Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería Electrónica

Robot navegador con detección de movimiento

Modalidad: Proyecto Tecnológico
Trimestre 2020 Otoño

Alumno:
César Angeles Barrios
Matrícula: 209205439
Correo electrónico
cesar.angeles.barrios@gmail.com

VoBo.
Asesor:
José Ignacio Vega Luna
Profesor Titular
Departamento: Electrónica
Correo electrónico
vlji@azc.uam.mx

Asesor:
Gerardo Salgado Guzmán
Profesor Titular
Departamento: Electrónica
Correo electrónico
gsg@azc.uam.mx

Ciudad de México, 10 de marzo de 2021

Declaratoria

Yo, José Ignacio Vega Luna, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

José Ignacio Vega Luna

Yo, Gerardo Salgado Guzmán, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Gerardo Salgado Guzmán

Yo, César Angeles Barrios, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

César Angeles Barrios

Resumen

En esta propuesta se realizó el armado y programación de un robot móvil que hace un recorrido previamente establecido en el perímetro interior de la sala de equipos de un centro de datos. En caso de que detecte movimiento procede a capturar una imagen del área donde detecta dicho movimiento.

La imagen capturada se analiza utilizando una técnica de sustracción de fondo a través de la librería OpenCV del lenguaje Python y envía un mensaje al teléfono móvil del responsable de la seguridad y a un servidor FTP ubicado en el CPD (Centro de Procesamiento de Datos). Esto permite al responsable de la seguridad acceder a la información desde Internet, con la posibilidad de identificar la persona capturada en la imagen o lo que haya generado la alerta de movimiento.

Dedicatoria

A mi madre,

por siempre apoyarme y sacarme adelante.

A mi padre,

por ayudarme a pesar de la distancia.

A Lizbeth Noguez Moreno,

por ser mi compañera y estar a mi lado siempre, eres mi inspiración.

A mi hija Akari Vanessa,

siempre me animas a seguir adelante.

A mi asesor José Ignacio Vega Luna,

por compartir su conocimiento y gran guía en esta etapa.

A mis amigos y compañeros,

todos los recuerdos y experiencias adquiridos juntos me ayudaron a crecer y mejorar.

Contenido

Índice de Figuras

Introducción

Con la acelerada evolución de la Internet y la necesidad de estar conectadas en todo momento, las empresas e instituciones se ven obligadas a contar con un alto nivel de confiabilidad y seguridad en la operación de sus sistemas de información, ubicando los equipos de cómputo, telecomunicaciones y de almacenamiento en un CPD (Centro de Procesamiento de Datos). De esta forma garantizan la continuidad del servicio a clientes, empleados, proveedores y colaboradores. Es vital que el CPD cuente con mecanismos seguros y eficientes para el funcionamiento del equipo, ya que contiene información crítica y necesaria para las operaciones diarias de la empresa a fin de evitar poner en riesgo la productividad y el negocio.

Los CPD son auditados periódicamente por organismos y empresas externas para poder estar certificados y ofrecer servicios garantizados a sus clientes. Un punto importante que consideran las auditorías son los procedimientos y técnicas usados en la seguridad lógica y física. La seguridad lógica tiene como fin proteger la confidencialidad, integridad, disponibilidad y autenticidad de la información contra amenazas físicas y cibernéticas. La seguridad física consiste en establecer los procedimientos y medidas que aseguran la integridad física de equipos instalados, incluyendo: sistemas de control de acceso, de monitoreo y control de mecanismos de variables de ambiente, de vigilancia y de suministro de energía y recursos [1].

Los sistemas de vigilancia en los centros de datos usan cámaras de video y sensores de presencia instalados para cubrir todos los puntos de las instalaciones desde una oficina central de control y monitoreo. Los sistemas de vigilancia también tienen como objetivo la seguridad perimetral interna y externa del CPD. El acceso a las salas de equipo de cómputo es controlado. Solo está permitida la entrada a personal autorizado de administración y mantenimiento. Con el desarrollo de la tecnología, las auditorias actualmente exigen la monitorización remota desde la Internet de variables, procesos, procedimientos y acceso [2].

Antecedentes

Se han realizado varios trabajos relacionados con el aquí propuesto que son los siguientes:

1- “Robot para obtención de imágenes de objetos en movimiento” [3]. Se realizó en agosto de 2007 en la UAM Iztapalapa. Objetivo: captar y manipular imágenes de un objeto en movimiento, usando software especializado en el tratamiento de imágenes y así poder obtenerlas de buena calidad y “útiles”. Las diferencias de este trabajo respecto al aquí propuesto son en general, en esta propuesta se usará lo desarrollado para una aplicación específica, la vigilancia y monitoreo de un CPD. A diferencia de la propuesta aquí planteada, el proyecto se enfocó en el manejo y procesamiento de las imágenes para ser más nítidas y claras. En la actualidad, la mayoría de los dispositivos para fotografía digital integran las tareas llevadas a cabo en el trabajo realizado para reducir los efectos negativos al capturar un objeto en movimiento.

2- “Localización y mapeo simultaneo de un robot” [4]. Realizada en 2014 en la UAM Iztapalapa tuvo como objetivo el desarrollo e implementación de la localización de un robot móvil y del mapeado simultáneo de un entorno cerrado, basándose en la información recibida por un sistema compuesto por odometría y múltiples sensores. Las diferencias de este trabajo respecto al aquí propuesto son: el robot aquí propuesto se usará para una aplicación específica, el mapeo se realizará para poder

obtener la detección de movimiento más que para encontrar una ruta de salida de un laberinto. Además, en el proyecto realizado anteriormente se usó el recurso de la plataforma Webots para simular y programar al robot.

3- "Robot barredor autónomo" [5]. Elaborada en 2003 en la UAM Iztapalapa, el objetivo principal del robot fue realizar la limpieza de un espacio físico, no importando si en este hay obstáculos o no. Para el robot barredor autónomo que se diseñó, se optó por implementarlo en base de la teoría de las cartas ASM. La mayor diferencia entre este proyecto y la propuesta aquí elaborada es la aplicación del robot, en esta propuesta el robot se usará como base para el sistema de detección de movimiento, estableciendo una ruta definida en lugar de un barrido amplio para cubrir la mayoría de la superficie en el recorrido.

4- "Navegación de un robot móvil para la supervisión de una residencia monitoreada vía Internet" [6]. Realizada en 2018 en la Facultad de Ingeniería de UNAM, tuvo como objetivo integrar un robot móvil como un nuevo dispositivo dentro del ecosistema de SmartThings. Así como construir y controlar el robot móvil, con la finalidad de implementar un sistema que le permita navegar de forma autónoma y tele operada en un ambiente de trabajo casero y estático. Siendo este el proyecto, usado como antecedente, más parecido al expuesto en esta propuesta, se diferencia principalmente al entorno en que se desenvuelve, siendo usado en un hogar principalmente, mientras que en esta propuesta es para un CPD. Otro punto para destacar es que en esta propuesta se realizará la detección de movimiento mientras que en el proyecto mencionado se incluyen diferentes sensores para realizar otras tareas, como la detección de fugas de gas o la presencia de humo en el ambiente, además de que se usó como principal ambiente de desarrollo el ecosistema de SmartThings mientras que en esta propuesta se plantea usar Python para programar los diferentes módulos en la tarjeta Arty Z7-10.

5- "Sistema cliente servidor para visión de un robot móvil usando una Wireless LAN" [7]. Fue realizado en 2006 en la Facultad de Ciencias e Ingeniería de la Pontificia Universidad Católica del Perú, tuvo por finalidad, analizar, desarrollar y demostrar el modo de aplicación de un Sistema Cliente-Servidor que transmite imágenes digitales, captadas por un robot móvil hacia una PC remota o servidor. La primera diferencia visible es que este proyecto se llevó a cabo el envío de la información captada a una PC o servidor remoto, mientras que en esta propuesta se enviará la información a un dispositivo móvil o teléfono inteligente a través de un SMS y un mensaje de WhatsApp. Otra diferencia es que en el trabajo realizado los dispositivos a los que se envía la información deben estar conectados al mismo segmento de red local, mientras que en esta propuesta el dispositivo del usuario al que se notificará la alerta de movimiento estará conectado en la Internet.

Por otra parte, durante los últimos años se ha desarrollado una cantidad grande de investigaciones y trabajos de robótica, con características y funciones diversas que incorporan tecnologías recientes para resolver necesidades en muchos campos de la vida humana y procesos industriales. Particularmente, el desarrollo de robots de dos ruedas se ha enfocado en tres líneas: algoritmos de navegación, detección de obstáculos y aplicaciones [8]. Los algoritmos de navegación recientes se basan en el desarrollo de: conjuntos de robots que trabajan cooperativamente para evitar colisiones usando sumatorias lineales y funciones de cuellos de botella [9], el método de localización de Monte Carlo [10], sistemas operativos robóticos (ROS-Robot Operating System), curvas de Bézier, seguimiento usando video para identificar el movimiento de objetos y redes neuronales [11] y asistentes activados por voz [12]. En lo que respecta a la detección de obstáculos, las últimas

investigaciones realizadas usan: sensores ultrasónicos, sistemas de medida y detección de objetos con láser (LIDAR-Laser Imaging Detection and Ranging) y señales de radar [13], lógica difusa [14], control cooperativo [15], sistemas de visión y control adaptivo por visión [16]. Se ha creado un gran número de aplicaciones de robots exploradores, entre las que se encuentran: rescate de personas y agricultura, recolección de basura, exploración alpina, minas, terrenos rocosos y submarina, realización de cirugías y plantaciones.

Justificación

La detección de objetos, personas y movimiento de estas se realiza usando dos técnicas diferentes. La primera utiliza dispositivos electrónicos que incluyen sensores, cámaras de video o cámaras térmicas que activan un actuador o registran el evento. La segunda técnica combina el uso de un dispositivo, como los indicados anteriormente, y un algoritmo o método de visión por computadora. Esta técnica ha sido un tema estudiado extensamente dada su importancia en diversas aplicaciones de visión por computadora. La captura y procesamiento de las imágenes puede ser en dos tipos de escenas: donde la cámara está fija y el que se mueve es el objeto por detectar o donde la cámara y el objeto están en movimiento. En el primer escenario existen diferentes algoritmos de los cuales la mayoría se fundamenta en la sustracción de fondo o la diferencia temporal. La técnica o algoritmo usado depende de la aplicación, necesidades y escenario. En este trabajo no se utilizó un algoritmo sofisticado, se planteó usar el de extracción o sustracción de fondo o segundo plano ya que ha demostrado ser el más efectivo para la detección de movimiento por su sencillez, tiempo de respuesta y porque no consume tiempo excesivo de procesamiento.

No se ha realizado hasta ahora una aplicación como la aquí presentada para un centro de datos, en la cual el robot estará equipado con un módulo de navegación, un módulo de detección de movimiento y un servidor web, controlados por una tarjeta Arty Z7-10. Las ventajas y aportaciones de este trabajo son las siguientes: A) El uso del robot no es intrusivo, ya que el acceso a la Internet es inalámbrico y no es necesario modificar la infraestructura del cableado del centro de datos; B) Se encuentra disponible comercialmente una gran variedad de soluciones de detección de movimiento que integran una cámara de video, la cual se encuentra permanentemente capturando y almacenando la señal de video. La ventaja de esta aplicación es capturar la imagen donde detecta movimiento y en tiempo real transmitirla al teléfono móvil permitiendo tomar decisiones y reaccionar oportunamente; C) La tecnología usada es de reciente creación, las herramientas software usadas son de código abierto lo cual reducirá la complejidad y costo del sistema; y D) El sistema desarrollado resuelve una necesidad real, monitoreando desde la Internet la presencia de personas en la sala de equipos.

Objetivos

General

Diseñar y construir un robot móvil para navegar en el perímetro interior de un centro de datos y detectar movimiento por medio del método de sustracción de fondo.

Específicos

1- Desarrollar un robot móvil que incluya la tarjeta de control, sensores ultrasónicos y cámara de video.
2- Desarrollar el algoritmo de movimiento y navegación del robot evitando obstáculos por medio de sensores ultrasónicos.
3- Implantar el algoritmo de movimiento de la cámara de -90° a 90° usando una estructura de dos servomotores.
4- Contar con las imágenes del entorno usadas como primer plano.
5- Desarrollar el algoritmo del método de sustracción de fondo para la detección de movimiento.
6- Implantar el servidor web para mostrar la interfaz de usuario y acceder las imágenes donde se detectó movimiento.

Marco Teórico

Placa de desarrollo Arty Z7 Zynq-7000

La Arty Z7-10 (Figura 4) es una plataforma de desarrollo diseñada en torno al sistema en chip totalmente programable (AP SoC) Xilinx Zynq-7000™. La arquitectura de Zynq-7000 integra un núcleo doble, procesador ARM Cortex-A9 de 650 MHz con lógica FPGA serie Artix-7. Este emparejamiento rodea a un potente procesador con un conjunto único de controladores y periféricos definidos por software. Los conjuntos de herramientas Vivado, Petalinux y SDSoC proporcionan una

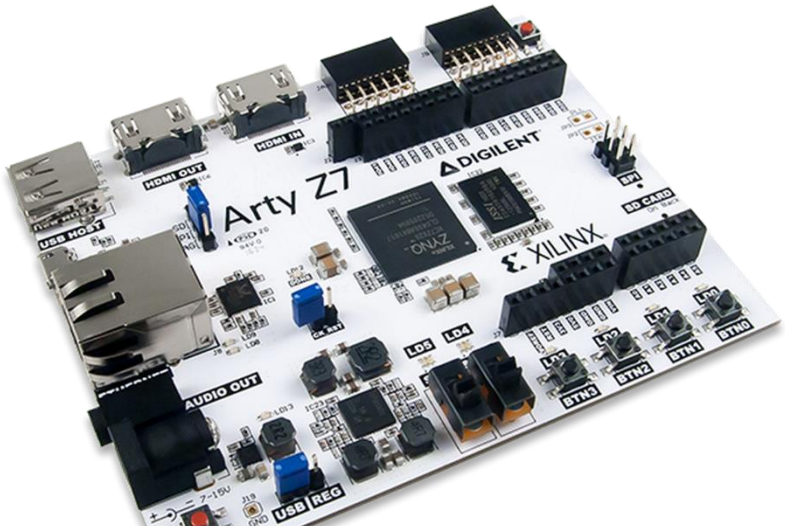

Figura 1: Tarjeta Arty Z7-10

ruta entre definir el conjunto de periféricos personalizado y llevar su funcionalidad hasta un programa o Linux OS en ejecución en el procesador [17].

Python

Python es un lenguaje de programación interpretado de tipado dinámico cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma y disponible en varias plataformas.

Al hacer uso de una sintaxis legible, la curva de aprendizaje es muy rápida, siendo de este modo, uno de los mejores lenguajes para iniciarse en la programación en modo texto. Por ejemplo, si comparamos un código escrito en lenguaje de programación por bloques como Scratch y el mismo código lo escribimos utilizando Python, vemos las similitudes en las instrucciones.

Python contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero. Pero lo que más resultó útil es por la compatibilidad con la tarjeta Arty Z7-10 y la capacidad de poder utilizar los pines GPIO para conectar el mundo físico con el mundo digital [18].

Componentes físicos del robot

- 1 módulo de cámara 5MP (Figura 2).
- 1 placa de controlador L293D.
- 1 tarjeta SD de 16 GB.
- 1 módulo de sensor ultrasónico HC-SR04.
- 1 módulo de seguimiento de línea.
- 1 servomotor micro de 9 g.
- 1 soporte para HC-SR04.
- 1 chasis de coche.
- 4× Motores engranados (1:48).
- 4 ruedas de 65 mm.
- 1 contenedor de batería.
- 2 pilas.
- 1 cargador de batería.

Figura 2: Módulo de cámara 5MP

OpenCV

OpenCV (Biblioteca de visión por computadora de código abierto: http://opencv.org) es una biblioteca de código abierto que incluye varios cientos de algoritmos de visión por computadora. El documento describe la llamada API OpenCV 2.x, que es esencialmente una API C ++, a diferencia de la API OpenCV 1.x basada en C (la API C está obsoleta y no se ha probado con el compilador "C" desde las versiones de OpenCV 2.4) [19].

OpenCV tiene una estructura modular, lo que significa que el paquete incluye varias bibliotecas compartidas o estáticas. Están disponibles los siguientes módulos:

- Funcionalidad principal (núcleo): un módulo compacto que define las estructuras de datos básicas, incluido el denso arreglo multidimensional Mat y las funciones básicas utilizadas por todos los demás módulos.
- Procesamiento de imágenes (imgproc): un módulo de procesamiento de imágenes que incluye filtrado de imágenes lineales y no lineales, transformaciones geométricas de imágenes (cambio de tamaño, deformación afín y perspectiva, reasignación genérica basada en tablas), conversión de espacio de color, histogramas, etc.
- Análisis de video (video): un módulo de análisis de video que incluye algoritmos de estimación de movimiento, sustracción de fondo y seguimiento de objetos.
- Calibración de cámara y reconstrucción 3D (calib3d): algoritmos básicos de geometría de múltiples vistas, calibración de cámara única y estéreo, estimación de pose de objeto, algoritmos de correspondencia estéreo y elementos de reconstrucción 3D.

- Marco de características 2D (features2d): detectores de características destacadas, descriptores y comparadores de descriptores.
- Detección de objetos (objdetect): detección de objetos e instancias de las clases predefinidas (por ejemplo, rostros, ojos, tazas, personas, automóviles, etc.).
- GUI de alto nivel (highgui): una interfaz fácil de usar para funciones de interfaz de usuario simples.
- E/S de video (videoio): una interfaz fácil de usar para la captura de video y los códecs de video.
- ... algunos otros módulos auxiliares, como los contenedores de prueba de FLANN y Google, los enlaces de Python y otros.

Cloudinary

Cloudinary fue fundada en 2012 por tres veteranos militares de alta tecnología y desarrolladores experimentados que continuaron enfrentándose al mismo conjunto de tediosos desafíos de gestión de medios una y otra vez. Al igual que los desarrolladores, tomaron medidas y crearon una solución para cambiar por completo la forma en que los medios digitales podían administrarse, transformarse y entregarse en el mundo omnicanal y siempre activo de hoy.

Es una compañía de tecnología SaaS con sede en Santa Clara, California, y una oficina en Israel. La compañía ofrece servicios de gestión de imagen y video basados en la nube. Permite a los usuarios cargar, almacenar, administrar, manipular y entregar imágenes y videos para sitios web y aplicaciones [20].

Programación del robot

Para la programación del robot se dividió el proceso en 4 etapas:

- La primera consistió en desarrollar el algoritmo para que el robot recorriera el perímetro a vigilar con autonomía.
- En la segunda se enfocó el trabajo a la captura de imágenes, tanto las de base como aquellas que se van a comparar.
- Para la tercera se implementó el algoritmo para la comparación de las imágenes y así poder detectar movimiento o variaciones entre ellas.
- En la cuarta se realizó el envío del mensaje para notificar en caso de que haya una alerta de movimiento.

Desarrollo del Proyecto

Armado del robot

Para el armado del robot se ensamblaron al chasis de madera laminada los 4 motores conectados a cada rueda, teniendo cuidado de que la orientación de los motores sea la correcta. Al frente del chasis se colocaron, en la parte inferior, tanto el sensor ultrasónico HC-SR04 en su soporte, como el sensor de seguimiento de línea, mientras que en la parte superior se colocó el módulo de la cámara de 5MP con sus servomotores. Todos estos componentes van conectados a la placa de controlador L293D y dicha placa se coloca sobre la tarjeta Arty Z7-10 la cual va sujeta a la parte central de chasis, junto al contenedor para las baterías, el cual se conecta a la tarjeta para alimentar a todo el robot.

Figura 3: Componentes del robot

Arquitectura de la tarjeta Arty Z7-10

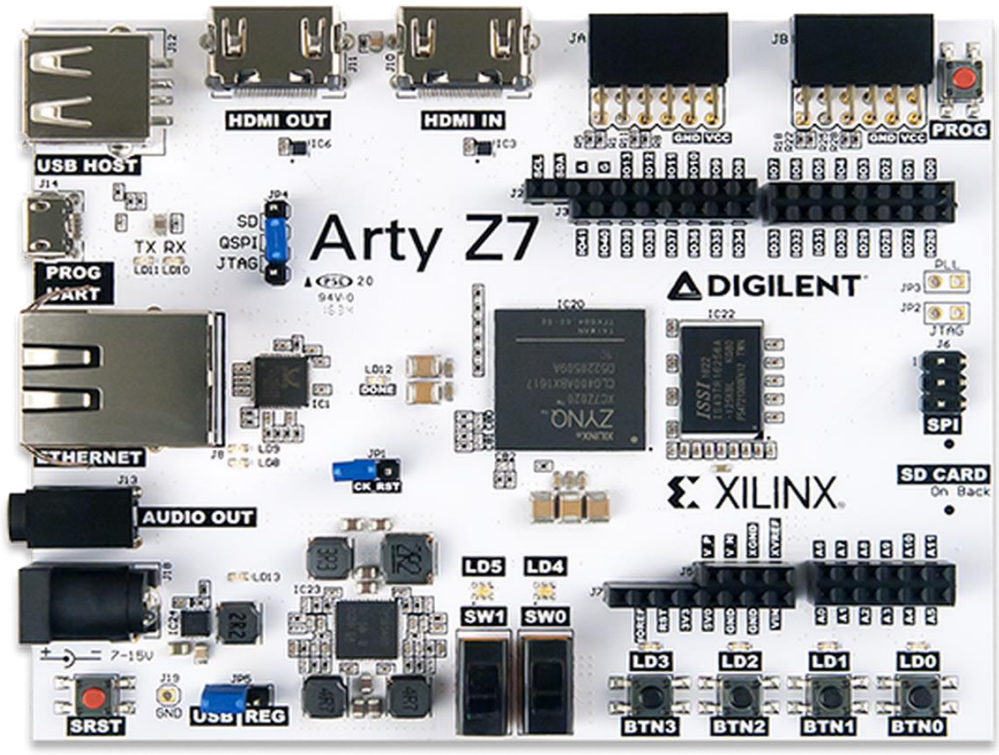

Figura 4: Arquitectura de la tarjeta Arty Z7-10

Xilinx Zynq-7000 (XC7Z010-1CLG400C)

- Procesador: núcleo ARM Cortex-A9 dual de 650 MHz
- Tablas de búsqueda (LUTs): 53200
- Flip-flops: 106400
- Memoria RAM de bloque: 630 KB
- Pilas de administración de reloj: 4
- E/S de placa disponibles: 49
- Controlador de memoria DDR3 con 8 canales DMA y 4 puertos AXI3 esclavos
- Controladores de periféricos de gran ancho de banda: 1G Ethernet, USB 2.0, SDIO
- Controlador de periféricos de bajo ancho de banda: SPI, UART, CAN, I^2C
- Programable desde JTAG, Flash Quad-SPI o tarjeta microSD
- Lógica programable equivalente a memoria Artix-7 FPGA

Memoria en placa y alimentación

• SDRAM DDR3 de 512 MB IS43TR16256A-125KBL integrada con bus de 16 bits a 1.050 Mbps

• Flash Quad-SPI de 16 MB S25FL128S integrada con identificador compatible con EUI-48/64™ único de 48 bits

• Ranura de tarjeta microSD

• Alimentación por USB o +7 V a +15 V dc a través de conector cilíndrico

Conectividad del sistema

• 10/100/1000 Ethernet a través de conector RJ45

• Puente USB-UART a través de conector hembra micro-B

• Programación USB-JTAG a través de conector hembra micro-B

• USB OTG PHY (admite solo Host) a través de conector hembra USB-A

Audio y vídeo

• Puerto con disipador HDMI (entrada)

• Puerto fuente HDMI (salida)

• Salida de audio mono de accionamiento PWM a través de conector hembra jack de 3,5 mm

Interruptores y LED

• Pulsadores de reset y programación

• 4 botones pulsadores de usuario

• 2 interruptores deslizantes de usuario

• 4 LED de usuario

• 2 LED RGB de usuario

Conectores de expansión

• Conectores de placa chipKIT/Arduino R3

• 2 conectores Pmod™ de 12 contactos estándar

Un FPGA es un dispositivo semiconductor compuesto de una matriz de bloques lógicos configurables (CLB) conectados mediante interconexiones programables. El usuario determina estas interconexiones mediante programación SRAM. Un CLB puede ser simple (puertas Y, O, etc) o

complejo (un bloque de RAM). FPGA permite realizar cambios en un diseño incluso después de haber soldado el dispositivo en un PCB [17].

Programación del robot

Para elaborar la programación del robot se inició con el módulo de movimiento del mismo, inicialmente se consideró que fuera de dos ruedas pero al llegar al módulo de procesamiento y comparación de imágenes se tuvo el problema que al presentar la más mínima diferencia entre la imagen base y la imagen recién tomada notificaba que había dicha diferencia lo que provocaba que se notificara constantemente sin una relevancia real.

Para evitar esto, se hicieron pruebas con cuatro ruedas, lo cual proporcionó mayor estabilidad para la captura de imágenes, además de agregar un sensor seguidor de línea para definir correctamente el circuito a recorrer y los lugares donde se debían capturar dichas imágenes (ver Apéndice A: Código principal del robot).

Con el módulo de movimiento del robot terminado se procedió a la implementación del algoritmo para la obtención y comparación de las imágenes, esto se realizó con un método de sustracción de fondo utilizando la biblioteca de código abierto OpenCV.

Figura 5: Imagen base

Figura 6: Imagen con detección de movimiento

Específicamente para el proceso de comparación de imágenes, primero se analizó si son del mismo tamaño como una medida adicional de control, seguido de una comparación píxel por píxel (ver Apéndice A: Código para comparar imágenes). Al terminar la comparación, en caso de que haya alguna diferencia entre las imágenes se procederá a subir la imagen a la página de Cloudinary; de lo contrario, se descartará dicha imagen.

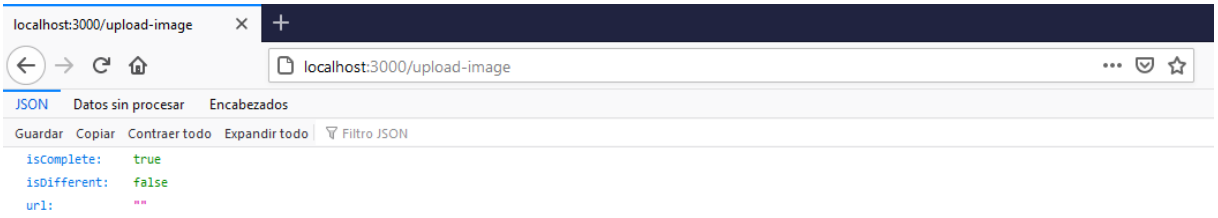

Figura 7: Api call de imágenes sin alerta

Al terminar la comparación, si es que se detectó alguna diferencia, se realiza la notificación a través de un correo electrónico enviado a la dirección proporcionada previamente por el encargado de la seguridad del CPD (ver Apéndice A: Código principal para el procesamiento, comparación de imágenes y notificación).

Resultados

A continuación, se muestra el api call enviada cuando se detecta movimiento o alteraciones relevantes en las imágenes comparadas.

Figura 8: Api call de imágenes con alerta

Después de realizarse el api call se envia el correo notificando al encargado del CPD de la detección de movimiento.

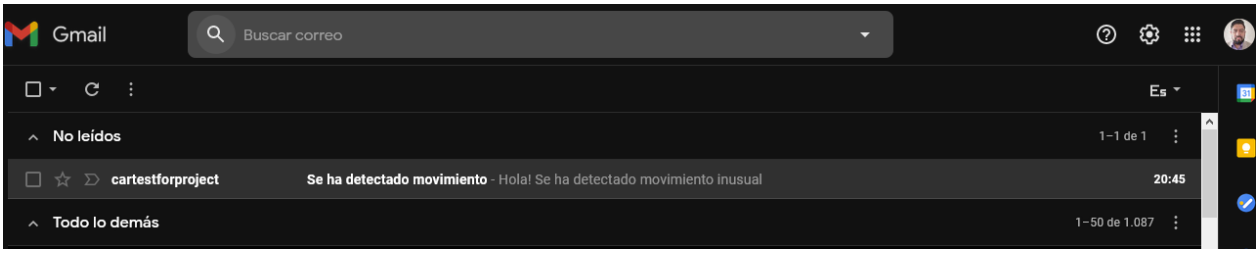

Figura 9: Correo notificando una alerta de movimiento

Para finalizar se muestra el contenido del correo con la imagen en la cual se detecto movimiento para que sea vista por el encargado del CPD y poder identificar a la persona que ingreso o el motivo de la notificación.

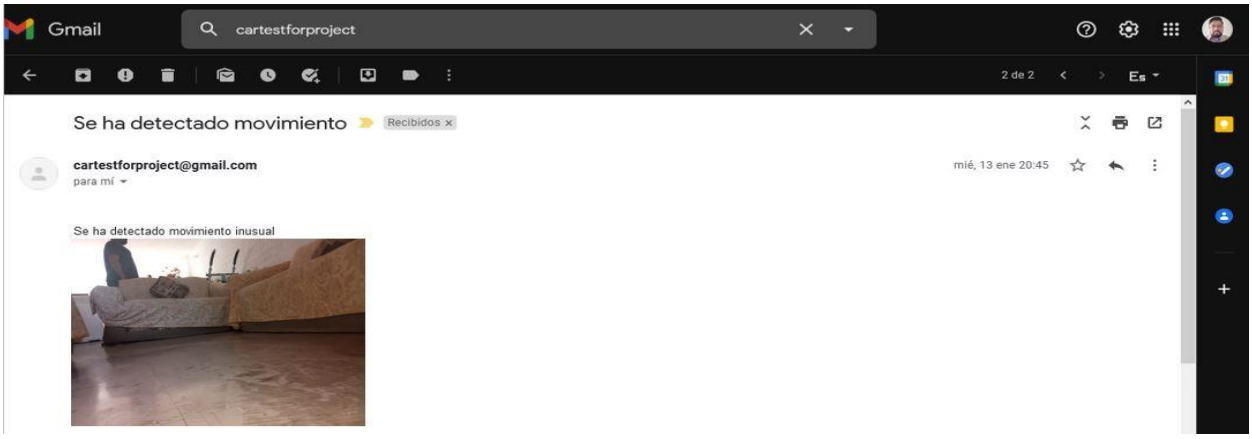

Figura 10: Contenido del correo de notificación

Análisis y discusión de resultados

El proyecto se realizó inicialmente como estaba planeado, se ensamblo el robot con todos los elementos y se procedió a programar todo lo referente al movimiento de este además del control de la cámara. Sin embargo, al llegar al módulo de procesamiento de la imagen para poder detectar movimiento o alteraciones, se presentaron diversos obstáculos no previstos y se tuvieron que tomar medidas para solucionarlos.

El principal problema era que se notificaban alteraciones por cambios apenas perceptibles, tanto en ángulo como en posición o hasta en la iluminación de la imagen. Para solucionar el problema de la variación en ángulo al capturar la imagen, se opto por un chasis para 4 ruedas, dándole una mayor estabilidad. Con respecto a la posición en que era tomada la fotografía, se recurrió al uso de un sensor para seguimiento de línea, así se mantenía un circuito estable y se podían marcar los lugares exactos donde detenerse para capturar la imagen.

Finalmente, se había considerado hacer un servidor web e interfaz propios, pero al realizar una investigación para elaborar ese módulo, se encontró como servicios ya existentes podían complementar la implementación de este. Por un lado, Cloudinary facilita el manejo de los archivos e imágenes, mientras que al usar un servicio de correo para la notificación permite tanto la flexibilidad de usarlo tanto en PC con un navegador común, como en dispositivos móviles a través de su respectiva app. Otro beneficio de ambos recursos, que es la principal razón de su uso, es su calidad de uptime o tiempo en línea, el cual es de 99% o mayor.

Conclusiones

Las diferentes opciones para obtener seguridad privada a través de medios digitales presentan diversas fallas que no siempre permiten su utilización para todas las actividades donde se pudieran necesitar. Tenemos problemas con la calidad de la imagen, los puntos ciegos o las fallas propias de un sistema que esta a cargo de un tercero, es cierto que poco a poco la tecnología mejora tanto en su calidad como en su precio, pero muchos de estos sistemas no son tan accesibles en su costo.

Es aquí donde radica la fortaleza del proyecto realizado, se elimina el problema de los puntos ciegos, el costo de elaboración es menor y al usar servicios con un uptime mayor al 99% se logra reducir, y casi eliminar, los problemas de notificación y la transmisión adecuada de la información, además de que al usar el servicio de correo común te permite acceder a otros recursos de las diferentes plataformas, por ejemplo respaldar la imagen capturada donde hay detección de movimiento en un Drive para que no este almacenada únicamente como archivo adjunto en el correo. Todos estos cambios y variables, que inicialmente se consideraron alteraciones de los objetivos iniciales, terminaron siendo mejoras que le dieron mas seguridad y flexibilidad en diferentes aspectos.

Debido a limitaciones propias del tiempo del proyecto y las circunstancias fue necesario limitar la optimización del proyecto, pero para apegarse mas a los objetivos iniciales se podría implementar un chasis de 4 ruedas recortado y con una rueda loca para darle estabilidad. Otra posible mejora futura sería el uso de una celda de energía fija recargable para sustituir las pilas recargables. Finalmente se podría considerar el uso de una mejor cámara conforme mejore la tecnología y se reduzcan los costos.

Apéndice A

Código principal del robot

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <wiringPi.h>
#include <softPwm.h>
#include <netdb.h>
#include <netinet/in.h>
#include <strings.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include "motordriver.h"
#include "remoteserver.h"
#include <pthread.h>
#include <sys/time.h>

#include <signal.h>
#include <stdarg.h>
#include <getopt.h>


#include "clk.h"
#include "gpio.h"
#include "dma.h"
#include "pwm.h"
#include "ws2811.h"

int baseSpeed, addLeftSpeed, addRightSpeed;
unsigned long getColour = 0xFF0000;
unsigned int getBrightness = 100;
static int speedVal_1 = 5000;
static int speedVal_2 = 5000;
static int speedVal_3 = 5000;
static int speedVal_4 = 5000;


struct motionstate carstate = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
static unsigned char disWarning  = 0;
```

```
static unsigned char poweroffFlag = 0;
static unsigned char turnLeftFlag = 0;
static unsigned char turnBackFlag = 0;
static unsigned char servoBeep = 0;
unsigned char buf[IR_LIMITS];  // bytes buffer
unsigned int bits;  // 32768 bits capable
unsigned char done;
int sockfd, newsockfd, portno, clilen;
unsigned char client_Connected = 0;
unsigned char count;

#define BLOCK_SIZE (4*1024)

#define ARRAY_SIZE(stuff)       (sizeof(stuff) / sizeof(stuff[0]))

// defaults for cmdline options
#define TARGET_FREQ             WS2811_TARGET_FREQ
#define GPIO_PIN                18
#define DMA                     10
#define STRIP_TYPE            WS2811_STRIP_RGB    // WS2812/SK6812RGB integrated
chip+leds
//#define STRIP_TYPE              WS2811_STRIP_GBR    // WS2812/SK6812RGB integra
ted chip+leds
//#define STRIP_TYPE            SK6812_STRIP_RGBW   // SK6812RGBW (NOT SK6812RGB)
#define WIDTH                   4
#define HEIGHT                  4
#define LED_COUNT               (WIDTH * HEIGHT)

void  INThandler(int sig);
void exit_UCTRONICS_Robot_Car(void);
int  mem_fd;
void *gpio_map;
// I/O access
volatile unsigned *gpio;
int width = WIDTH;
int height = HEIGHT;
int led_count = LED_COUNT;
int clear_on_exit = 0;

const rpi_hw_t *my_rpi_hw;

ws2811_t ledstring =
{
  .freq = TARGET_FREQ,
  .dmanum = DMA,
```

```
  .channel =
  {
    [0] =
    {
      .gpionum = GPIO_PIN,
      .count = LED_COUNT,
      .invert = 0,
      .brightness = 100,
      .strip_type = STRIP_TYPE,
    },
    [1] =
    {
      .gpionum = 0,
      .count = 0,
      .invert = 0,
      .brightness = 0,
    },
  },
};

ws2811_led_t *matrix;

unsigned long  grb_colour_table[] =
{
  0xFF0000,  // green
  0x00FF00,  // red
  0x0000FF,  // blue
  0xFFFF00,  //yellow
  0xFF00FF,
  0x00FFFF,  //pink
  0xFFFFFF,  // white
  //0x000000,
};
unsigned long  receive_colour_table[4] =
{
  0xFF0000,  // green
  0x00FF00,  // red
  0x0000FF,  // blue
  0xFFFF00,  //yellow
};



int PhaseScratchCmd(char command);
```

```
/* Creates a server socket and listens for a command from the remote.
*/
int main(int argc, char *argv[])
{
  usleep(10);
  char buffer[BUFFER_SIZE];
  struct sockaddr_in serv_addr, cli_addr;
    struct sigaction sa;
  int  n, pulsenum, count ;
  int on = 1;
  static unsigned long previous_time = 0;
  static unsigned long now_time = 0;
  static unsigned long time_stamp = 0;
  static int getColIndex = 0;
  static  unsigned char readNowTime = 1;
  signal(SIGINT, INThandler);
  /* Initialise GPIO */
  if (ControllerInit() < 0) return -1;
  setup_io();
  ultraInit();
  servoInit();
  trackModeInit();
  beepInit();
  irInit();
  myPWMInit();
  GRBInit();
  pthread_t t1, t2;
  //creat two thread
  pthread_create(&t1, NULL, fun1, NULL);
  pthread_create(&t2, NULL, fun2, NULL);
  for (pulsenum = 0; pulsenum < 10; pulsenum++) {
    servoCtrl(servo_1, 1140);
    servoCtrl(servo_2, 630);
  }
  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if (sockfd < 0) {
    perror("ERROR opening socket");
    exit(1);
  }
  /* Initialize socket structure */
  bzero((char *) &serv_addr, sizeof(serv_addr));
  portno = 2001;
  serv_addr.sin_family = AF_INET;
```

```
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(portno);

/*Add support port reuse*/
  if ((setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on))) < 0)
  {
    perror("setsockopt failed");
    exit(EXIT_FAILURE);
  }
  /* Now bind the host address using bind() call.*/
  if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
    perror("ERROR on binding");
    exit(1);
  }

  /* Add support reconnection */
sa.sa_handler = SIG_IGN;
sigaction( SIGPIPE, &sa, 0 );
/* Now start listening for the clients, here process will
  go in sleep mode and will wait for the incoming connection
*/
while (1) {
  listen(sockfd, 5);
  clilen = sizeof(cli_addr);
  printf("Waitting connection...\r\n");
  client_Connected = 0;
  clearFlag();
  stop();
  BEEP_OPEN();
  /* Accept actual connection from the client */
  newsockfd = accept(sockfd, (struct sockaddr *)&cli_addr, (void *) &clilen);
  if (newsockfd < 0) {
    perror("ERROR on accept");
    //exit(1);
  }
  /* If connection is established then start communicating */
  printf("connect successfully\r\n");

  sleep(0.5);
  GRB_work(3, getColour, getBrightness);
  n = write(newsockfd, "{\"version\":2}", 13);
  client_Connected = 1;
  sleep(0.001);
  bzero(&buffer, BUFFER_SIZE);
  while ((n = read(newsockfd, &buffer, BUFFER_SIZE)) > 0)
```

```
{
  if (buffer[0] == 's') { //0x73
    baseSpeed = buffer[1];
    addLeftSpeed = buffer[2];
    addRightSpeed = buffer[3];
    speedVal_1 = 10000 * ((buffer[1] + buffer[2]) / 255.0);
    speedVal_3 = 10000 * ((buffer[1] + buffer[2]) / 255.0);

    speedVal_2 = 10000 * ((buffer[1] + buffer[3]) / 255.0);
    speedVal_4 = 10000 * ((buffer[1] + buffer[3]) / 255.0);

  } else if (buffer[0] == 'c') { //0x63
    getColour = (buffer[1] << 16) | (buffer[2] << 8) | buffer[3];
    getBrightness = buffer[4];
    GRB_work(3, getColour, getBrightness);
if(readNowTime){
  readNowTime = 0;
  previous_time = get_pwm_timestamp();
}
now_time = get_pwm_timestamp();
  time_stamp = now_time - previous_time;
  if (time_stamp <2000000) {
   printf("set current direction\r\n");
  receive_colour_table[getColIndex] = getColour;
  }else{
  readNowTime = 1;
  printf("set next direction\r\n");
  getColIndex = (1+getColIndex)%4;
  receive_colour_table[getColIndex] = getColour;
  previous_time = get_pwm_timestamp();
}

  } else if (buffer[0] == 'v') {
    printf("Reveive value %d\n", buffer[0]);
    write(newsockfd, "{\"version\":2}", 13);
  }
  else {
  for(count = 0; count <n; count ++){
  printf("receive data %d\r\n",buffer[count]);
}
  if(buffer[0]==0xFF && buffer[1] == 0x55){
    for (count = 2; count < n; count ++) {
       PhaseScratchCmd(buffer[count]);
      }
}else{
```

```
      for (count = 0; count < n; count ++) {
            updateCarState(buffer[count]);
            updateCarMotion();
          }
    }
      }
      bzero(&buffer, BUFFER_SIZE);
    }
    stop();
    clearFlag();
    if (n < 0) {
      printf("Connection exception!\n");
      //break;
    }
  }
  client_Connected = 0;
  close(sockfd);
  printf("ERROR\r\n");
  return 0;
}

void *fun1(void *arg) {
  unsigned char cnt = 0;
  float dis = 0, temp_min = 0, temp_max = 0, temp_value = 0, temp = 0;
  while (1) {
    usleep(1);
    temp_min  = temp_max;
    temp_value = temp_max;
    for (cnt = 0; cnt < 3; cnt ++ ) {
      temp = disMeasure(); usleep(1);
      if (temp_max < temp) temp_max = temp;
      if (temp_min > temp) temp_min = temp;
      temp_value += temp;
    }
    dis = (temp_value - temp_max - temp_min);
    temp_value = 0; temp_min = 0; temp_max = 0; temp_value = 0; temp = 0;
    if (dis > 0 && dis <= 50) {
      disWarning = 1;
      if (carstate.forward) {
        if (!(carstate.autoAvoid)) {
          stop();
        }

      }
    } else disWarning = 0;
```

```
    if (!poweroffFlag) {
      beepWarning();
    }
  if(servoBeep){
    servoBeep = 0;
    digitalWrite(BEEP, HIGH);
    delay(100);
    digitalWrite(BEEP, LOW);
  }


  }
}

void *fun2(void *arg) {
  int IRVal = 0;
  while (1) {
    usleep(1);
    if (carstate.trackenable) {
      printf("Come in track mode \n");
      trackModeWork();
    }
    if (carstate.autoAvoid) {
      avoidance();
    }
    if (done == 1) {
      done = 0;
      IRVal = buf[2]; IRVal = IRVal << 8;
      IRVal = IRVal | buf[3];
      IR_updateCarState(IRVal);
      IR_updateCarMotion();
      IRVal = 0;
    }
    mySoftPwmWrite1(speedVal_1);
    mySoftPwmWrite2(speedVal_2);
    mySoftPwmWrite3(speedVal_3);
    mySoftPwmWrite4(speedVal_4);
    if (!poweroffFlag)  getLedSta();
  }

}
```

```
int updateCarMotion(void) {
  static int angleA = 1140;
  static int angleB = 1140;
  char direction[16];
  if (carstate.forward && !carstate.back) {
    if ((!carstate.left && !carstate.right) ||
        (carstate.left && carstate.right)) {
      strcpy(direction, "forward");
      go_forward();
    } else if (carstate.left && !carstate.right) {
      strcpy(direction, "forward left");
      go_forward_left();
    } else if (carstate.right && !carstate.left) {
      strcpy(direction, "forward right");
      go_forward_right();
    } else {
      strcpy(direction, "stop");
      stop();
    }
  }
  else if (carstate.back && !carstate.forward) {
    if ((!carstate.left && !carstate.right) ||
        (carstate.left && carstate.right)) {
      strcpy(direction, "back");
      go_back();

    } else if (carstate.left && !carstate.right) {
      strcpy(direction, "back left");
      go_back_left();
    } else if (carstate.right && !carstate.left) {
      strcpy(direction, "back right");
      go_back_right();
    } else {
      strcpy(direction, "stop");
      stop();
    }
  } else if (carstate.left && !carstate.right) {
    strcpy(direction, "left");
    go_left();
  } else if (carstate.right && !carstate.left) {
    strcpy(direction, "right");
    go_right();
  } else if( carstate. stop){
    carstate. stop = 0;
    stop();
```

```
}


if (carstate.servoLeft) {
  carstate.servoLeft = 0;
  strcpy(direction, "servo_left");
  if (angleA < 2000)
    angleA = angleA + 50;
  else{
    angleA = 2000;servoBeep =1;
    //BEEP_OPEN();
}
servoCtrl(servo_1,  angleA);
  printf("angleA = %d\r\n", angleA);
}
if (carstate.servoRight) {
  carstate.servoRight = 0;
  strcpy(direction, "servo_right");
  if (angleA > 600)
    angleA = angleA - 50;
  else{
angleA = 600;servoBeep =1;
}
 servoCtrl(servo_1,  angleA);
  printf("angleA = %d\r\n", angleA);
}
if (carstate.servoUp) {
  carstate.servoUp = 0;
  strcpy(direction, "servo_UP");
  if (angleB > 600)
    angleB = angleB - 50;
  else{
   angleB = 600;servoBeep =1;
}
servoCtrl(servo_2,  angleB);

  printf("angleB = %d\r\n", angleB);
} else if (carstate.servoDown) {
  carstate.servoDown = 0;
  strcpy(direction, "servo_down");
  if (angleB < 2000)
    angleB = angleB + 50;
  else{
   angleB = 2000;servoBeep =1;
}
```

```
      servoCtrl(servo_2,  angleB);
    printf("angleB = %d\r\n", angleB);
  }
  printf("Motion: %s \n", direction);
  return 0;
}
void  clearFlag(void) {
  carstate.left = 0;
  carstate.forward = 0;
  carstate.right = 0;
  carstate. stop = 0;
  carstate. back = 0;
  carstate.servoLeft = 0;
  carstate. servoRight = 0;
  carstate. servoUp = 0;
  carstate. servoDown = 0;
  carstate. speedUp = 0;
  carstate. speedDown = 0;
  carstate. autoAvoid = 0;

}

int IR_updateCarMotion(void) {
  static int angleA = 1140;
  static int angleB = 630;
  char direction[16];
  if (carstate.forward) {
    strcpy(direction, "forward");
    go_forward(); printf("Motion: %s \n", direction);
  }
  if (carstate.back) {

    strcpy(direction, "back");
    go_back(); printf("Motion: %s \n", direction);

  }
  if (carstate.left) {
    strcpy(direction, "left");
    go_left(); printf("Motion: %s \n", direction);
  }

  if (carstate.right) {
    strcpy(direction, "right");
    go_right(); printf("Motion: %s \n", direction);
  }
```

```
  if (carstate.servoLeft) {
    carstate.servoLeft = 0;
    strcpy(direction, "servo_left");
    if (angleA < 2300)
      angleA = angleA + 50;
    else
      angleA = 2300;
    servoCtrl(servo_1,  angleA);
    printf("angleA = %d\r\n", angleA);
  }
  if (carstate.servoRight) {
    carstate.servoRight = 0;
    strcpy(direction, "servo_right");
    if (angleA > 300)
      angleA = angleA - 50;
    else
      angleA = 300;
    servoCtrl(servo_1,  angleA);
    printf("angleA = %d\r\n", angleA);
  }
  if (carstate.servoUp) {
    carstate.servoUp = 0;
    strcpy(direction, "servo_UP");
    if (angleB > 300)
      angleB = angleB - 50;
    else
      angleB = 300;
    servoCtrl(servo_2,  angleB);
    printf("angleB = %d\r\n", angleB);
  }
  if (carstate.servoDown) {
    carstate.servoDown = 0;
    strcpy(direction, "servo_down");
    if (angleB < 1160)
      angleB = angleB + 50;
    else
      angleB = 1160;
    servoCtrl(servo_2,  angleB);

    printf("angleB = %d\r\n", angleB);
  }
  return 0;
}

int IR_updateCarState(int command) {
```

```
if ( command == IR_up || command == IR_up_v2) {
  if (!disWarning)
    carstate.forward = 1;
  else
    carstate.forward = 0;
}
if ( command == IR_down || command == IR_down_v2) {
  carstate.back = 1;
}
if ( command == IR_Left || command == IR_Left_v2) {
  carstate.left = 1;
}
if ( command == IR_right || command == IR_right_v2) {
  carstate.right = 1;
}
if ( command == IR_stop || command == IR_stop_v2) {
  carstate.forward = 0;
  carstate.back = 0;
  carstate.right = 0;
  carstate.left = 0;
  stop(); printf("Motion: stop \n" );
}
if ( command == IR_speed_up) {
  carstate.speedUp = 1;
}
if ( command == IR_speed_down) {
  carstate.speedDown = 1;
}
if ( command == IR_servo_up) {
  carstate.servoUp = 1;
}
if ( command == IR_servo_down) {
  carstate.servoDown = 1;
}
if ( command == IR_servo_left) {
  carstate.servoLeft = 1;
}
if ( command == IR_servo_right) {
  carstate.servoRight = 1;
}
if ( command == IR_track) {
  carstate.trackenable = 1;
}
if ( command == IR_track_stop) {
```

```
    carstate.trackenable = 0;
    stop();
    printf("Motion: stop \n" );
  }
  return 0;
}


int PhaseScratchCmd(char command){
  static int angleA = 1140;
  static int angleB = 630;

  switch (command) {
    case 1: // go forward
      go_forward();
      GRB_work(3, receive_colour_table[2], getBrightness);
      break;
    case 2: //go backward
      if (!disWarning) {
        go_back();
        GRB_work(3, receive_colour_table[0], getBrightness);
      }
    else
    stop();
    carstate.trackenable = 0;
    carstate.autoAvoid = 0;
      break;
    case 3: //go left
      go_left();
      GRB_work(3, receive_colour_table[3], getBrightness);
      carstate.trackenable = 0;
      carstate.autoAvoid = 0;
      break;
    case 4: //go right
      go_right();
      GRB_work(3, receive_colour_table[1], getBrightness);
      carstate.trackenable = 0;
      carstate.autoAvoid = 0;
      break;
    case 5: //stop
      stop();
      carstate.trackenable = 0;
      carstate.autoAvoid = 0;
      break;
    case 7: /* servo left */
```

```
      if (angleA < 2300)
       angleA = angleA + 50;
      else
        angleA = 2300;
      servoCtrl(servo_1,  angleA);
    break;
   case 8: /*servo right */
     if (angleA > 300)
     angleA = angleA - 50;
    else
     angleA = 300;
    servoCtrl(servo_1,  angleA);
    break;
   case 9: /* servo up */
     if (angleB > 300)
       angleB = angleB - 50;
     else
       angleB = 300;
     servoCtrl(servo_2,  angleB);
   break;
   case 10: /* servo down */
    if (angleB < 1160)
       angleB = angleB + 50;
    else
     angleB = 1160;
    servoCtrl(servo_2,  angleB);
   break;
   case 11: /* enable track */
     carstate.trackenable = 1;
     break;
   case 12: /* disable track */
     carstate.trackenable = 0;
     break;
   case 13:
     carstate.speedUp = 1;
     break;
   case 14:
     carstate.speedDown = 1;
     break;
   case 15: /* disable track */
     digitalWrite(BEEP, HIGH);
     break;
   case 16: /* disable track */
     digitalWrite(BEEP, LOW);
     break;
```

```
    case 17: /*automatic avoidance*/
      carstate.autoAvoid = 1;
      break;
    case 18: /*stop automatic avoidance*/
      carstate.autoAvoid = 0;
      break;
    case 19: /*turn off the robot car*/
      poweroffFlag = 1;
      exit_UCTRONICS_Robot_Car();
      printf("power off\n");
      system("sudo poweroff");
      break;
  }
  return 0;
}

/* Updates the struct MotionState of the car.
*/
int updateCarState(char command) {
  switch (command) {
    case 0: /* left */
      carstate.left = 1;
      GRB_work(3, receive_colour_table[2], getBrightness);
      break;
    case 1: /* up */
      if (!disWarning) {
        carstate.forward = 1;
        GRB_work(3, receive_colour_table[0], getBrightness);
      }
    else
      carstate.forward = 0;
    carstate.trackenable = 0;
    carstate.autoAvoid = 0;
      break;
    case 2: /* right */
      carstate.right = 1;
      GRB_work(3, receive_colour_table[3], getBrightness);
      carstate.trackenable = 0;
      carstate.autoAvoid = 0;
      break;
    case 3: /* down */
      carstate.back = 1;
      GRB_work(3, receive_colour_table[1], getBrightness);
      carstate.trackenable = 0;
      carstate.autoAvoid = 0;
```

```
    break;
  case 4: /* stop left */
    carstate.left = 0;
    carstate.trackenable = 0;
    carstate.autoAvoid = 0;
    break;
  case 5: /* stop*/
    carstate.forward = 0;
    carstate.back = 0;
  carstate.left = 0;
  carstate.right = 0;
    carstate.trackenable = 0;
    carstate.autoAvoid = 0;
  carstate. stop = 1;
    break;
  case 6: /* stop right */
    carstate.right = 0;
    carstate.trackenable = 0;
    carstate.autoAvoid = 0;
    break;
  case 7: /* servo left */
    carstate.servoLeft = 1;
    break;
  case 8: /*servo right */
    carstate.servoRight = 1;
    break;
  case 9: /* servo up */
    carstate.servoUp = 1;
    break;
  case 10: /* servo down */
    carstate.servoDown = 1;
    break;
  case 11: /* enable track */
    carstate.trackenable = 1;
    break;
  case 12: /* disable track */
    carstate.trackenable = 0;
    break;
  case 13: /* enable track */
    carstate.speedUp = 1;
    break;
  case 14: /* disable track */
    carstate.speedDown = 1;
    break;
  case 15: /* disable track */
```

```
      //  carstate.beepenable = 1;
      digitalWrite(BEEP, HIGH);
      break;
    case 16: /* disable track */
      //  carstate.beepenable = 0;
      digitalWrite(BEEP, LOW);
      break;
    case 17: /*automatic avoidance*/
      carstate.autoAvoid = 1;
      break;
    case 18: /*stop automatic avoidance*/
      carstate.autoAvoid = 0;
      break;
    case 19: /*turn off the robot car*/
      poweroffFlag = 1;
      exit_UCTRONICS_Robot_Car();
      printf("power off\n");
      system("sudo poweroff");

      break;
  }
  return 0;
}
void ultraInit(void)
{
  pinMode(Echo, INPUT);
  pinMode(Trig, OUTPUT);
}

float disMeasure(void)
{
  struct timeval tv1;
  struct timeval tv2;
  long start, stop;
  float dis;
  long waitCount = 0;

  digitalWrite(Trig, LOW);
  delayMicroseconds(2);

  digitalWrite(Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);
  waitCount = 0;
  while (!(digitalRead(Echo) == 1)) {
```

```
    if (++waitCount >= 5000)
      break;
    else sleep(0.001);
  }
  gettimeofday(&tv1, NULL);
  waitCount = 0;
  while (!(digitalRead(Echo) == 0)) {
    if (++waitCount >= 5000)
      break;
    else sleep(0.001);
  }
  gettimeofday(&tv2, NULL);
  /*
    int gettimeofday(struct timeval *tv, struct timezone *tz);
    The functions gettimeofday() and settimeofday() can get and set the time as w
ell as a timezone.
    The use of the timezone structure is obsolete; the tz argument should normall
y be specified as NULL.
  */
  start = tv1.tv_sec * 1000000 + tv1.tv_usec;
  stop  = tv2.tv_sec * 1000000 + tv2.tv_usec;

  dis = (float)(stop - start) / 1000000 * 34000 / 2;

  return dis;
}

void trackModeInit() {
  INP_GPIO(leftSensor);
  GPIO_PULL = 1 << leftSensor;
  INP_GPIO(middleSensor);
  GPIO_PULL = 1 << middleSensor;
  INP_GPIO(rightSensor);
  GPIO_PULL = 1 << rightSensor;


}
void beepInit() {
  pinMode(BEEP, OUTPUT);
  digitalWrite(BEEP, LOW);
}
void getLedSta() {

  static unsigned long previous_time = 0;
  static unsigned long now_time = 0;
```

```
  static unsigned long time_stamp = 0;
  static unsigned char flag = 0;
  static unsigned char colour_index = 0;
  if (!disWarning) {

    if (!client_Connected) {
      if (!flag) {
        flag = 1;
        previous_time = get_pwm_timestamp();
      }
      now_time = get_pwm_timestamp();
      time_stamp = now_time - previous_time;
      if (time_stamp > 500000) {
        time_stamp = 0;
        flag = 0;
        colour_index = (1 + colour_index) % 7;
        //GRB_work(3, grb_colour_table[colour_index], 50);
        GRB_MultiColour_work(3, 100 );
      }
    }
  }


}
void beepWarning() {
  static unsigned long previous_time = 0;
  static unsigned long now_time = 0;
  static unsigned long time_stamp = 0;
  static unsigned char flag = 0;
  static unsigned char canStopBeep = 0;
  if (disWarning) {
    canStopBeep = 1;
    if (!flag) {
      flag = 1;
      previous_time = get_pwm_timestamp();
    }
    now_time = get_pwm_timestamp();
    time_stamp = now_time - previous_time;
    if (time_stamp > 0 && time_stamp <= 50000 ) { //1/2T
      digitalWrite(BEEP, HIGH);
      GRB_work(3, 0, 0);
    }
    if (time_stamp > 50000 && time_stamp <= 2 * 50000 ) { //1T
      digitalWrite(BEEP, LOW);
      GRB_work(3, getColour, getBrightness);
```

```
    }
    if (time_stamp > 2 * 50000) {
      flag = 0;
    }
  } else if (canStopBeep) {
    canStopBeep = 0;
    digitalWrite(BEEP, LOW);
    GRB_work(3, getColour, getBrightness);
  }
}

void trackModeWork() {
  int num1 = 0, num2 = 0, num3 = 0;
  while (1) {
    if (!(carstate.trackenable))break;

    mySoftPwmWrite1(speedVal_1);
    mySoftPwmWrite2(speedVal_2);
    mySoftPwmWrite3(speedVal_3);
    mySoftPwmWrite4(speedVal_4);


    num1 = GET_GPIO(leftSensor);
    num2 = GET_GPIO(middleSensor);
    num3 = GET_GPIO(rightSensor);
    if ((num2 == 0) && (num1 == 0) && (num3 == 0)) {
      stop(); continue;
    } else if ( (num1 == 0) && num3) { //go to right
      GRB_work(3, grb_colour_table[0], 100 ) ;
      go_forward_left();
      while (1) {
      GRB_work(3, grb_colour_table[0], 100 ) ;
        num2 = GET_GPIO(middleSensor);
        mySoftPwmWrite1(speedVal_1);
        mySoftPwmWrite2(speedVal_2);
        mySoftPwmWrite3(speedVal_3);
        mySoftPwmWrite4(speedVal_4);
        if (num2) {
          if (!(carstate.trackenable))break;
          if (disWarning) {
            stop();
          }
          else {
        GRB_work(3, grb_colour_table[1], 100 ) ;
            go_forward_left();
```

```
          }
          continue;
        } else
          break;
      }
    } else if ((num3 == 0) && num1) { // go to left
      go_forward_right();
      while (1) {
        num2 = GET_GPIO(middleSensor);
        mySoftPwmWrite1(speedVal_1);
        mySoftPwmWrite2(speedVal_2);
        mySoftPwmWrite3(speedVal_3);
        mySoftPwmWrite4(speedVal_4);
        if (num2) {
          if (!(carstate.trackenable))break;
          if (disWarning) {
            stop();
          }
          else {
        GRB_work(3, grb_colour_table[2], 100 ) ;
            go_forward_right();
          }
          continue;
        } else
          break;
      }
    } else if (disWarning) {
      stop();
    }
    else {
      go_forward();
    }

  }

}

unsigned char countLow(void)
{
  unsigned char i = 0;
  while ( digitalRead(IRIN) == 0 ); // wait
  while ( digitalRead(IRIN) == 1 ) {
    ++i;
    delayMicroseconds(26);
    if (i == 0) return 0; // timeout
```

```
  }
  return i;
}

void getIR() {
  unsigned char i;
  unsigned short j;   // capable 32768 bits = 4096 bytes
  unsigned char k;
  bits = 0;
  i = countLow();
  for (j = 0; j < IR_LIMITS; j++) { // buffer bytes LIMITS
    for (i = 0; i < 8; i++) { // 8 bits
      k = countLow();
      if (k == 0) {
        buf[j] >>= (8 - i);
        done = 1; printf("  \n");
        return;
      }
      buf[j] >>= 1;
      buf[j] += ((k > 30) ? 0x80 : 0);
      ++bits;
    }
  }
  done = 1;
}
void turn() {
  static unsigned long previous_time = 0;
  static unsigned long now_time = 0;
  static unsigned long time_stamp = 0;
  static unsigned char flag = 0;
  if (!flag) {
    flag = 1;
    previous_time = get_pwm_timestamp();
  }
  now_time = get_pwm_timestamp();
  time_stamp = now_time - previous_time;
  if (time_stamp > 0 && time_stamp <= turnTime ) { //1/2T
    go_back();turnBackFlag = 1;
  }
  if (time_stamp > turnTime && time_stamp <= 2 * turnTime ) { //1T
    go_left();turnLeftFlag = 1;
  }
  if (time_stamp > 2 * turnTime) {
    stop(); flag = 0;turnLeftFlag = 0;turnBackFlag = 0;
  }
```

```
}

void avoidance(void)
{

  while (1) {
    if (!(carstate.autoAvoid)) {
      stop();
      break;
    }
    mySoftPwmWrite1(speedVal_1);
    mySoftPwmWrite2(speedVal_2);
    mySoftPwmWrite3(speedVal_3);
    mySoftPwmWrite4(speedVal_4);
    if (disWarning || turnLeftFlag||turnBackFlag) {
      turn();
    } else {
      printf("Go forward\n");
      go_forward();
    }
  }
}
void BEEP_INT (void) {
  digitalWrite(BEEP, HIGH);
}

void BEEP_OPEN (void) {
  digitalWrite(BEEP, HIGH);
  delay(500);
  digitalWrite(BEEP, LOW);
}

// Set up a memory regions to access GPIO
void setup_io()
{
 my_rpi_hw = rpi_hw_detect();

  /* open /dev/mem */
  if ((mem_fd = open("/dev/mem", O_RDWR | O_SYNC) ) < 0) {
    printf("can't open /dev/mem \n");
    exit(-1);
  }
  /* mmap GPIO */
  gpio_map = mmap(
              NULL,             //Any adddress in our space will do
```

```
              BLOCK_SIZE,       //Map length
              PROT_READ | PROT_WRITE, // Enable reading & writting to mapped mem
ory
              MAP_SHARED,       //Shared with other processes
              mem_fd,           //File to map
              my_rpi_hw-
>periph_base + 0x200000//GPIO_BASE         //Offset to GPIO peripheral
            );

  close(mem_fd); //No need to keep mem_fd open after mmap

  if (gpio_map == MAP_FAILED) {
    printf("mmap error %d\n", (int)gpio_map);//errno also set!
    exit(-1);
  }
  // Always use volatile pointer!
  gpio = (volatile unsigned *)gpio_map;
} // setup_io


uint64_t get_pwm_timestamp()
{
  struct timespec t;
  if (clock_gettime(CLOCK_MONOTONIC_RAW, &t) != 0) {
    return 0;
  }
  return (uint64_t) t.tv_sec * 1000000 + t.tv_nsec / 1000;
}

void mySoftPwmWrite1( int value)
{
  static int previous_time = 0;
  static int now_time = 0;
  static int time_stamp = 0;
  static  unsigned char flag1 = 0;

  if (!flag1) {
    flag1 = 1;
    previous_time = get_pwm_timestamp();
  }
  now_time = get_pwm_timestamp();
  time_stamp = now_time - previous_time;
  if (time_stamp > 0 && time_stamp <= halfPWMPeriod ) { //1/2T
    if (time_stamp <= value ) {
      GPIO_SET = 1 << MOTOR_1_PWM;
```

```
    }
    else {
      GPIO_CLR = 1 << MOTOR_1_PWM;
    }
  }
  if (time_stamp > halfPWMPeriod && time_stamp <= 2 * halfPWMPeriod ) { //1T
    if (time_stamp <= value ) {
      GPIO_SET = 1 << MOTOR_1_PWM;
    }
    else {
      GPIO_CLR = 1 << MOTOR_1_PWM;
    }
  }
  if (time_stamp > 2 * halfPWMPeriod) {
    flag1 = 0;
  }
}
void mySoftPwmWrite2( int value)
{
  static int previous_time = 0;
  static int now_time = 0;
  static int time_stamp = 0;
  static  unsigned char flag2 = 0;
  if (!flag2) {
    flag2 = 1;
    previous_time = get_pwm_timestamp();
  }
  now_time = get_pwm_timestamp();
  time_stamp = now_time - previous_time;
  if (time_stamp > 0 && time_stamp <= halfPWMPeriod ) { //1/2T
    if (time_stamp <= value ) {
      GPIO_SET = 1 << MOTOR_2_PWM;
    }
    else {
      GPIO_CLR = 1 << MOTOR_2_PWM;
    }
  }
  if (time_stamp > halfPWMPeriod && time_stamp <= 2 * halfPWMPeriod ) { //1T
    if (time_stamp <= value ) {
      GPIO_SET = 1 << MOTOR_2_PWM;
    }
    else {
      GPIO_CLR = 1 << MOTOR_2_PWM;
    }
  }
```

```
  if (time_stamp > 2 * halfPWMPeriod) {
    flag2 = 0;
  }
}
void mySoftPwmWrite3( int value)
{
  static int previous_time = 0;
  static int now_time = 0;
  static int time_stamp = 0;
  static  unsigned char flag3 = 0;
  if (!flag3) {
    flag3 = 1;
    previous_time = get_pwm_timestamp();
  }
  now_time = get_pwm_timestamp();
  time_stamp = now_time - previous_time;
  if (time_stamp > 0 && time_stamp <= halfPWMPeriod ) { //1/2T
    if (time_stamp <= value ) {
      GPIO_SET = 1 << MOTOR_3_PWM;
    }
    else {
      GPIO_CLR = 1 << MOTOR_3_PWM;
    }
  }
  if (time_stamp > halfPWMPeriod && time_stamp <= 2 * halfPWMPeriod ) { //1T
    if (time_stamp <= value ) {
      GPIO_SET = 1 << MOTOR_3_PWM;
    }
    else {
      GPIO_CLR = 1 << MOTOR_3_PWM;
    }
  }
  if (time_stamp > 2 * halfPWMPeriod) {
    flag3 = 0;
  }
}
void mySoftPwmWrite4( int value)
{
  static int previous_time = 0;
  static int now_time = 0;
  static int time_stamp = 0;
  static  unsigned char flag4 = 0;

  if (!flag4) {
    flag4 = 1;
```

```
    previous_time = get_pwm_timestamp();
  }
  now_time = get_pwm_timestamp();
  time_stamp = now_time - previous_time;
  if (time_stamp > 0 && time_stamp <= halfPWMPeriod ) { //1/2T
    if (time_stamp <= value ) {
      GPIO_SET = 1 << MOTOR_4_PWM;
    }
    else {
      GPIO_CLR = 1 << MOTOR_4_PWM;
    }
  }
  if (time_stamp > halfPWMPeriod && time_stamp <= 2 * halfPWMPeriod ) { //1T
    if (time_stamp <= value ) {
      GPIO_SET = 1 << MOTOR_4_PWM;
    }
    else {
      GPIO_CLR = 1 << MOTOR_4_PWM;
    }
  }
  if (time_stamp > 2 * halfPWMPeriod) {
    flag4 = 0;
  }
}

void servoInit(void)
{
  INP_GPIO(servo_1); // must use INP_GPIO before we can use OUT_GPIO
  OUT_GPIO(servo_1);
  INP_GPIO(servo_2); // must use INP_GPIO before we can use OUT_GPIO
  OUT_GPIO(servo_2);
}

void servoCtrl(int servoNum, int dutyCycle)
{
  int count =0;
  for(count = 0; count < 3; count ++){
    GPIO_SET = 1 << servoNum;
    delayMicroseconds(dutyCycle);
    GPIO_CLR = 1 << servoNum;
    delayMicroseconds(25000 - dutyCycle);
  }
}
```

```
void servoAControl( int value)
{
  static int previous_time = 0;
  static int now_time = 0;
  static int time_stamp = 0;
  static  unsigned char flag1 = 0;

  if (!flag1) {
    flag1 = 1;
    previous_time = get_pwm_timestamp();
  }
  now_time = get_pwm_timestamp();
  time_stamp = now_time - previous_time;
  if (time_stamp > 0 && time_stamp <= 10000 ) { //1/2T
    if (time_stamp <= value ) {
      GPIO_SET = 1 << servo_1;
    }
    else {
      GPIO_CLR = 1 << servo_1;
    }
  }
  if (time_stamp > 10000 && time_stamp <= 2 * 10000 ) { //1T
    if (time_stamp <= value ) {
      GPIO_SET = 1 << servo_1;
    }
    else {
      GPIO_CLR = 1 << servo_1;
    }
  }
  if (time_stamp > 2 * 10000) {
    flag1 = 0;
  }
}




void GRB_work(unsigned int ledNum, unsigned long colour, int brightness ) {
  int i;
  ledstring.channel[0].brightness = brightness;
  for (i = 0; i < ledNum; i++) {
    ledstring.channel[0].leds[i] = colour;
```

```
  }
  ws2811_render(&ledstring) ;

}

void GRB_MultiColour_work(unsigned int ledNum,  int brightness ) {
  int i;
  static int colour_index = 0;

  ledstring.channel[0].brightness = brightness;
  for (i = 0; i < ledNum; i++) {
    colour_index = (1 + colour_index) % 7;
    ledstring.channel[0].leds[i] = grb_colour_table[colour_index];
  }
  ws2811_render(&ledstring) ;

}


void irInit() {
  pinMode(IRIN, INPUT);
  pullUpDnControl(IRIN, PUD_UP);
  // to detect falling edge
  wiringPiISR(IRIN, INT_EDGE_FALLING, (void*)getIR);
  done = 0;
}
void myPWMInit() {
  // Switch GPIO 7..11 to output mode
  INP_GPIO(MOTOR_1_PWM); // must use INP_GPIO before we can use OUT_GPIO
  OUT_GPIO(MOTOR_1_PWM);
  INP_GPIO(MOTOR_2_PWM); // must use INP_GPIO before we can use OUT_GPIO
  OUT_GPIO(MOTOR_2_PWM);
  INP_GPIO(MOTOR_3_PWM); // must use INP_GPIO before we can use OUT_GPIO
  OUT_GPIO(MOTOR_3_PWM);
  INP_GPIO(MOTOR_4_PWM); // must use INP_GPIO before we can use OUT_GPIO
  OUT_GPIO(MOTOR_4_PWM);
}
void GRBInit() {
  ws2811_return_t ret;
  matrix = malloc(sizeof(ws2811_led_t) * width * height);
  if ((ret = ws2811_init(&ledstring)) != WS2811_SUCCESS)
  {
    fprintf(stderr, "ws2811_init failed: %s\n", ws2811_get_return_t_str(ret));
  }
```

```
}

void exit_UCTRONICS_Robot_Car(void)
{
  int pulsenum;
  client_Connected = 0;
  close(sockfd);
  for (pulsenum = 0; pulsenum < 10; pulsenum++) {
    servoCtrl(servo_1, 1300);
    servoCtrl(servo_2, 1300);
  }
  digitalWrite(BEEP, LOW);
  GRB_work(3, 0, 0 );
}

void  INThandler(int sig)
{
  char  c;
  signal(sig, SIG_IGN);
  printf("You hit Ctrl-C\n"
         "Do you really want to quit? [y/n] ");
  c = getchar();
  if (c == 'y' || c == 'Y') {
    exit_UCTRONICS_Robot_Car();
    printf("Bye.\n");
    exit(0);
  }

  else
    signal(SIGINT, INThandler);
  getchar(); // Get new line character
}
```

Código principal para el procesamiento, comparación de imágenes y notificación

```
import os
from dotenv import load_dotenv
from flask import Flask, jsonify, request
import cv2
import json
import numpy as np
```

```
import urllib.request as urllib
import requests
from flask_cors import CORS
import smtplib

from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

#cloudinary
from cloudinary.api import delete_resources_by_tag, resources_by_tag, delete_reso
urces
from cloudinary.uploader import upload
from cloudinary.utils import cloudinary_url
import cloudinary

app = Flask(__name__)
CORS(app)
project_folder = os.path.expanduser('~/images-compare')  # adjust as appropriate
load_dotenv(os.path.join(project_folder, '.env'))

API_KEY = os.getenv('API_KEY')
API_SECRET = os.getenv('API_SECRET')
EMAIL_PASSWORD = os.getenv('EMAIL_PASSWORD')

# Email config
me = "cartestforproject@gmail.com"
## a donde se mandan
you = "cesar.angeles.barrios@gmail.com"

msg = MIMEMultipart('alternative')
msg['Subject'] = "Se ha detectado movimiento"
msg['From'] = me
msg['To'] = you

#CLOUDINARY function handler
cloudinary.config(
cloud_name = "choskas",
api_key = API_KEY,
api_secret = API_SECRET
)


def dump_response(response):
    print("Upload response:")
    for key in sorted(response.keys()):
```

```
        print("  %s: %s" % (key, response[key]))

#API QUE CHECA LA PRIMERA IMAGEN
@app.route('/upload-image')
def upload_files():
    print("--- Upload a local file")
    isDifferent = False
    ## foto que se tomo recientemente
    response = upload("img/escala.jpg", tags="")
    dump_response(response)
    ## se obtiene la imagen
    url, options = cloudinary_url(
        response['public_id'],
        format=response['format'],
    )
    public_id = response['public_id']
    ## abrir la imagen
    url_response = urllib.urlopen(url)
    img_array = np.array(bytearray(url_response.read()), dtype=np.uint8)
    duplicate = cv2.imdecode(img_array, -1)
    ## url de la imagen original
    originalUrl = "https://res.cloudinary.com/choskas/image/upload/v1607385168/cg
9lnefzs5wbdslubtzz.jpg"
    url_response2 = urllib.urlopen(originalUrl)
    img_array2 = np.array(bytearray(url_response2.read()), dtype=np.uint8)
    original = cv2.imdecode(img_array2, -1)
    print(duplicate, original)
    # duplicate = cv2.imread(img)
    image1 = original.shape
    image2 = duplicate.shape
    print(image1, image2)
    if original.shape == duplicate.shape:
        difference = cv2.subtract(original, duplicate)
        b, g, r = cv2.split(difference)
        print(b, g, r)
        if cv2.countNonZero(b) == 0 and cv2.countNonZero(g) == 0 and cv2.countNon
Zero(r) == 0:
            print("Las imagenes son iguales")
            delete_resources(public_id)
            url = ""
        else:
            ## correo en html
            html = f"""\
                    <html>
                    <head></head>
```

```
                    <body>
                    <p>Hola!<br>
                    Se ha detectado movimiento inusual<br>
                    <img width=300 height=200 src={url}> </img>
                    </p>
                    </body>
                    </html>
                    """
            ## funciones para mandar emails
            part2 = MIMEText(html, 'html')
            msg.attach(part2)
            s = smtplib.SMTP('smtp.gmail.com', 587)
            s.starttls()
            s.login('cartestforproject@gmail.com', EMAIL_PASSWORD)
            s.sendmail(me, you, msg.as_string())
            s.quit()
            print("las imagenes son diferentes")
            print("mostando la imagen diferente ->")
            isDifferent = True
    return jsonify({"isComplete": True, "url": url, "isDifferent" : isDifferent})
#####################################################



if __name__ == '__main__':
    app.run(debug=True, port=3000)
```

Código para comparar imágenes

```
import cv2
import numpy as np

original = cv2.imread("original.jpg")
duplicate = cv2.imread("escala.jpg")

# Checa si las imagenes son iguales en tamaño ->
image1 = original.shape
image2 = duplicate.shape

# compara pixel por pixel si el tamaño es igual ->
if original.shape == duplicate.shape:
    print("Las imagenes tienen el mismo tamaño")
    difference = cv2.subtract(original, duplicate)
    b, g, r = cv2.split(difference)
```

```
    print(cv2.countNonZero(b))
    if cv2.countNonZero(b) == 0 and cv2.countNonZero(g) == 0 and cv2.countNonZero
(r) == 0:
        print("Las imagenes son iguales")
    else:
        print("las imagenes son diferentes")
        print("mostando la imagen diferente ->")
        cv2.imshow("Imagen diferente", duplicate)

    cv2.imshow("difference", difference)

cv2.imshow("Original", original)
cv2.imshow("duplicate", duplicate)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Referencias

[1] V. Chang and M. Ramachandran, "Towards Achieving Data Security with the Cloud Computing Adoption Framework", IEEE Transactions on Services Computing, vol. 9, no. 1, pp. 138-151, Jan.-Feb. 2016.

[2] Y. Zhang, L. Zhang and W. Wang, "Design and Implementation of a Two-Wheel and Hopping Robot With a Linkage Mechanism", IEEE Access, Vol. 6, pp. 42422-42430, August 2018.

[3] L. A. Orozco, G. García, "Robot para obtención de imágenes de objetos en movimiento", Proyecto Terminal de Ingeniería en Electrónica y Comunicaciones, UAM-Iztapalapa, disponible en: http://tesiuami.izt.uam.mx/uam/aspuam/presentatesis.php?recno=13966&docs=UAMI13966.PDF (2007).

[4] V. Ruíz, A. M. García, "Localización y mapeo simultaneo de un robot", Proyecto Terminal de la Licenciatura en Computación, UAM-Iztapalapa, disponible en: http://tesiuami.izt.uam.mx/uam/aspuam/presentatesis.php?recno=16755&docs=UAMI16755.pdf (2014).

[5] R. Rivera, "Robot barredor autónomo", Proyecto Terminal de Ingeniería Electrónica, UAM-Iztapalapa, disponible en: http://tesiuami.izt.uam.mx/uam/aspuam/presentatesis.php?recno=11208&docs=UAMI11208.PDF (2003)

[6] J.L. Azuara, "Navegación de un robot móvil para la supervisión de una residencia monitoreada vía Internet", Proyecto Terminal de Ingeniería Mecatrónica, UNAM Facultad de Ingeniería, disponible en: http://132.248.9.195/ptd2018/mayo/0774776/Index.html (2018).

[7] E.G. Arteaga, "Sistema cliente servidor para visión de un robot móvil usando una Wireless LAN", Proyecto Terminal de Ingeniería Electrónica, PUCP Facultad de Ciencias e Ingeniería, disponible en: http://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/984/ARTEAGA_OSORIO_EDWARD_ROBOT_MOVIL_WIRELESS_LAN.pdf?sequence=1&isAllowed=y (2006)

[8] K. D. Do, "Bounded Assignment Formation Control of Second-Order Dynamic Agents", IEEE/ASME Transactions on Mechatronics, Vol. 19, Issue: 2, pp. 477-489, February 2013.

[9] Z. Lye, H. Nisar and K. Lai, "Localization and feature recognition: Implementation on an indoor navigator robot", 10th France-Japan/ 8th Europe-Asia Congress on Mecatronics (MECATRONICS2014), January 2015.

[10] F. Santos de Oliveira and A. M. Dias, "Development of a Navigator with Artificial Intelligence Applied to Smoothing and Execution of Paths in a Dynamic Two-Dimensional Environment", IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), April 2019.

[11] E. Zahir, S. A. Shihab and S. M. Akash, "Introducing a Voice Synthesized Visual and Auditory Navigator Implemented for a University Campus", IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), Dec. 2018.

[12] A. Mancini, E. Frontoni and P. Zingaretti, "Embedded Multisensor System for Safe Point-to-Point Navigation of Impaired Users", IEEE Transactions on Intelligent Transportation Systems, Vol. 16, Issue: 6, pp. 3543-3555, November 2015.

[13] C. F. Juang, M. G. Lai and W. T. Zeng, "Evolutionary Fuzzy Control and Navigation for Two Wheeled Robots Cooperatively Carrying an Object in Unknown Environments", IEEE Transactions on Cybernetics, Vol. 45, Issue: 9, pp. 1731-1743, November 2014.

[14] B. Mu, J. Chen and Y. Shi, "Design and Implementation of Nonuniform Sampling Cooperative Control on A Group of Two-Wheeled Mobile Robots", IEEE Transactions on Industrial Electronics, Vol. 64, Issue: 6, pp. 5035-5044, December 2016.

[15] D. Chwa, "Robust Distance-Based Tracking Control of Wheeled Mobile Robots Using Vision Sensors in the Presence of Kinematic Disturbances", IEEE Transactions on Industrial Electronics, Vol. 63, Issue: 10, pp. 6172-6183, July 2016.

[16] W. Ye, Z. Li and C. Yang, "Vision-Based Human Tracking Control of a Wheeled Inverted Pendulum Robot", IEEE Transactions on Cybernetics, Vol. 46, Issue: 11, pp. 2423-2434, October 2015.

[17] Digilent, «Arty Z7: Zynq-7000 SoC Development Board,» [En línea]. Available: https://store.digilentinc.com/arty-z7-zynq-7000-soc-development-board/. [Último acceso: 15 Febrero 2021].

[18] «¿Qué es Python? - Curso de iniciación a la programación con Python en Raspberry Pi,» [En línea]. Available: https://www.programoergosum.com/cursos-online/raspberry-pi/244-iniciacion-a-python-en-raspberry-pi/que-es-python. [Último acceso: 15 Febrero 2021].

[19] OpenCV, «OpenCV: Introduction,» [En línea]. Available: https://docs.opencv.org/master/d1/dfb/intro.html. [Último acceso: 15 Febrero 2021].

[20] Cloudinary, «About | Cloudinary,» 2021. [En línea]. Available: https://cloudinary.com/about. [Último acceso: 25 Febrero 2021].