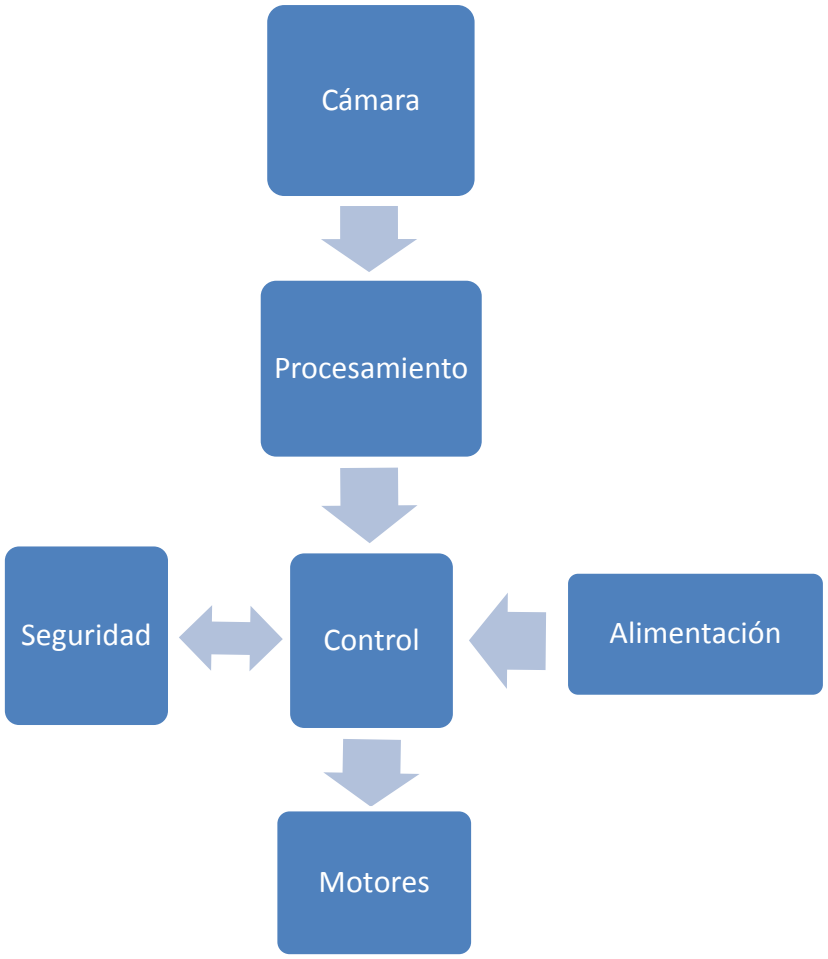


Fuente: IRFZ44M https://www.vishay.com/docs/91291/91291.pdf. Consulta: 13 de abril de 2018.

2.12. Diagrama de bloques

En la figura 13 se podrá visualizar el funcionamiento como diagrama de bloques.

Figura 13. **Diagrama de bloques del sistema**



Fuente: elaboración propia.

2.13. Bloque de cámara

Este bloque consta únicamente de una cámara que toma imágenes del ojo del usuario. Es acoplada a unos anteojos sin espejuelos, de manera que no sea invasivo para el usuario. La cámara se conecta a la Raspberry (p. 10). Debido a que el módulo de cámara de la Raspberry (p. 11) no cuenta con corrección automática de luz y su cable no es largo, es preferible usar una cámara que sí tenga estas características, como la *Logitech* c270 (p. 14).

2.14. Bloque de procesamiento

Las imágenes obtenidas son procesadas en este bloque. Un programa, hecho en Python (p. 19), identifica el área donde se encuentra el iris y rastrea los movimientos que hace.

2.15. Bloque de control

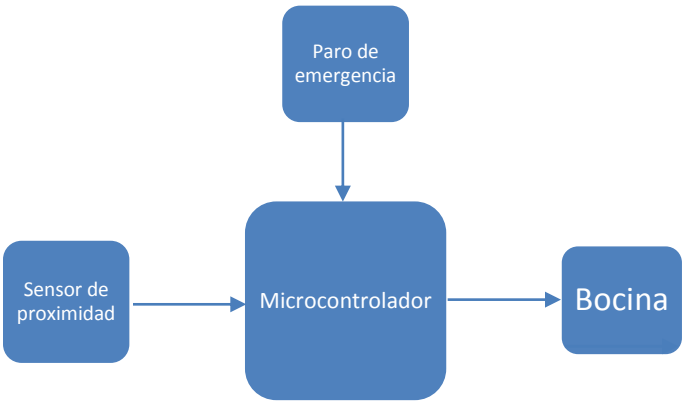
La tarjeta de desarrollo Tiva C (p. 12) es la que recibe los comandos de la Raspberry a través de conexión serial por medio de cable USB y ejecuta instrucciones dependiendo de los comandos identificados. Envía señales para encender y apagar los motores, regula la velocidad de estos y detecta las señales provenientes del bloque de seguridad en caso que haya un obstáculo.

2.16. Bloque de seguridad

Este bloque contiene las herramientas necesarias para garantizar la seguridad del usuario. Un botón de paro de emergencia que hará detener los motores cuando el usuario lo presione, un sensor de proximidad para detectar

obstáculos en el camino y una pequeña bocina que funciona como alarma en caso de encontrar un obstáculo.

Figura 14. **Diagrama del bloque de seguridad**



Fuente: elaboración propia.

2.17. Bloque de motores

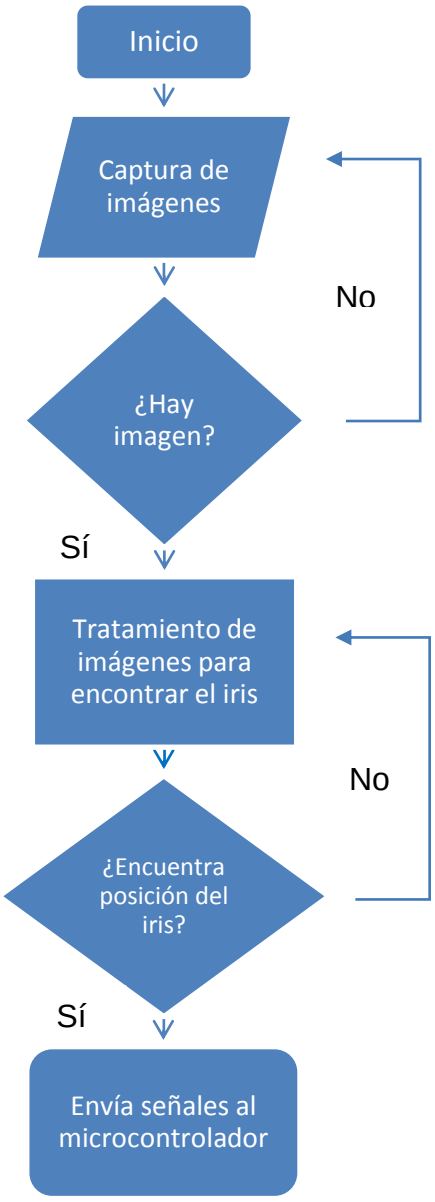
Dos motores son los que forman este bloque, conectados a los pines PWM del microcontrolador a través de los MOSFET IRFZ44M (p. 16). Los motores a utilizar son motores de corriente continua sin escobillas de 24V y 250W.

2.18. Bloque de alimentación

Dos baterías de ácido plomo recargables de 12V, conectadas en serie para obtener una fuente de 24V, son utilizadas para suministrar la alimentación

a los motores. Con un regulador de voltaje (p. 14), para alimentar con 5V a la Raspberry.

Figura 15. **Diagrama de flujo del bloque de procesamiento**



Fuente: elaboración propia.

3. DESCRIPCIÓN DEL PROGRAMA DE RASTREO

En este capítulo se detalla el código en Python utilizado para obtener y rastrear la posición de la pupila. Hay que tener en consideración que es indispensable tener instaladas las librerías Numpy, PySerial y OpenCV.

Figura 16. **Código del programa, parte 1**

```
1   import numpy as np
2   import cv2
3   import serial
4   import time
5
6   camara = cv2.VideoCapture(0)
7   cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
8   contador = 0
9   direccion = ""
10  enviar = serial.Serial('COM3', 115200, None)
11
12  while True:
13
14      (_, cuadro) = camara.read()
15      gris = cv2.cvtColor(cuadro, cv2.COLOR_RGB2GRAY)
16      borr = cv2.GaussianBlur(gris, (11,11), 0)
17      copia = cuadro.copy()
18
19      deteccion = cascade.detectMultiScale(borr, 1.3, 20)
20
21      try:
22          deteccion.any()
23          t1 = time.time()
```

Fuente: elaboración propia.

Figura 17. **Código del programa, parte 2**

```
24      except AttributeError:
25              t2 = time.time() - t1
26
27              if t2 > 3:
28                  enviar.write('d')
29
30      for (x,y,w,h) in deteccion:
31          cv2.rectangle(copia, (x,y), ((x+w), (y+h)), (0,255,0), 1)
32
33          corte = borr[y:y+h, x:x+w]
34          vent = cv2.resize(corte, (340, 340))
35
36          masc = cv2.inRange(vent, 0, 26)
37          ero = cv2.erode(masc, None, iterations = 2)
38          dil = cv2.dilate(ero, None, iterations =2)
39          canny = cv2.Canny(dil, 80, 90)
40
41          copia1 = cuadro[y:y+h, x:x+w]
42          salida = cv2.resize(copia1, (340, 340))
43
44          (_, cont, _) = cv2.findContours(canny.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
45
46          if len(cont) >0:
47              c = max(cont, key=cv2.contourArea)
48              ((x,y),r) = cv2.minEnclosingCircle(c)
49              centro = cv2.moments(c)
50              cx = int(centro['m10']/centro['m00'])
51              cy = int(centro['m01']/centro['m00'])
52              if r >26:
53                  cv2.circle(salida, (cx,cy), 10, (255,255,255), -1)
```

Fuente: elaboración propia.

Figura 18. **Código del programa, parte 3**

```
55
56                  (dirX, dirY) = ("", "")
57
58                  if cx > 200:
59                      dirX = "Izquierda"
60                      enviar.write('a')
61                  if cx < 150:
62                      dirX = "Derecha"
63                      enviar.write('b')
64
65                  if cy < 170:
66                      dirY = "Arriba"
67                      enviar.write('c')
68
69                  if dirX != "" and dirY != "":
70                      direccion = "{}-{}".format(dirY,dirX)
71                  else:
72                      direccion = dirX if dirX != "" else dirY
73
74      cv2.putText(salida, direccion, (50,30), cv2.FONT_HERSHEY_SIMPLEX, 0.65,
75                  (0,0,255), 3)
76
77      cv2.imshow('prubea 3', salida)
78      key = cv2.waitKey(1) & 0xFF
79
80
81      if key == ord("q"):
82          break
83
84  camara.release()
85  cv2.destroyAllWindows()
86  enviar.close()
```

Fuente: elaboración propia.

Figura 19. **Líneas 1 a 4**

```
1   import numpy as np
2   import cv2
3   import serial
4   import time
```

Fuente: elaboración propia.

En primeras 4 líneas se importan los paquetes que se necesitan para el correcto funcionamiento del programa. *Numpy* para manipular imágenes como vectores, *cv2* para utilizar los comandos de la librería de OpenCV, *serial* para hacer la comunicación de la Raspberry con el Microcontrolador y *time* para realizar operaciones basadas en tiempo.

Figura 20. **Líneas 6 a 10**

```
6   camara = cv2.VideoCapture(0)
7   cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
8
9   direccion = ""
10  enviar = serial.Serial('COM3', 115200, None)
```

Fuente: elaboración propia.

En esta sección se inicializan variables. La función *cv2.VideoCaputure* permite capturar un archivo de video (el argumento *0* indica que se captura desde una cámara), y se le asigna a la variable *cámara*. La función *cv2.CascadeClassifier* permite detectar objetos en un video. En este caso, la variable *cascade* almacena el archivo *haarcascade_eye.xml*, que se utiliza para

detectar los ojos en un rostro. El comando *serial Serial*permite abrir un puerto serial (para la Raspberry debe utilizarse el puerto *ttyUSB0)*.

Figura 21. **Líneas 12 a 17**

```
12  while True:
13
14      (_, cuadro) = camara.read()
15      gris = cv2.cvtColor(cuadro, cv2.COLOR_RGB2GRAY)
16      borr = cv2.GaussianBlur(gris, (11,11), 0)
17      copia = cuadro.copy()
```

Fuente: elaboración propia.

En la línea 12 se inicia el ciclo que hace funcionar el código indefinidamente. La función *read( )*, en la línea 14, retorna una tupla de dos valores: un indicador booleano cuando se obtiene exitosamente un cuadro de video y el cuadro de video en sí. Únicamente es necesario el cuadro de video, por lo que se almacena en la variable *cuadro*.

Figura 22. **Cuadro de video**

Fuente: elaboración propia.

En la línea 15 empieza el procesamiento. La función *cv2.cvtColor* cambia el color de todos los píxeles de una imagen, en este caso las imágenes guardadas en *cuadro*, a blanco y negro con el argumento *cv2.COLOR_RGB2GRAY*para trabajar con menor dificultad las imágenes.

Figura 23. **Cuadro en blanco y negro**

Fuente: elaboración propia.

Las imágenes deben ser difuminadas de primero para suavizarlas y que el resto del procesamiento tenga mejor rendimiento. En la línea 16 se utiliza la función *cv2.GaussianBlur*, la cual aplica un método Gaussiano para difuminar, que crea un efecto más natural a comparación de otros. Por último, se crea una copia de la imagen original para dibujar sobre ella, esto para fines ilustrativos.

En la línea 19 se hace la detección de los ojos, la función *detectMultiScale*re torna los objetos detectados como una lista de rectángulos (valores *x*, *y*, *w*, *h*). Es una lista porque se detectan en cada cuadro del video y no solamente una imagen. En el caso que no se detecte un ojo, se inicia un temporizador. Si han pasado 2 segundos o más, se envía una señal al Microcontrolador que le indica que detenga los motores.

Figura 24. **Cuadro difuminado**

Fuente: elaboración propia.

Figura 25. **Líneas 19 a 28**

```
19      deteccion = cascade.detectMultiScale(borr, 1.3, 20)
20
21      try:
22          deteccion.any()
23          t1 = time.time()
24      except AttributeError:
25              t2 = time.time() - t1
26
27              if t2 > 3:
28                  enviar.write('d')
```

Fuente: elaboración propia.

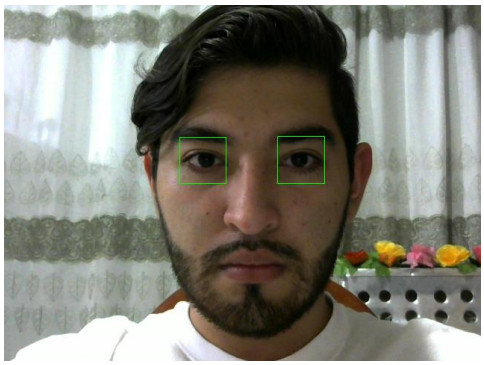
En la línea 30 se obtienen los valores de los rectángulos cada vez que detecta un ojo y en la 31 se dibujan sobre el área que abarcan los ojos. Esto último para fines ilustrativos. Debido a que las funciones para dibujar son destructivas, los dibujos se realizan en copias de las imágenes originales.

Figura 26. **Líneas 30 y 31**

```
30        for (x,y,w,h) in deteccion:
31            cv2.rectangle(copia, (x,y), ((x+w), (y+h)), (0,255,0), 1)
```

Fuente: elaboración propia.

Figura 27. **Detección de ojos**



Fuente: elaboración propia.

En la línea 33 se realiza un recorte a la imagen a blanco y negro y difuminada para que solo se trabaje sobre el área del ojo. En la línea 34 se establece un tamaño fijo de 340x340 píxeles para la imagen recortada, ya que esta va variando por cada cuadro capturado.

Figura 28. **Líneas 33 y 34**

```
33            corte = borr[y:y+h, x:x+w]
34            vent = cv2.resize(corte, (340, 340))
```

Fuente: elaboración propia.

Figura 29. **Ojo recortado**

Fuente: elaboración propia.

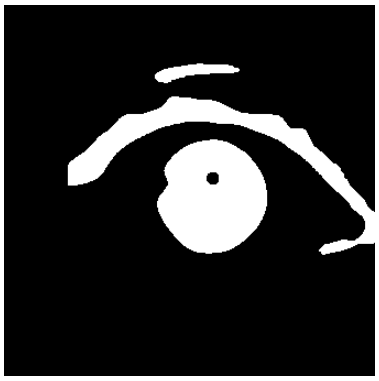
Figura 30. **Líneas 36 a 39**

```
36          masc = cv2.inRange(vent, 0, 26)
37          ero = cv2.erode(masc, None, iterations = 2)
38          dil = cv2.dilate(ero, None, iterations =2)
39          canny = cv2.Canny(dil, 80, 90)
```

Fuente: elaboración propia.

Se aplica una máscara binara que delimita los píxeles con los cuales se debe trabajar con la función *cv2.inRange*. Esta función muestra los píxeles que se encuentran dentro del rango establecido, en este caso los píxeles con intensidad entre 0 y 26.

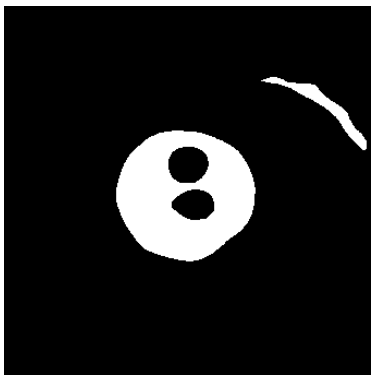
Figura 31. **Ojo con máscara**



Fuente: elaboración propia.

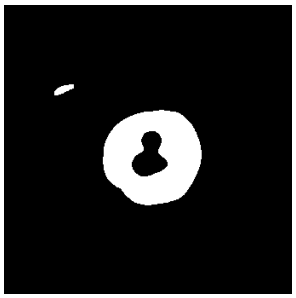
La imagen resultante es erosionada y dilatada en las líneas 37 y 38 para remover masas innecesarias o ruido que podría encontrarse en la imagen.

Figura 32. **Imagen erosionada**



Fuente: elaboración propia.

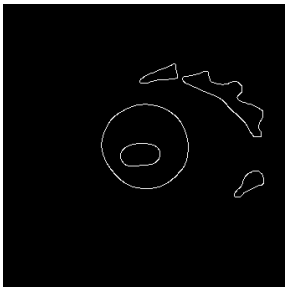
Figura 33. **Imagen dilatada**



Fuente: elaboración propia.

Se aplica el detector de bordes *Canny* en la línea 39 utilizando la función *cv2.Canny*. El primer argumento suministrado es la imagen procesada anteriormente. Luego, se necesitan 2 límites. Cualquier valor por debajo del primer límite no es considerado como un borde y cualquier valor por encima del segundo límite es considerado como un borde. Los valores que quedan en medio de este rango son clasificados como borde o no borde, dependiendo de la intensidad asociada a cada pixel. Este paso se realiza para que, más adelante, se puedan filtrar todas las manchas pequeñas que pueden ser confundidas como masas redondas.

Figura 34. **Imagen con bordes**



Fuente: elaboración propia.

Nuevamente se hace una copia y un recorte de la imagen original para ilustrar el resultado más adelante.

Figura 35. **Líneas 41 y 42**

```
41          copia1 = cuadro[y:y+h, x:x+w]
42          salida = cv2.resize(copia1, (340, 340))
```

Fuente: elaboración propia.

OpenCV provee un método para encontrar contornos dentro de una imagen. En orden para identificar los contornos, es necesario que se haya aplicado algún método de detección de bordes en la imagen que es procesada. En la línea 44 se utiliza la función *cv2.findContours* que devuelve una tupla de 3 valores: la imagen después de haber aplicado la detección de contornos, una lista con los contornos detectados y la jerarquía de los mismos. El valor de interés es la lista con los contornos. Por lo tanto, se almacena en la variable *cont*.

Seguidamente, se entra en otro ciclo cuando existe algún contorno dentro de la lista, es decir que el valor de *cont* sea mayor a 0. En la línea 47 se obtiene el área del contorno más grande dentro de la lista y en la línea 48 se encuentra el círculo (coordenadas y radio), más pequeño que pueda encerrarla.

En la línea 49 se computa el centro de del objeto. En las líneas 50 y 51 se calculan las coordenadas aproximadas (*x*, *y*), del centro. En la línea 52 se hace un chequeo para asegurar que el radio del círculo que cubre el área sea lo suficientemente grande y así no identificar círculos fantasmas. Con fin ilustrativo, en la línea 53 se dibuja un círculo con las coordenadas del centro de la masa.

Figura 36. **Líneas 44 a 53**

```
44          (_, cont, _) = cv2.findContours(canny.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
45
46          if len(cont) >0:
47              c = max(cont, key=cv2.contourArea)
48              ((x,y),r) = cv2.minEnclosingCircle(c)
49              centro = cv2.moments(c)
50              cx = int(centro['m10']/centro['m00'])
51              cy = int(centro['m01']/centro['m00'])
52              if r >26:
53                  cv2.circle(salida, (cx,cy), 10, (255,255,255), -1)
```

Fuente: elaboración propia.

Figura 37. **Líneas 55 a 74**

```
55                      (dirX, dirY) = ("", "")
56
57                      if cx > 200:
58                          dirX = "Izquierda"
59                          enviar.write('a')
60                      if cx < 150:
61                          dirX = "Derecha"
62                          enviar.write('b')
63
64                      if cy < 170:
65                          dirY = "Arriba"
66                          enviar.write('c')
67
68                      if dirX != "" and dirY != "":
69                          direccion = "{}-{}".format(dirY,dirX)
70                      else:
71                          direccion = dirX if dirX != "" else dirY
72
73          cv2.putText(salida, direccion, (50,30), cv2.FONT_HERSHEY_SIMPLEX, 0.65,
74                      (0,0,255), 3)
```

Fuente: elaboración propia.

Una vez identificadas las coordenadas del centro del ojo, se determina la posición y, dependiendo de esta, se envía una señal al microcontrolador para que los motores giren al lado al cual se está viendo. Las instrucciones de las líneas 55, 58, 61, 65 y 68 a la 74 son para mostrar la dirección en pantalla, no son necesarias para el funcionamiento del código, solamente para fines ilustrativos. Las figuras a continuación están volteadas en modo espejo.

Figura 38. **Detección de iris: centro**

Fuente: elaboración propia.

Figura 39. **Detección de iris: izquierda**

Fuente: elaboración propia.

Figura 40. **Detección de iris: derecha**

Fuente: elaboración propia.

Figura 41. **Detección de iris: arriba**

Fuente: elaboración propia.

La parte final del código es para indicar al programa que el procesamiento termine, deje de utilizar la cámara y que cierre las ventanas abiertas por el programa. No son vitales para el funcionamiento final. Son solamente para fines ilustrativos a excepción de la línea 85, donde se cierra el puerto serial abierto al inicio del programa.

Figura 42. **Líneas 76 a 85**

```
76          cv2.imshow('prubea 3', salida)
77          key = cv2.waitKey(1) & 0xFF
78
79
80          if key == ord("q"):
81              break
82
83      camara.release()
84      cv2.destroyAllWindows()
85      enviar.close()
```

Fuente: elaboración propia.

Por último, se configura la Raspberry para que el código sea ejecutado cada vez que es encendida.

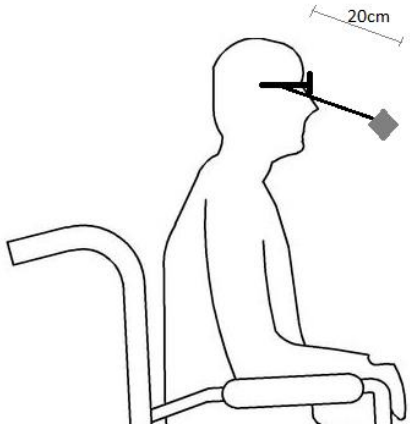
4. DISEÑO FINAL

El sistema se podrá adaptar a cualquier silla de ruedas ordinaria. Con todas las partes ensambladas, el diseño final quedará de la manera descrita a continuación.

El sistema contará con un interruptor principal para encenderlo y apagarlo, un botón para paro de emergencia en caso que lo amerite y un sensor, al pie de la silla de ruedas, para detectar cuando se aproxime a un obstáculo y enviar una señal para alertar al usuario que cambie de dirección o se detenga.

Anteojos sin espejuelos con la cámara adaptada a una distancia de 20cm del ojo de cual se tomarán las imágenes.

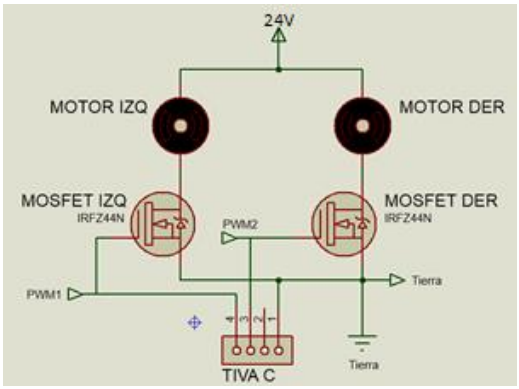
Figura 43. **Posición de anteojos con la cámara**



Fuente: elaboración propia.

Una caja de control donde estarán las conexiones de la Raspberry, el microcontrolador, la cámara, el sensor ultrasónico, una bocina pequeña, y los motores.

Figura 44. **Diseño de circuito de control de motores**



Fuente: elaboración propia.

Un compartimiento donde estarán las baterías conectadas a los motores y un espacio para recargarlas cuando sea necesario.

Figura 45. **Compartimiento de baterías**

Fuente: elaboración propia.

El funcionamiento de los motores quedará de la siguiente manera:

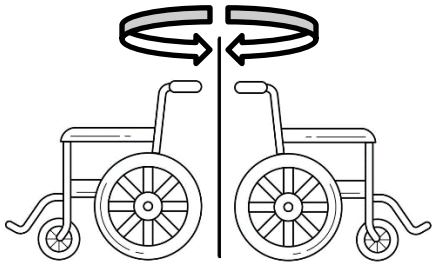
- Para avanzar, el usuario deberá ver hacia arriba por 2s. La velocidad en marcha será de 2Km/h.

- Para detenerse, el usuario deberá cerrar el ojo durante 2s. Se detendrá gradualmente.

- Para moverse a la izquierda, el usuario deberá ver hacia su izquierda; si ve a la derecha, se moverá hacia esa dirección. Si está en movimiento, se reducirá la velocidad del motor asociado a la dirección a la que desea ir. Es decir, Si ve a la izquierda mientras avanza, se reducirá la velocidad del motor izquierdo. De igual manera sucede con el motor derecho si se ve hacia ese lado.

- No tendrá opción de retroceso. Por lo tanto, si el usuario desea dirigirse para atrás, deberá detenerse y ver hacia la izquierda o derecha hasta haber girado 180 grados de su posición original o hasta donde desee.

Figura 46. **Cambio de orientación**



Fuente: elaboración propia.

CONCLUSIONES

1. A través del análisis y procesamiento digital de imágenes, se desarrolló un programa capaz de rastrear el movimiento de los ojos.

2. El programa puede rastrear el movimiento de los ojos en personas que utilizan anteojos o lentes de contacto.

3. El color del iris no es un factor que impida la detección del mismo. El programa puede ser modificado para cumplir de igual manera su función.

4. Mediante la realización de este proyecto, se demostró que es posible la implementación de sistemas electrónicos para la solución de problemas en el ámbito médico en Guatemala.

5. Se brinda independencia de movimiento al paciente, siempre y cuando el sistema tenga carga o el mismo paciente sea capaz de ponerlo a cargar.

6. El programa de detección de los ojos puede ser utilizado como base para el desarrollo de aplicaciones de realidad virtual, realidad aumentada, seguridad, control de calidad, monitoreo, entre otros.

RECOMENDACIONES

1. Deben realizarse pruebas para cada usuario final y asegurarse del correcto funcionamiento del sistema. Hacer los ajustes adecuados en caso que sean necesarios.

2. Utilizarse en ambiente con buena iluminación, ya que todo el sistema depende de las imágenes que captura la cámara.

3. No utilizarse bajo la lluvia o ambientes donde se corra el riesgo de mojar el sistema. Es un sistema electrónico, por lo que el agua puede causar un cortocircuito o un malfuncionamiento.

4. De ser posible, optimizar el código o implementar mejoras en él.

5. El sistema está enfocado a personas cuya capacidad de movimiento es limitada. Por lo tanto, si lo utiliza una persona con discapacidad de movimiento total, se recomienda que una persona supervise su actividad y auxilie cuando se deba colocar los anteojos o al momento de recargar las baterías.

6. Colocarle un disipador de calor a la Raspberry para que su funcionamiento sea óptimo.

BIBLIOGRAFÍA

1. ALLER, José Manuel. *Máquinas eléctricas rotativas: Introducción a la teoría general*. Departamento de Conversión y Transporte de Energía, Universidad Simón Bolívar. Venezuela, 2008. 117 p.

2. BARRIENTOS, David. *Introducción al diseño de sistemas embebidos.* Guatemala: Laboratorio de Electrónica, Facultad de Ingeniería, Universidad de San Carlos de Guatemala, 2017. 23 p.

3. BARRETT, Kim, et. al. *Ganong. Fisiología Médica*. 24a ed. México: McGraw-Hill, 2013. 195 p.

4. BOYLESTAD, Robert y NASHELSKY, Louis. *Electrónica: Teoría de Circuitos y Dispositivos Electrónicos.* 10a ed. México: Pearson Educación, 2009. 386 p.

5. DE ALBA, Carlos. y MUÑOZ, María*. Estudio y fabricación de una batería ácido plomo*. [en línea]. <http://www.utm.mx/edi_anteriores/temas61/T61_1E3_Estudio_y_fabricacion_bateria.pdf>. [Consulta: 13 abril 2018].

6. OpenCV Team. [en línea] <https://opencv.org/about.html>. [Consulta: 10 de marzo de 2018].

7. Texas Instruments. *Tiva™ C Series TM4C123G LaunchPad Evaluation Board: User's Guide.* [en línea]. <http://www.ti.com/lit/ug/spmu296/spmu296.pdf>. [Consulta: 10 de marzo de 2018].

8. Raspberry Pi Foundation. *Raspberry Pi Getting Started.* [en línea]. <https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started/3>. [Consulta: 10 de marzo de 2018].

9. ROSEBROCK, Adrian. *Practical Python and OpenCV: An Introductory, Example Driven Guide to Image Processing and Computer Vision.* 2a ed. Estados Unidos: Pyimagesearch. 2016. 104 p.