

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE TECNOLOGÍA
CARRERA ELECTRÓNICA Y TELECOMUNICACIONES



**“SISTEMA DE CONTROL BASADO EN LA DETECCIÓN
DE GESTOS DE LOS DEDOS”**

Trabajo de Aplicación – Examen de Grado presentado para obtener el Grado de Licenciatura

POR: WILSON CONDORI APAZA

LA PAZ – BOLIVIA

MAYO, 2023

DEDICATORIA

A mi Esposa, Nelly Amelia Mamani Espejo, quien siempre me brindo todo el apoyo que pudo en todo momento en especial en esta última etapa, quien me ayudo a crecer como persona y profesional.

A mis hijos Wilhelm y William por ser el pilar de mi vida y alegrarme todos los días de mi vida por estar conmigo y verme crecer como profesional.

A mis padres, Vicente condori y Genoveva Apaza por haberme apoyado y guiado durante toda mi vida.

AGRADECIMIENTO

A Dios, por darme un día más de vida y permitirme realizar este trabajo.

A la Facultad de Tecnología por todos brindarme un espacio de estudio.

A los docentes de la Carrera de Electrónica y Telecomunicaciones, por todo el conocimiento brindado y por la ayuda en solventar dudas como profesional.

A la directora de carrera Lic. Julia Torrez Soria por siempre estar dispuesta a orientar ante cualquier duda que se tenía sobre cualquier tema que fuera.

RESUMEN	7
CAPITULO I	1
PLANTEAMIENTO DEL PROBLEMA	1
1.1. PLANTEAMIENTO DEL PROBLEMA	1
1.1.1. FORMULACION DEL PROBLEMA	2
1.2. JUSTIFICACION DEL TRABAJO.....	2
1.1.2. Justificación tecnológica	3
1.1.3. Justificación social.....	3
1.1.4. Justificación económica	3
1.1.5. Justificación académica.....	3
1.3. OBJETIVOS.....	4
1.3.1. OBJETIVO GENERAL.....	4
1.3.2. OBJETIVOS ESPECIFICOS.....	4
CAPITULO II	5
FUNDAMENTO TEORICO	5
2.1. DISCAPACIDAD	5
2.1.1. Tipos de Discapacidad.....	5
2.1.2. Discapacidad y Autonomía.....	6
2.1.3. La Importancia de la Autonomía en la Calidad de Vida de las Personas con Discapacidad.	6
2.2. SISTEMAS DE CONTROL DE ILUMINACIÓN Y PUERTAS.....	7
a) Componentes y funcionamiento	7
b) Tecnologías utilizadas para el control de la iluminación y puertas en espacios Interiores y Exteriores.....	8
2.2.1. Sensores de iluminación	10
a) Sensor LDR	10
b) Sensor PIR	12
2.2.2. Relés.....	16
a) Principio de funcionamiento.....	16
c) Tipos de relés utilizados para el control de circuitos de iluminación y otros dispositivos eléctricos	19
2.2.3. Servomotores.....	20

a)	Principios de Funcionamiento.....	20
b)	Características del servomotor (Micro servo 9g – SG90).....	22
c)	Partes de un servomotor	23
d)	Aplicaciones en el Control de puertas, persianas y otros dispositivos	24
2.3.	VISIÓN ARTIFICIAL.....	25
2.3.1.	Detección de dedos levantados en una imagen	25
2.3.2.	Librerías y Herramientas para la Detección de Manos y Dedos en imágenes utilizando Visión Artificial	26
2.3.2.1.	OpenCV	26
a.	Comandos básicos de OpenCV.....	28
b.	Aplicaciones de OpenCV	28
2.3.2.2.	MediaPipe	30
a.	Comandos básicos de mediaPipe.....	31
b.	Modelos para el procesamiento de las coordenadas de los puntos de referencia de las manos.....	33
c.	Aplicaciones de mediaPipe	35
2.3.2.3.	Integrando OpenCV y mediaPipe.....	35
2.3.2.4.	Comunicación serial	36
a)	Configurar la Comunicación Serial de Python a ESP32	38
b)	Configurar la Comunicación Serial de ESP32 a Python	38
2.4.	ESP32.....	39
2.4.1.	Descripción del Microcontrolador ESP32	41
2.4.2.	Características.....	41
2.4.3.	Memoria.....	42
2.4.4.	Pines disponibles en el ESP32	43
2.4.4.1.	Pines de Entradas y salidas	44
2.4.4.2.	Pines UART	46
2.4.4.3.	ADC.....	47
2.4.4.4.	PWM.....	49
2.4.5.	IDE de Arduino y como Programar el ESP32	49
2.4.5.1.	Descargar e instalar ESP32 en Arduino	49
2.4.5.2.	Configurar ESP32 en Arduino IDE	50

a)	Conectamos ESP32 vía USB a la computadora	50
b)	Agregamos las librerías de ESP32	51
2.4.5.3.	Buscar el puerto de ESP32	53
2.4.5.4.	Cargar un código de ejemplo	53
2.4.5.5.	Estructura de un programa	53
c)	Función de configuración ()	54
d)	Función loop ()	54
2.4.6.	Versiones de ESP32	54
2.4.7.	Ventajas y Desventajas	55
2.4.7.1.	Ventajas del ESP32	55
2.4.7.2.	Desventajas del ESP32:	56
CAPITULO III	57
DESARROLLO DEL TRABAJO DE APLICACIÓN	57
3.1.	DESCRIPCION DEL TRABAJO	57
3.2.	DIAGRAMA DE BLOQUES	57
3.3.	REPRESENTACIÓN DE CIRCUITO.....	58
3.4.	PROGRAMACION DEL TRABAJO	59
3.4.1.	Arduino IDE	59
3.4.2.	Python	68
3.5.	COSTOS DEL TRABAJO Y DURACION	77
3.5.1.	Costo de Materiales	77
3.5.2.	Costo de Mano de Obra	77
3.5.3.	Tiempo Empleado	78
CAPITULO IV	79
CONCLUSIONES Y RECOMENDACIONES	79
4.1.	CONCLUSIONES	79
4.2.	RECOMENDACIONES	80
BIBLIOGRAFIA	82
ANEXOS	85

RESUMEN

En este trabajo se diseñó un sistema de control de iluminación y puertas basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32, con el objetivo de mejorar la autonomía de las personas con discapacidad.

En el marco teórico se revisaron los sistemas de control de iluminación y puertas existentes, así como las tecnologías de visión artificial y reconocimiento de gestos. Se realizó el diseño del sistema, que incluyó la selección y utilización de componentes electrónicos como relés, sensor LDR, PIR y servomotores.

En la implementación del sistema se utilizaron técnicas de programación en Python y se realizó la integración del software y hardware. Se realizaron pruebas y se evaluó el rendimiento del sistema, demostrando su efectividad en la detección de gestos y control de los dispositivos.

Como conclusiones generales se destaca la importancia de desarrollar soluciones tecnológicas que mejoren la calidad de vida de las personas con discapacidad, y la efectividad del sistema diseñado en el cumplimiento de este objetivo.

Entre las recomendaciones generales se destacan la importancia de tomar en cuenta los cuidados en programación en Python y la utilización adecuada de los componentes electrónicos como relés, LDR, PIR y servomotores para garantizar un funcionamiento correcto y seguro del sistema.

En resumen, el sistema de control de iluminación y puertas basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32 diseñado en este trabajo es una solución efectiva y accesible para mejorar la autonomía de las personas con discapacidad en su vida diaria.

CAPITULO I

PLANTEAMIENTO DEL PROBLEMA

1.1. PLANTEAMIENTO DEL PROBLEMA

El planteamiento del problema consiste en diseñar un sistema de control de iluminación y puertas que permita mejorar la autonomía de las personas con discapacidad. Actualmente, las personas con discapacidad se enfrentan a limitaciones en su capacidad para realizar actividades cotidianas, como encender y apagar la iluminación de un ambiente o abrir y cerrar puertas. Estas limitaciones pueden generar frustración e incomodidad, lo que afecta su calidad de vida y su independencia.

Actualmente, muchas personas con discapacidad enfrentan barreras físicas y tecnológicas que limitan su capacidad para controlar y operar los dispositivos en su entorno. Las soluciones actuales, como los interruptores de pared y los mandos a distancia, no son adecuados para las necesidades de estas personas, lo que limita su autonomía y capacidad para realizar tareas cotidianas

Aunque existen algunas tecnologías similares que se han desarrollado para mejorar la accesibilidad de las personas con discapacidad, estas tienen limitaciones y no son efectivas en todos los casos. Por lo tanto, es importante desarrollar una solución innovadora que permita a las personas con discapacidad tener mayor autonomía en su vida diaria y que sea fácil de implementar y utilizar.

Por lo tanto, se requiere un sistema de control que permita a las personas con discapacidad controlar sus dispositivos de forma fácil y accesible. El uso de la visión artificial y el reconocimiento de gestos específicos de las manos puede ser una solución efectiva para este problema. Este sistema puede ser diseñado para controlar diferentes dispositivos en un entorno doméstico, como iluminación, ventilación, puertas, entre otros.

El objetivo de este trabajo es diseñar un sistema de control de iluminación y puertas basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32, que permita a las

personas con discapacidad controlar estos dispositivos de forma autónoma y sin la necesidad de utilizar dispositivos tradicionales de control, como interruptores y cerraduras.

Este sistema permitirá a las personas con discapacidad tener un mayor grado de autonomía en su vida cotidiana, mejorando su calidad de vida y reduciendo la dependencia de terceros para realizar tareas básicas. Además, se espera que este sistema pueda ser implementado de manera económica y accesible para la mayoría de las personas con discapacidad.

1.1.1.FORMULACION DEL PROBLEMA

¿Cuáles son las características técnico-económico que tendrá el Diseño de un sistema Control de iluminación y puertas para personas con discapacidad, basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32?

1.2. JUSTIFICACION DEL TRABAJO

El objetivo de diseñar un sistema de control de iluminación y puertas basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32 es mejorar la calidad de vida de las personas con discapacidad, ofreciéndoles una mayor autonomía e independencia en su vida cotidiana.

Este sistema permitirá a las personas con discapacidad controlar los dispositivos de su entorno de manera autónoma, sin la necesidad de depender de terceros o de dispositivos tradicionales de control que pueden ser difíciles de manejar o inaccesibles para ellos.

Además, el sistema propuesto es innovador y utiliza tecnologías de última generación, como la visión artificial y el reconocimiento de gestos, para ofrecer una solución efectiva y de alta precisión para las personas con discapacidad.

Este sistema también es escalable y puede ser adaptado para controlar diferentes dispositivos en un entorno doméstico, lo que lo convierte en una solución completa y versátil para mejorar la calidad de vida de las personas con discapacidad.

En resumen, la justificación general del sistema de control de iluminación y puertas basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32 es mejorar la autonomía e independencia de las personas con discapacidad, ofreciéndoles una solución innovadora, accesible, económica y escalable para controlar los dispositivos en su entorno.

1.1.2. Justificación tecnológica

La visión artificial y el reconocimiento de gestos específicos de las manos pueden ser una solución efectiva para mejorar la accesibilidad y autonomía de las personas con discapacidad.

1.1.3. Justificación social

El diseño de un sistema de control de iluminación y puertas basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32 permitirá a las personas con discapacidad controlar sus dispositivos de forma autónoma, mejorando su calidad de vida y reduciendo la dependencia de terceros para realizar tareas básicas.

1.1.4. Justificación económica

El diseño de un sistema de control de iluminación y puertas basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32 plantea contemplar el uso de equipos tecnológicos y dispositivos existentes en el medio local. Buscando minimizar costos en su futura implementación. Se espera que este sistema pueda ser implementado de manera económica y accesible para la mayoría de las personas con discapacidad, lo que permitirá mejorar su autonomía sin incurrir en grandes costos.

1.1.5. Justificación académica

El diseño y desarrollo de este sistema representa una oportunidad para explorar nuevas tecnologías y aplicaciones en el campo de la accesibilidad y la tecnología asistida, contribuyendo al avance del conocimiento y la investigación en este campo.

1.3. OBJETIVOS

1.3.1.OBJETIVO GENERAL

Diseñar un prototipo de un sistema Control de iluminación y puertas, basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32, para mejorar la autonomía de las personas con discapacidad

1.3.2.OBJETIVOS ESPECIFICOS

Describir la situación actual de las necesidades y limitaciones de las personas con discapacidad en relación con el control de iluminación y puertas.

Describir las características técnicas que tendrá el Diseño del prototipo de un sistema Control de iluminación y puertas, basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32.

Implementar un modelo a escala del prototipo de un sistema Control de iluminación y puertas, basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32.

CAPITULO II

FUNDAMENTO TEORICO

2.1. DISCAPACIDAD

La discapacidad se define como una limitación en la capacidad de una persona para realizar actividades cotidianas debido a una condición física, mental o sensorial. Esta definición está respaldada por la Organización Mundial de la Salud (OMS) en su Clasificación Internacional del Funcionamiento, la Discapacidad y la Salud (CIF) (WHO, 2001).

El presente trabajo tiene como objetivo es el reducir esa limitación para que puedan realizar sus actividades con la mayor normalidad posible y con la mayor autonomía posible.

2.1.1. Tipos de Discapacidad

Existen diferentes tipos de discapacidad, que se definen según la causa y la naturaleza de la limitación. Según la OMS, los tipos de discapacidad más comunes son (WHO, 2011):

- **Discapacidad física:** se refiere a una limitación en la movilidad o la capacidad física de una persona debido a una enfermedad o lesión en el sistema nervioso, músculo-esquelético o sensorial.
- **Discapacidad mental:** se refiere a una limitación en las funciones mentales de una persona, incluyendo el aprendizaje, la memoria, el pensamiento, la percepción y la comunicación.
- **Discapacidad sensorial:** se refiere a una limitación en la capacidad de una persona para recibir o procesar información sensorial, como la vista, el oído o el tacto.
- **Discapacidad intelectual:** se refiere a una limitación en las habilidades cognitivas de una persona, incluyendo la capacidad de aprendizaje, razonamiento y resolución de problemas.

El presente trabajo pretende ayudar en el mayor porcentaje posible a las personas con discapacidad física y sensorial, como ya se mencionó lograr reducir sus limitaciones al realizar sus actividades cotidianas más sencillas como el control de su entorno.

2.1.2. Discapacidad y Autonomía

La discapacidad puede tener un impacto significativo en la autonomía de una persona. Según la Organización Mundial de la Salud (OMS), la discapacidad se define como "un término general que abarca las deficiencias, las limitaciones de la actividad y las restricciones de la participación" (OMS, 2011, p. 7). Esto significa que la discapacidad puede afectar tanto la capacidad de una persona para realizar actividades cotidianas como su capacidad para participar en la sociedad de manera plena y efectiva.

La autonomía, por otro lado, se refiere a la capacidad de una persona para tomar decisiones y controlar su vida de manera independiente. La discapacidad puede afectar la autonomía de una persona al limitar su capacidad para realizar ciertas tareas y actividades, y al hacer que dependa de otros para su cuidado y apoyo.

Por lo tanto, como se pretende reducir las limitaciones al realizar actividades cotidianas teniendo el control de su entorno, entonces le brindaremos a la persona con discapacidad el poder de tomar decisiones de controlar en encendido de luces y apertura de puertas, de la forma que pueda controlar los mismos con sus dedos.

2.1.3. La Importancia de la Autonomía en la Calidad de Vida de las Personas con Discapacidad.

La autonomía es un aspecto fundamental en la calidad de vida de las personas con discapacidad. La capacidad de una persona para tomar decisiones y controlar su vida de manera independiente puede tener un impacto significativo en su bienestar físico, emocional y social.

Cuando una persona con discapacidad tiene la capacidad de tomar decisiones y controlar su vida de manera independiente, puede tener un mayor control sobre su entorno y su vida diaria, lo que

puede mejorar su autoestima, autoeficacia y bienestar emocional. Además, una mayor autonomía también puede mejorar la participación de la persona en la comunidad y en la sociedad en general.

Por otro lado, la falta de autonomía puede llevar a una mayor dependencia de otras personas, lo que puede tener un impacto negativo en la calidad de vida de la persona. La falta de control sobre su vida puede provocar estrés, ansiedad y depresión, y también puede limitar su capacidad para participar en actividades cotidianas y sociales.

Por lo tanto, el trabajo tiene como objetivo otorgarles a las personas con discapacidad esa autonomía para vivir de manera independiente, entonces este trabajo pretende ser un apoyo para reducir la dependencia de otras personas y mejorar su autoestima, autoeficacia y bienestar emocional.

2.2. SISTEMAS DE CONTROL DE ILUMINACIÓN Y PUERTAS

Los sistemas de control de iluminación y puertas son tecnologías que "permiten la automatización de los sistemas de iluminación, puertas en edificios y espacios públicos para lograr una mayor eficiencia energética, seguridad y accesibilidad para personas con discapacidad" (Alonso, Rodríguez, & Martínez, 2019, p. 51).

En resumen, los sistemas de control de iluminación y puertas pueden mejorar la accesibilidad y la seguridad de los edificios para personas con discapacidad. Estos sistemas de control pueden incluir sensores de movimiento y luz

En el presente proyecto se diseña un circuito que le permita a la persona con discapacidad controlar la iluminación y puertas gracias a la automatización de los mismos solo utilizando señas específicas de los dedos.

a) Componentes y funcionamiento

Los componentes de un sistema de control de iluminación y puertas pueden incluir, según Zhou et al. (2020), los siguientes elementos:

- **Controladores:** dispositivos que se encargan de recibir las señales de los sensores y procesarlas para controlar los sistemas de iluminación y puertas.
- **Sensores:** dispositivos que detectan la presencia, el movimiento o la luminosidad en un espacio determinado y envían señales a los controladores para que ajusten el nivel de iluminación o activen las puertas.
- **Actuadores:** dispositivos que se encargan de ejecutar las órdenes emitidas por los controladores, como por ejemplo encender o apagar las luces, o abrir o cerrar las puertas.
- **Redes de comunicación:** sistemas que permiten la transmisión de señales entre los distintos elementos del sistema de control de iluminación y puertas, como los controladores, sensores y actuadores.

En nuestro trabajo los actuadores vendrían a ser los relés para controlar la iluminación y servomotores para el control de puertas, los sensores son el sensor de movimiento para detectar la presencia de personas y LDR como sensor de luz para detectar si es día o noche, y cámara de la computadora para detectar que dedo esta levantado y para que todo esto funcione de manera correcta se requiere de un controlador que en nuestro casos es el módulo ESP32 el cual contiene un microcontrolador el cual es programables que se pueden utilizar para controlar los relés y servomotores así como los sensores como sensores, actuadores y controladores. Para finalizar nuestra red de comunicación vendría a ser la comunicación serial utilizada entre la computadora y el módulo ESP32, funciona enviando y recibiendo datos en serie a través de un solo canal de comunicación, lo que hace que la transmisión sea más simple y económica en términos de cableado y conectividad

b) Tecnologías utilizadas para el control de la iluminación y puertas en espacios Interiores y Exteriores.

Las tecnologías utilizadas para el control de la iluminación y puertas en espacios interiores y exteriores pueden variar en función de la complejidad del sistema y el tipo de espacio en el que se están instalando. A continuación, se describen algunas de las tecnologías comúnmente utilizadas:

- **Sistemas de automatización del hogar:** Los sistemas de automatización del hogar pueden integrar el control de la iluminación y las puertas en un sistema único y fácil de usar. Los usuarios pueden controlar la iluminación y las puertas desde una aplicación móvil o un dispositivo inteligente.
- **Sensores de movimiento:** Los sensores de movimiento pueden utilizarse para controlar tanto la iluminación como las puertas en un espacio determinado. Los sensores de movimiento pueden activar la iluminación o abrir las puertas automáticamente cuando se detecta movimiento en un espacio.
- **Interruptores inteligentes:** Los interruptores inteligentes pueden controlar tanto la iluminación como las puertas y pueden integrarse con otros dispositivos en un sistema de automatización del hogar. Los interruptores inteligentes pueden controlarse a través de una aplicación móvil o un dispositivo inteligente.
- **Controladores programables:** Los controladores programables permiten programar diferentes niveles de iluminación y el estado de las puertas según la hora del día o la actividad que se está llevando a cabo en un espacio. Estos pueden ser programados para ajustarse automáticamente según las necesidades del usuario.

Según el libro "Smart Home Systems and Automation: IoT Enabled Design and Implementation" de Naveen Chilamkurti, en su 1ra edición del año 2020, página 84, se menciona que "Los sistemas de automatización del hogar son la tecnología más comúnmente utilizada para el control de la iluminación y las puertas en espacios interiores y exteriores. Estos sistemas permiten controlar la iluminación y las puertas desde una aplicación móvil o un dispositivo inteligente".

En nuestro caso el tipo de tecnología utilizada es los Sistemas de automatización del hogar, al cual se incorpora los sensores de movimiento y los LDR como sensores de luz y esto nos permitirá el control de la iluminación y la apertura de puertas controladas de forma remota a través de señas específicas de los dedos.

2.2.1.Sensores de iluminación

Según el libro "Lighting Design Basics" de Mark Karlen y Christina Spangler, en su 3ra edición del año 2017, página 141, se menciona que "Los sensores de movimiento y los sensores de luz ambiental son los tipos más comunes de sensores utilizados para el control de la iluminación en interiores y exteriores".

Los sensores de iluminación pueden ser dispositivos utilizados para detectar el movimiento, la luz y otras variables ambientales y, en consecuencia, controlar la iluminación. A continuación, se describen los tipos de sensores utilizados en el presente trabajo para el control de la iluminación.

a) Sensor LDR

Utilizado como Sensores de luz ambiental miden el nivel de luz en un espacio y ajustan la iluminación en consecuencia. Estos sensores pueden utilizarse para aumentar o disminuir la intensidad de la iluminación según las condiciones ambientales, pero en nuestro caso será utilizado para el encendido de la alarma de presencia de personas y encendido de la luz del patio a una determinada luz ambiental.

i. principio de funcionamiento del LDR

Según el libro "Electronic Principles" de Albert Malvino y David Bates, en su 8va edición del año 2007, página 838, se menciona que "Un LDR es un dispositivo semiconductor que tiene una resistencia eléctrica que varía según la cantidad de luz que incide sobre él. El principio básico de funcionamiento se basa en la propiedad de algunos materiales semiconductores, como el sulfuro de cadmio, de absorber fotones de luz y liberar electrones".

Un LDR (Resistencia Dependiente de la Luz, por sus siglas en inglés) funciona según el principio de que la resistencia eléctrica de un material semiconductor varía en función de la cantidad de luz que incide sobre él.



Figura 1: LDR

Fuente: <https://1.bp.blogspot.com/-igAJ1wrVzIE/XfJuyJvY8VI/AAAAAAAAANL4/0wEBUydDYOc38oKTc1uEGYdMxEYXqIKZ/ACLCBGAsYHQ/s320/ldr.jpg>

(ii) Aplicaciones en el control de iluminación

Por lo tanto, cuando se utiliza un LDR como sensor de iluminación, la cantidad de luz que incide sobre él determina su resistencia eléctrica y, por lo tanto, la corriente que fluye a través de él. Esta genera una caída de tensión la cual puede ser leída y digitalizada por el módulo ESP32 a través de sus pines analógicos y de esta manera lo utilizaremos para controlar la activación del sensor de movimiento y luz intermitente, la misma que se activara por la tarde antes de anochecer.

Según la página web de Diwo BQ (diwo.bq.com, 2023): Existen diferentes aplicaciones en las que utilizaremos el LDR (por ejemplo, junto con un transistor o relé para activar las luces nocturnas de la calle), pero la configuración básica siempre es la misma: conectado en serie con una resistencia y con la salida de tensión entre ambos.

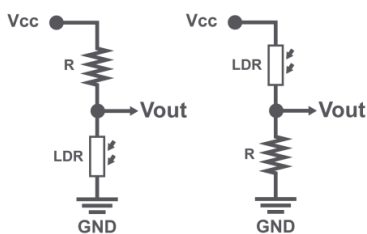


Figura 2: Conexión de LDR

Fuente: <http://diwo.bq.com/descubre-el-ldr/>

En el primer caso, la señal de salida V_{out} será 0 si incide luz, ya que la resistencia será muy baja y se “conectará” a tierra, mientras que si no incide la luz, la señal será alta.

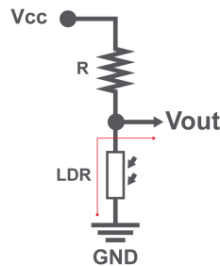


Figura3: LDR conectado a tierra

Fuente: <http://diwo.bq.com/descubre-el-ldr/>

En el segundo caso, la señal de salida V_{out} será igual a V_{cc} si incide luz, ya que la resistencia será muy baja y se “conectará” a la alimentación, mientras que si no incide la luz, la señal será baja.

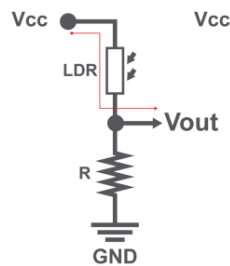


Figura4: LDR conectado a VCC

Fuente: <http://diwo.bq.com/descubre-el-ldr/>

b) Sensor PIR

Según Gómez-Expósito y Suárez-Álvarez (2018), un sensor PIR (del inglés Passive Infrared Sensor) es un tipo de sensor de movimiento que se basa en la detección de la radiación infrarroja emitida por los objetos en su campo de visión. Estos sensores se utilizan comúnmente en sistemas de seguridad y automatización del hogar.

Estos sensores de movimiento son dispositivos que detectan la presencia de personas o animales en un espacio y activan o desactivan la iluminación automáticamente. Estos sensores utilizan tecnologías como infrarrojos, ultrasonidos y microondas para detectar el movimiento en nuestro

caso se utilizará para detectar la proximidad de una persona a la puerta de la casa y que una persona esta de camino al cuarto de la persona con discapacidad para informarle mediante una luz intermitente y de esta manera la persona con discapacidad decidirá si abre la puerta con antelación o no la abre.

(i) principio de funcionamiento

De acuerdo con Rezazadeh et al. (2021), el funcionamiento de un sensor PIR se basa en la detección de cambios en el campo de radiación infrarroja presente en su entorno. El sensor consta de dos elementos: una lente que se encarga de enfocar la radiación infrarroja de la zona que se desea monitorizar y un detector que se encarga de medir la radiación infrarroja.

Cuando un objeto emite radiación infrarroja, esta es detectada por el sensor PIR y se produce una señal eléctrica que es amplificada y procesada para determinar si ha habido algún cambio en la radiación infrarroja en la zona de detección. Si se detecta un cambio significativo, se interpreta como la presencia de un objeto en movimiento y se envía una señal de activación para el sistema de seguridad o automatización correspondiente.

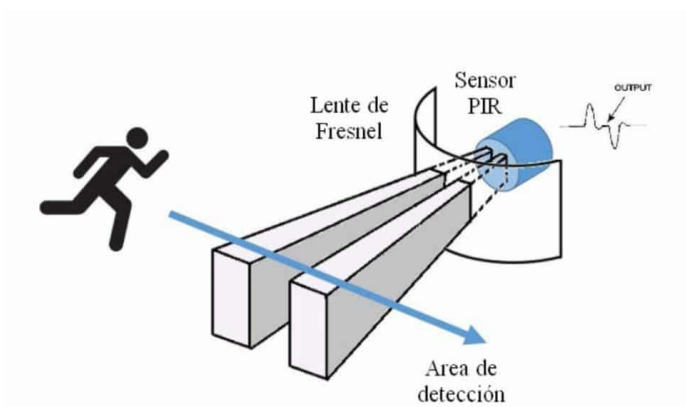


Figura5: sensor PIR

Fuente: <https://pasionelectronica.com/pir-sensor-de-movimiento/>

Out: Se pone a 3.3v cuando detecta, esta tensión es suficiente para activar una entrada digital de Arduino ya que a partir de 2,5 voltios lo interpreta como un 1 lógico.

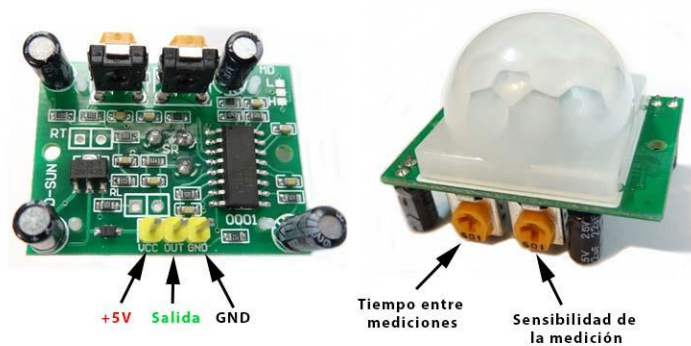


Figura6: PINES de sensor PIR

Fuente: https://www.zonamaker.com/images/contenido/arduino/modulos_sensores_shields/PIR/Sensor_PIR.jpg

Gnd: Ground masa 0 voltios

Hay otro modelo equivalente de este módulo que es el HC-SR501 que es exactamente igual salvo que en los pads nombrados en la imagen como L y H dispone de unos pines y un JUMPER o puente en el que podemos seleccionar dos modos de funcionamiento, según puenteemos entre el central y L o H.

En L pone la patilla OUT a nivel alto durante el tiempo programado y después se pone a nivel bajo haya detección o no.

En la posición H hace un re-disparo es decir mientras está detectando está reseteando el temporizador de apagado por lo tanto mientras se detecte movimiento estará OUT a nivel alto.

En el módulo que nos ocupa el pad central viene puenteado con H, por lo que mientras detecte movimiento estará activa la salida cuando deje de detectar contará el tiempo.

Si no hay movimiento no se dispara la salida OUT.

También hay que tener en cuenta que cuando la salida vuelve a su estado inactivo, hay un periodo de tres segundos que no detecta y que al alimentarlo está inicializando un tiempo inferior a 1 minuto

(ii) Características técnicas del Sensor de Movimiento:

- Usa el PIR LHI778 y el controlador BISS0001
- Voltaje de alimentación: de 5 a 12 VDC
- Consumo promedio: <1 miliamperio
- Rango de distancia de 3 a 7 metros ajustable.
- 5 HC-SR501 PIR Sensor infrarrojo de movimiento.
- Angulo de detección: cono de 110°
- Ajustes: 2 potenciómetros para ajuste de rango de detección y tiempo de alarma activa.
- No disponible Jumper para configurar la salida de alarma en modo mono-disparo ó disparo repetitivo ('retriggerable')
- Salida de alarma de movimiento con ajuste de tiempo entre 3 segundos a 5 minutos.
- Salida de alarma activa Vo con nivel alto de 3.3 volts y 5 mA source, lista para conexión de un led, ó un transistor y relé

ALGO A TENER EN CUENTA

- Tiempo de inicialización: después de alimentar el módulo HC-SR05, debe transcurrir 1 minuto antes de que inicie su operación normal. Durante ese tiempo, es posible que el módulo active 2 ó 3 veces su salida.
- Tiempo de salida inactiva: cada vez que la salida pase de activa a inactiva, permanecerá en ese estado los siguientes 3 segundos. Cualquier evento que ocurra durante ese lapso es ignorado. -Temperatura de operación: -15° a +70° C. -Dimensiones: 3.2 x 2.4 x 1.8 cm

(iii) aplicaciones en el control de iluminación y detección de movimiento

Según Chen et al. (2020), los sensores PIR se utilizan ampliamente en aplicaciones de control de iluminación y detección de movimiento en hogares, edificios comerciales y públicos. Algunas de las aplicaciones más comunes son:

- Control de iluminación: Los sensores PIR se utilizan en sistemas de iluminación automáticos para encender o apagar las luces en función de la presencia o ausencia de

personas en una habitación o área determinada. Esto ayuda a reducir el consumo de energía y aumentar la eficiencia energética.

- Detección de movimiento: Los sensores PIR se utilizan en sistemas de seguridad para detectar la presencia de personas en áreas restringidas o peligrosas. Por ejemplo, se pueden utilizar en sistemas de alarma para detectar intrusiones en hogares o negocios.
- Control de climatización: Los sensores PIR también se utilizan en sistemas de climatización para detectar la presencia de personas y ajustar la temperatura de la habitación en consecuencia. Esto ayuda a reducir el consumo de energía y aumentar la comodidad de los ocupantes.
- Sistemas de control de acceso: Los sensores PIR se pueden utilizar en sistemas de control de acceso para detectar la presencia de personas y permitir o denegar el acceso a áreas restringidas.

Después de todo lo expuesto el presente trabajo utilizará al sensor PIR como Control de iluminación y Sistemas de control de acceso el cual será controlado por la persona con discapacidad con señas específicas de los dedos.

2.2.2.Relés

Los relés son dispositivos electromecánicos que permiten la conexión o desconexión de circuitos eléctricos mediante el control de una señal eléctrica de entrada. El relé está compuesto por un conjunto de contactos que se abren o cierran dependiendo de la señal de entrada, y por una bobina que se encarga de activar dichos contactos (Hernández, 2013).

a) Principio de funcionamiento

El principio de funcionamiento de los relés se basa en el uso de una señal eléctrica de entrada para activar la bobina del relé. Al aplicar una corriente eléctrica a la bobina, se genera un campo magnético que atrae el conjunto de contactos del relé, permitiendo la conexión o desconexión de circuitos eléctricos (Hernández, 2013).

Según el sitio web Infootec (s.f.), Una característica importante de los relés es que la parte de emite la señal para activar el relé está aislada de la parte del relé que pone en marcha o enciende el receptor que en nuestro caso son los focos.

Uno de los extremos de la bombilla deberá estar conectado a uno de los contactos abiertos del relé, el otro al contacto abierto o común del relé estará conectado a tensión como se ve en el siguiente gráfico.

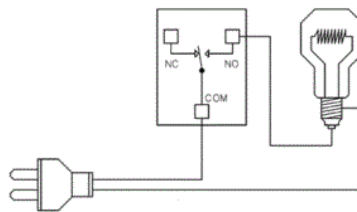


Figura7: Conexión de un RELE

Fuente: <https://www.infootec.net/rele-electromecanico/>

Ahora Cuando el contacto abierto del relé se cierre y pase la corriente a través de él, la bombilla recibirá tensión por los dos extremos y se encenderá como se ve en el siguiente gráfico.

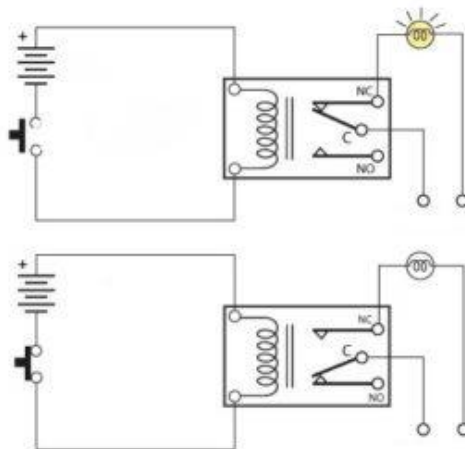


Figura8: Funcionamiento de un RELE

Fuente: <https://www.infootec.net/rele-electromecanico/>

Dato importante, según el gráfico anterior para que el contacto abierto del relé se cierre, deberemos activar la bobina del relé, por medio de los contactos de conexión de la bobina excitaremos la bobina con la tensión de trabajo que tenga (por ejemplo 5, 12 o 24 voltios en continua) y crearemos el campo magnético atrayendo el hierro inducido y cerrando los contactos conectados al receptor.

b) Partes de un relé electromagnético

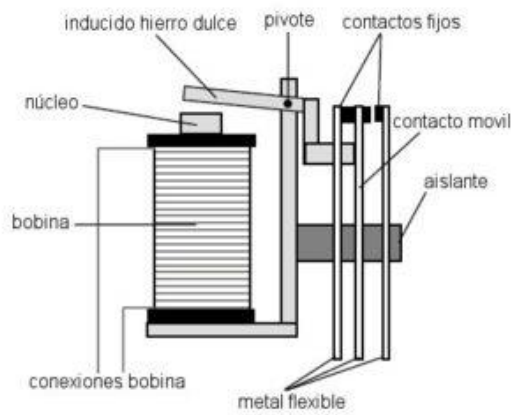


Figura9: Partes de un RELE

Fuente: <https://www.infootec.net/rele-electromecanico/>

- La bobina – La bobina de este relé es la encargada de generar una corriente inducida en el bobinado crear un campo magnético.
- Conexiones de la bobina – Mediante estas conexiones daremos tensión a la bobina, normalmente serán tensiones de 12 voltios o 24 voltios en corriente continua.
- Núcleo – El núcleo está situado en el interior de la bobina y se magnetiza con la intención de atraer la parte metálica llamada hierro inducido.
- Hierro inducido – El hierro inducido se moverá atraído por el núcleo y provocará la unión de los contactos abiertos.
- Contactos abiertos – Los contactos abiertos los utilizaremos para dar tensión al receptor que queramos hacer actuar.

c) **Tipos de relés utilizados para el control de circuitos de iluminación y otros dispositivos eléctricos**

Existen varios tipos de relés que se utilizan para el control de circuitos de iluminación y otros dispositivos eléctricos, como los relés electromecánicos, los relés de estado sólido y los relés fotoeléctricos. Los relés electromecánicos son los más comunes y utilizan un campo magnético para abrir y cerrar los contactos eléctricos. Los relés de estado sólido utilizan componentes electrónicos para controlar la conexión o desconexión de los circuitos eléctricos. Y los relés fotoeléctricos utilizan un haz de luz para activar o desactivar los contactos eléctricos (Machado, 2019).

En el presente trabajo se realizará la utilización de los relés electromecánicos los cuales serán controlados por señales específicas de los dedos estos relés en el mercado vienen en módulos, los cuales permitirán el control de varios receptores este módulo de relés se muestra en la siguiente imagen.

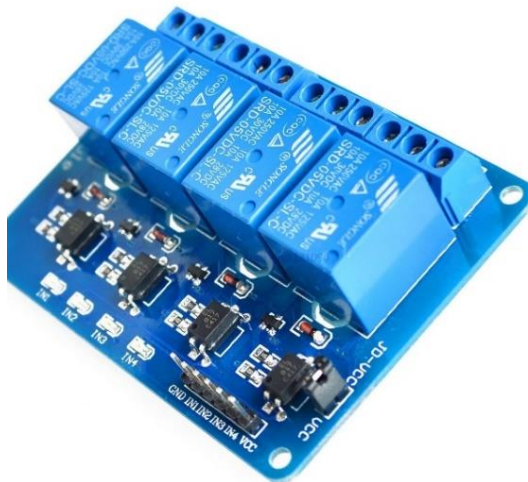


Figura10: Modulo de RELES

Fuente: <https://www.e-ika.com/modulo-de-reles-4-canales-para-arduino>

Su funcionamiento es simple podemos activar todo tipo de dispositivos y pequeños electrodomésticos (10A máximo por canal), lámparas, ventiladores, cafeteras, etc. Se alimenta con 5VDC y cada relé lo podemos activar mediante un pin digital del Arduino conectado a los terminales IN1, IN2, IN3 o IN4. Se activa a nivel bajo

2.2.3.Servomotores

Los servomotores son dispositivos de control de movimiento que se utilizan en una variedad de aplicaciones, desde robots hasta equipos industriales y de automatización. Estos motores proporcionan un control de posición y velocidad preciso, y su operación se basa en la retroalimentación de un sensor para controlar la posición del motor. Además, pueden ser programados para realizar movimientos precisos y repetitivos (Hughes, 2005).

a) Principios de Funcionamiento

Los servomotores poseen tres cables, a diferencia de los motores comunes que sólo tienen dos. Estos tres cables casi siempre tienen los mismos colores, por lo que son fácilmente reconocibles.




Voltaje positivo	Tierra (ground)	Señal de control
		

Figura11: PINES servomotor

Fuente: <https://panamahitek.com/que-es-y-como-funciona-un-servomotor/>

La necesidad de una señal de control para el funcionamiento de este tipo de motores hace que sea imposible utilizarlos sin un circuito de control adecuado. Esto se debe a que para que el circuito de control interno funcione, es necesaria una señal de control modulada. Para esto se utiliza modulación por ancho de pulsos, es decir, PWM.

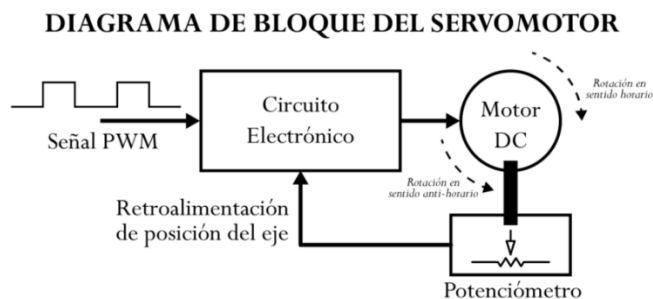


Figura12: Diagrama de un servomotor

Fuente:
<https://panamahitek.com/que-es-y-como-funciona-un-servomotor/>

El diagrama de bloque del servomotor representa de forma visual el servomotor como un sistema. El circuito electrónico es el encargado de recibir la señal PWM y traducirla en movimiento del Motor DC. El eje del motor DC está acoplado a un potenciómetro, el cual permite formar un divisor de voltaje. El voltaje en la salida del divisor varía en función de la posición del eje del motor DC.

El potenciómetro permite que el circuito de control electrónico pueda retroalimentarse con la posición del motor en un momento dado. Esto, en Teoría de Control se conoce como un sistema de lazo cerrado.

Las señales de PWM requeridas para que los circuitos de control electrónico son similares para la mayoría de los modelos de servo. Esta señal tiene la forma de una onda cuadrada. Dependiendo del ancho del pulso, el motor adoptará una posición fija.

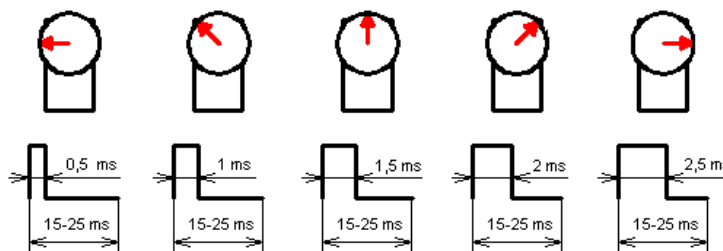


Figura13: SEÑALES PWM

Fuente:<https://panamahitek.com/que-es-y-como-funciona-un-servomotor/>

Las señales que vemos en la imagen son las que permiten que el eje del motor adquiera determinada posición. Éstas señales deben repetirse en el tiempo para que el motor mantenga una posición fija.

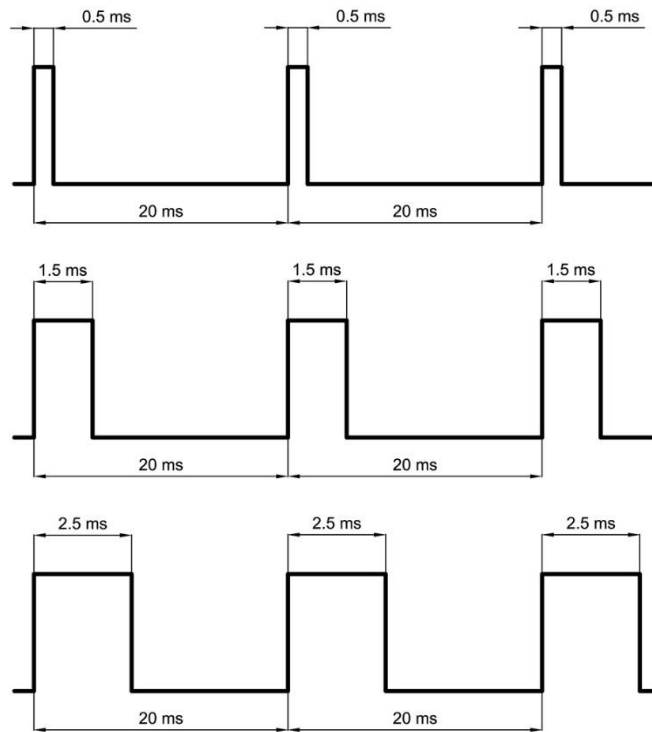


Figura14: SEÑAL PWM

Fuente: <https://panamahitek.com/que-es-y-como-funciona-un-servomotor/>

La duración del ciclo de trabajo varía entre 15 y 25 milisegundos. Las ondas mostradas en la imagen anterior representan ejemplos de trenes de pulsos con los que se puede mover un servomotor, utilizando un ciclo de trabajo de 20 milisegundos.

Este tren de pulsos puede ser generado por un circuito oscilador (como un 555) o por un microcontrolador. Es decir, con Arduino podemos controlar fácilmente un servomotor que en nuestro caso será el ESP32.

b) Características del servomotor (Micro servo 9g – SG90)

En nuestro caso particular ya que se trata de un diseño y se presentará una maqueta del funcionamiento del mismo se utilizará el servomotor de modelismo los cuales operan a voltajes

bajos entre 4 a 6 voltios, además se utilizará servomotores de rango de giro limitado los cuales permiten una rotación de 180 grados el cual se muestra en el siguiente gráfico.



Figura15: servomotor

Fuente: <https://codiziapp.com/introduccion-a-arduino/30-descripcion-servomotor>

- Este servo es un tipo de motor que puede posicionarse directamente en el ángulo deseado por el usuario.
- De rango de movimiento de entre 0 a 180°. No son capaces de dar la vuelta completa por el hecho de que cuentan con un tope que limita ese rango de movimiento.
- Funciona con una señal PWM, con un pulso de entre 1 ms y 2ms, con un periodo de 20ms (50 Hz).
- Admiten un voltaje entre 4.8v a 7.2v, siendo 6v el voltaje adecuado. Con voltajes bajos, el servo gira con una fuerza y velocidad baja y con voltajes altos, el servo comienza a oscilar a tal grado de ser poco útil.
- El servo cuenta con tres cables, dos de alimentación (Vcc y GND) y uno de datos (SIG).

c) Partes de un servomotor

- **Motor de corriente continua (DC):** Es el componente motriz del servo que le proporciona movimiento. Al aplicarse potencial a ambos terminales, el motor gira en uno de los sentidos a máxima velocidad. Si el voltaje que se aplica es inverso, también se invierte el giro.
- **Engranajes reductores:** El conjunto de engranajes sirve para disminuir la velocidad elevada del giro motor y así aumentar la capacidad de torque o par motor.
- **Sensor de desplazamiento:** Es un potenciómetro que va en el eje de salida del servomotor y se utiliza para saber cuál es la posición angular del motor.

- **Circuito de control:** Se trata de una placa electrónica con estrategia de control de posición usando realimentación. Para lograrlo, el circuito va a comparar la señal de referencia de entrada o la posición deseada con la posición medida por un potenciómetro. La diferencia entre posiciones se amplifica y se utiliza para mover el servo en la dirección requerida y reducir el error.

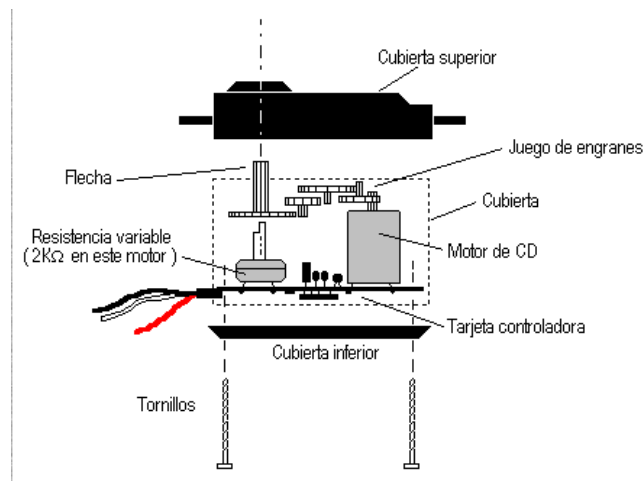


Figura16: Partes de un servomotor

Fuente:<https://como-funciona.co/un-servomotor/>

d) Aplicaciones en el Control de puertas, persianas y otros dispositivos

Los servomotores se utilizan en una variedad de aplicaciones, como el control de puertas, persianas y otros dispositivos. En estas aplicaciones, el servomotor se utiliza para controlar la posición de un objeto, lo que permite abrir o cerrar una puerta, levantar o bajar una persiana, o controlar la posición de otros dispositivos. Además, los servomotores también se utilizan en sistemas de automatización para controlar la posición y velocidad de los componentes del sistema (Ogata, 2010).

En el caso del presente la aplicación a ser utilizada es el control de puertas mediante una señal específica de los dedos el cual permitirá la apertura o cierre de las mismas generando en la persona con discapacidad autonomía y dejar de depender de terceros para realizar acciones tan sencillas pero limitadas para personas con discapacidad.

2.3. VISIÓN ARTIFICIAL

La visión artificial es una técnica que permite a las máquinas "ver" y analizar imágenes o videos, imitando la capacidad del ojo humano. Se basa en el procesamiento digital de la imagen para extraer características y reconocer patrones en las imágenes. Se utiliza en una amplia gama de aplicaciones, desde el control de calidad en la producción industrial hasta la identificación de objetos en imágenes médicas o de vigilancia (González & Woods, 2018).

En nuestro trabajo el procesamiento de la imagen captada por un web cam o cámara de la laptop o computadora recibirá la imagen la cual será procesada por la computadora o laptop y en específico se procesará los dedos levantados de una mano el cual nos permitirá realizar el control de la iluminación y puertas.

2.3.1. Detección de dedos levantados en una imagen

La detección de gestos de los dedos es una técnica de procesamiento de imágenes que permite identificar la posición y movimiento de los dedos en una imagen o video. Esta técnica es de gran importancia en aplicaciones de control para personas con discapacidad, ya que les permite interactuar con dispositivos tecnológicos mediante gestos manuales en lugar de tener que utilizar otros medios, como el teclado y el mouse.

Según González, R., & Pérez, A. (2019), la detección de gestos de los dedos se realiza mediante el uso de algoritmos de visión artificial que analizan los patrones y características de la imagen para identificar la posición y movimiento de los dedos. Para ello, se utilizan técnicas de procesamiento de imágenes como la segmentación, la detección de contornos y la extracción de características, entre otras.

En resumen, para la detección de manos y dedos, se utilizan librerías y herramientas como OpenCV y MediaPipe, que permiten la implementación de algoritmos de visión artificial para el análisis de imágenes y videos en tiempo real. Estas herramientas son muy útiles para la detección de gestos de los dedos en aplicaciones de control para personas con discapacidad, ya que permiten

una interacción más natural e intuitiva con los dispositivos y sin la necesidad de otros medios de entrada.

2.3.2.Librerías y Herramientas para la Detección de Manos y Dedos en imágenes utilizando Visión Artificial

La detección de manos y dedos en imágenes utilizando visión artificial es una tarea compleja que requiere de la utilización de librerías y herramientas especializadas. Según Bradski y Kaehler (2008), OpenCV es una librería de visión por computadora de código abierto que proporciona una amplia variedad de algoritmos y herramientas para procesamiento de imágenes, incluyendo la detección de objetos y características en las imágenes, como la detección de manos y dedos. Por otro lado, Cao et al. (2017) mencionan que MediaPipe es una plataforma de desarrollo de procesamiento de medios móviles y de visión por computadora de código abierto que incluye herramientas para la detección de manos y dedos en tiempo real en imágenes de video.

Estas librerías y herramientas utilizan una variedad de algoritmos y técnicas de procesamiento de imágenes para la detección de manos y dedos. Según Cao et al. (2017), estas técnicas incluyen el análisis de color y textura, el seguimiento de características como bordes y esquinas, y el uso de redes neuronales y aprendizaje profundo para la detección de patrones y características específicas en las imágenes.

La detección de manos y dedos es de gran importancia en aplicaciones de control para personas con discapacidad, ya que permite el control de dispositivos y sistemas electrónicos mediante gestos de las manos y dedos, mejorando la calidad de vida y la independencia de las personas con discapacidad.

2.3.2.1. OpenCV

En nuestro caso utilizaremos OpenCV como librería para programarlo en Python, para utilizar OpenCV en Python para la detección de manos, es necesario tener instalado Python en el equipo, así como la librería OpenCV. Para instalar OpenCV en Python, se puede utilizar el administrador de paquetes pip, ejecutando el siguiente comando en la terminal:

```
pip install opencv-python
```

Una vez instalada la librería, se puede empezar a utilizarla en un script de Python. A continuación, se muestra un ejemplo básico de detección de manos con OpenCV en Python:

```
import cv2
# Se carga el clasificador pre-entrenado para la detección de manos
hand_cascade = cv2.CascadeClassifier('path/to/hand/cascade.xml')
# Se carga la imagen a procesar
img = cv2.imread('path/to/image.jpg')
# Se convierte la imagen a escala de grises para facilitar el procesamiento
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Se detectan las manos en la imagen
hands = hand_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
# Se dibujan los rectángulos alrededor de las manos detectadas
for (x, y, w, h) in hands:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
# Se muestra la imagen con las manos detectadas
cv2.imshow('Hands detected', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

En este ejemplo, se carga un clasificador pre-entrenado para la detección de manos, el cual se utiliza para detectar las manos en una imagen. La detección se realiza mediante la función `detectMultiScale()` de OpenCV, la cual recibe como parámetros la imagen a procesar, el factor de escala, y el número mínimo de vecinos requeridos para aceptar una detección. Los rectángulos alrededor de las manos detectadas se dibujan utilizando la función `rectangle()` de OpenCV.

Es importante mencionar que este es solo un ejemplo básico de detección de manos con OpenCV en Python, y que existen muchas técnicas y algoritmos más avanzados para la detección y seguimiento de manos en tiempo real, como los que se utilizan en MediaPipe.

a. Comandos básicos de OpenCV

Según la página oficial de OpenCV. (2021), La biblioteca de OpenCV proporciona una variedad de herramientas y funciones para capturar imágenes de video en tiempo real utilizando una cámara web conectada a la computadora. Para ello, se pueden utilizar los siguientes comandos básicos en Python:

```
import cv2
cap = cv2.VideoCapture(0) # Crear objeto de captura de video
# Ciclo para leer imágenes del video
while(True):
    ret, frame = cap.read() # Leer frame del video
    cv2.imshow('Captura de video', frame) # Mostrar imagen en una ventana
    if cv2.waitKey(1) & 0xFF == ord('q'): # Esperar tecla para salir del ciclo
        break
# Liberar objeto de captura y cerrar ventanas
cap.release()
cv2.destroyAllWindows()
```

En este ejemplo, se utiliza la función `VideoCapture()` para crear un objeto que representa la cámara web conectada a la computadora. El número 0 como argumento indica que se utilizará la cámara predeterminada. Luego, se utiliza un ciclo `while` para leer imágenes del video capturado y mostrarlas en una ventana utilizando la función `imshow()`.

El comando `waitKey()` permite esperar por una tecla presionada, en este caso se espera por la tecla `q` para salir del ciclo y cerrar las ventanas. Finalmente, se liberan los recursos de la cámara y se cierran las ventanas utilizando las funciones `release()` y `destroyAllWindows()` respectivamente.

b. Aplicaciones de OpenCV

Bradski y Kaehler (2008) mencionan que "OpenCV es una librería de visión por computadora de código abierto diseñada para el procesamiento de imágenes y videos en tiempo real. Proporciona

una amplia gama de herramientas y funciones para el procesamiento de imágenes, como la detección de objetos, la segmentación de imágenes, la calibración de cámaras y la recuperación de la profundidad" (p. 1).

OpenCV (Open Source Computer Vision Library) es una librería de código abierto diseñada para el procesamiento de imágenes y videos en tiempo real. Fue desarrollada originalmente por Intel en el año 2000 y actualmente es mantenida por la comunidad de desarrolladores de todo el mundo. OpenCV proporciona una amplia gama de herramientas y funciones para el procesamiento de imágenes, que incluyen la detección de objetos, la segmentación de imágenes, la calibración de cámaras y la recuperación de la profundidad. Estas herramientas son implementadas en varios lenguajes de programación, como C++, Python, Java y MATLAB.

Para acceder a los comandos disponibles en OpenCV se puede consultar su documentación oficial, disponible en su sitio web oficial: <https://docs.opencv.org/>. La documentación incluye tutoriales, ejemplos de código, descripción de funciones, estructuras de datos y algoritmos, así como la referencia de los diferentes módulos de OpenCV.

OpenCV es una librería de visión por computadora que ha sido utilizada en diversas aplicaciones para personas con discapacidad, incluyendo el control de iluminación y puertas. En el control de iluminación, OpenCV ha sido utilizado para detectar el movimiento de las personas y ajustar la iluminación de la habitación de acuerdo con la ubicación de la persona, proporcionando una iluminación adecuada y mejorando la comodidad de las personas con discapacidad visual. En el control de puertas, OpenCV ha sido utilizado para detectar la presencia de las personas y abrir o cerrar las puertas automáticamente, mejorando la accesibilidad y la independencia de las personas con discapacidad física.

Además, OpenCV ha sido utilizado en la detección de gestos de las manos y dedos para el control de dispositivos electrónicos, como televisores y computadoras, permitiendo que las personas con discapacidad motriz controlen estos dispositivos mediante gestos de las manos y dedos.

En nuestro caso utilizaremos OpenCV como herramienta para procesar la imagen capturada y procesarla para poder incorporarla con la herramienta media pipe y lograr detectar los dedos de

las manos e identificar cual dedo esta levantado y de esta manera poder controlar la iluminación y puertas que le ayudaran a la persona con discapacidad tener la autonomía y dejar de depender de terceros.

2.3.2.2. MediaPipe

Según la página oficial de MediaPipe. (2021), MediaPipe es una librería de código abierto desarrollada por Google que permite el seguimiento y análisis de los movimientos del cuerpo y las manos en tiempo real. En particular, MediaPipe ofrece una solución para la detección de dedos en las manos mediante su módulo de seguimiento de manos.

Para utilizar MediaPipe en Python y detectar los dedos en las manos, se pueden seguir los siguientes pasos:

- Instalar MediaPipe: Es necesario instalar MediaPipe en Python mediante el comando "pip install mediapipe".
- Importar la librería: Se debe importar la librería en el archivo Python mediante el comando "import mediapipe as mp".
- Crear un objeto de detección de manos: Se debe crear un objeto de la clase "mp.solutions.hands.Hands" para inicializar el modelo de detección de manos.
- Capturar y procesar los fotogramas del video: Se debe capturar los fotogramas del video mediante una cámara o un archivo de video y procesarlos para detectar los dedos en las manos mediante el objeto de detección creado anteriormente.
- Obtener la posición de los dedos: Se puede obtener la posición de cada uno de los dedos mediante los puntos detectados por el modelo de MediaPipe.

Un ejemplo de código para la detección de dedos en las manos con MediaPipe en Python es el siguiente:

```
import cv2
import mediapipe as mp
hands = mp.solutions.hands.Hands()# Crear objeto de detección de manos
cap = cv2.VideoCapture(0) # Capturar video desde cámara
```

```

while True:
    ret, frame = cap.read() # Capturar fotograma del video
    # Convertir imagen a formato RGB
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = hands.process(image) # Detectar manos en la imagen
    # Dibujar puntos en los dedos detectados
    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            for id, lm in enumerate(hand_landmarks.landmark):
                h, w, c = frame.shape
                cx, cy = int(lm.x*w), int(lm.y*h)
                cv2.circle(frame, (cx, cy), 5, (255, 0, 0), cv2.FILLED)
    cv2.imshow("Hand Detection", frame) # Mostrar imagen procesada
    # Salir del bucle si se presiona la tecla "q"
    if cv2.waitKey(1) == ord('q'):
        break
# Liberar recursos
cap.release()
cv2.destroyAllWindows()

```

Este ejemplo utiliza la cámara del dispositivo para capturar el video en tiempo real y procesarlo para detectar los dedos en las manos mediante MediaPipe. Los puntos detectados se dibujan en la imagen procesada y se muestran en una ventana de visualización.

a. Comandos básicos de mediaPipe

MediaPipe es una biblioteca de procesamiento de medios y visión por computadora desarrollada por Google. Permite la detección y seguimiento de objetos y características en tiempo real utilizando modelos de aprendizaje automático pre-entrenados. A continuación, se presentan algunos de los comandos básicos de MediaPipe en Python:

Según la página oficial de mediapipe Google. (2021), proporciona algunos comandos básicos para poder realizar la detección de los dedos de las manos, pero también mediaPipe indica la manera correcta de realizar la detección de manos

- Importar MediaPipe: Para utilizar MediaPipe en Python, se debe importar la biblioteca utilizando el siguiente comando:

```
import mediapipe as mp
```

- Cargar el modelo de detección: Para utilizar los modelos pre-entrenados de MediaPipe, es necesario cargarlos utilizando el siguiente comando:

```
hands = mp.solutions.hands.Hands()
```

Este comando carga el modelo de detección de manos, que puede ser utilizado para detectar y seguir las manos en tiempo real.

- Procesamiento de imágenes o video: Para procesar imágenes o video utilizando MediaPipe, se puede utilizar el siguiente comando:

```
with mp.solutions.hands.Hands() as hands:
```

```
    # loop para procesar cada fotograma de la secuencia de video
```

```
    for frame in video_frames:
```

```
        # Conversión del fotograma a RGB para la detección de manos
```

```
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
        # Detección de las manos en el fotograma
```

```
        results = hands.process(frame_rgb)
```

- Obtener los resultados de la detección: Para obtener los resultados de la detección de MediaPipe, se puede utilizar el siguiente comando:

```
if results.multi_hand_landmarks:
```

```
    for hand_landmarks in results.multi_hand_landmarks:
```

```
        # Procesamiento de las coordenadas de los puntos de referencia de las manos
```

```
        x = hand_landmarks.landmark[0].x
```

```
        y = hand_landmarks.landmark[0].y
```

Este código utiliza los resultados de la detección de MediaPipe para obtener las coordenadas de los puntos de referencia de las manos detectadas en la imagen o video.

b. Modelos para el procesamiento de las coordenadas de los puntos de referencia de las manos

Según la página oficial de Mediapipe de Google (2021), El Hand Landmarker utiliza un paquete de modelos con dos modelos empaquetados: una detección de palma y un modelo de detección de puntos de referencia de mano. Necesita un paquete de modelos que contiene ambos modelos para ejecutar esta tarea.

El paquete de modelos de punto de referencia de mano detecta la localización del punto clave de 21 Coordenadas de nudillos de mano dentro de las regiones de mano detectadas. El modelo fue entrenado en aproximadamente 30K imágenes del mundo real, así como varias renderizadas sintéticas Modelos de mano impuestos sobre diversos fondos.

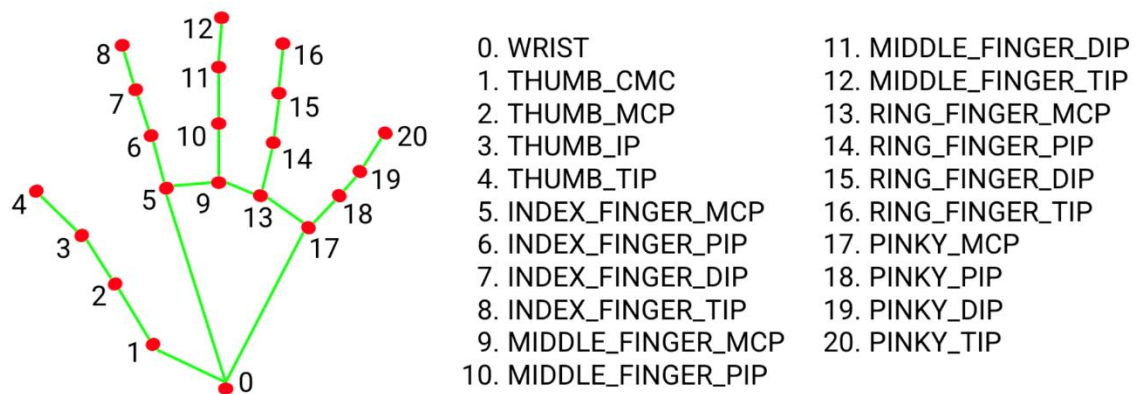


Figura17: Puntos de referencia de una Mano

Fuente: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

Según la página oficial de Mediapipe de Google (2021), Para usar Hand Landmarker de Mediapipe en Python, es necesario seguir los siguientes pasos:

- Importar las librerías necesarias:

```
import cv2
import mediapipe as mp
```
- Inicializar el modelo de detección de puntos de referencia de las manos:

```
mp_drawing = mp.solutions.drawing_utils
```



```

mp_hands = mp.solutions.hands
hands = mp_hands.Hands(
min_detection_confidence=0.5,
min_tracking_confidence=0.5)

```

En este caso, `min_detection_confidence` y `min_tracking_confidence` son los valores mínimos de confianza necesarios para detectar y seguir los puntos de referencia de las manos.

- Capturar la imagen de la cámara o cargar una imagen existente:

```

cap = cv2.VideoCapture(0) # Capturar imagen de la cámara #
ó
image = cv2.imread('ruta/de/la/imagen.jpg') # Cargar imagen existente

```

- Convertir la imagen a escala de grises:

```

image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

```

- Detectar los puntos de referencia de las manos:

```

results = hands.process(image_gray)

```

- Dibujar los puntos de referencia en la imagen:

```

if results.multi_hand_landmarks:
    for hand_landmarks in results.multi_hand_landmarks:
        mp_drawing.draw_landmarks(
            image, hand_landmarks, mp_hands.HAND_CONNECTIONS)

```

En este caso, `mp_drawing.draw_landmarks()` es utilizado para dibujar los puntos de referencia en la imagen.

- Mostrar la imagen con los puntos de referencia detectados:

```

cv2.imshow('Hand Landmarks', image)
cv2.waitKey(0)

```

Estos son los pasos básicos para utilizar Hand Landmarker de Mediapipe en Python para la detección de puntos de referencia de las manos.

c. Aplicaciones de mediaPipe

Según la página oficial de MediaPipe. (2021), MediaPipe es una biblioteca de código abierto desarrollada por Google que permite construir aplicaciones de percepción de computadora en tiempo real, como reconocimiento facial, seguimiento de mano, detección de objetos, entre otros. Esta herramienta puede ser utilizada para desarrollar aplicaciones para personas con discapacidad, como las siguientes:

- Reconocimiento de gestos: mediante la captura de movimiento de las manos y brazos, MediaPipe permite reconocer gestos específicos que pueden ser utilizados como comandos de control en aplicaciones para personas con discapacidad.
- Seguimiento ocular: MediaPipe permite el seguimiento en tiempo real del movimiento ocular, lo que puede ser utilizado para el control de aplicaciones mediante la mirada.
- Detección de emociones: MediaPipe puede ser utilizado para la detección de emociones en el rostro de una persona, lo que puede ser útil en aplicaciones de asistencia emocional para personas con discapacidad.

Estos son solo algunos ejemplos de las aplicaciones que se pueden desarrollar utilizando MediaPipe para personas con discapacidad en nuestro caso mediapipe será utilizado para reconocer gestos específicos de dedos levantados los cuales le permitirán a las personas con discapacidad controlar el encendido y apagado de luces, así como la apertura y cierre de las puertas.

2.3.2.3. Integrando OpenCV y mediaPipe

"OpenCV es una biblioteca de visión por computadora de código abierto ampliamente utilizada para procesamiento de imágenes y video. MediaPipe es una biblioteca de percepción de computadora que también es de código abierto y ofrece una variedad de soluciones para tareas de reconocimiento de objetos, seguimiento de mano, entre otros" (Redacción TICbeat, 2021). Ambas bibliotecas pueden ser utilizadas juntas para desarrollar aplicaciones más avanzadas, como detectores de dedos levantados el cual será utilizado para nuestro diseño de prototipo.

En concreto, se puede utilizar OpenCV para la captura de imágenes y la segmentación de la mano, mientras que MediaPipe puede ser utilizado para el seguimiento en tiempo real de la mano y la detección de dedos levantados. El resultado es un sistema de detección de dedos levantados en tiempo real que puede ser utilizado en aplicaciones de asistencia para personas con discapacidad, entre otros.

Es importante tener en cuenta que la integración de OpenCV y MediaPipe para la detección de dedos levantados es solo una de las muchas aplicaciones posibles que se pueden desarrollar con estas bibliotecas.

2.3.2.4. Comunicación serial

La comunicación serial es una forma de transmisión de datos en la que los bits se envían uno tras otro a través de un canal de comunicación, como un cable o un puerto de comunicación. Los datos se transmiten en serie, es decir, un bit después del otro, en lugar de en paralelo, donde varios bits se transmiten simultáneamente.

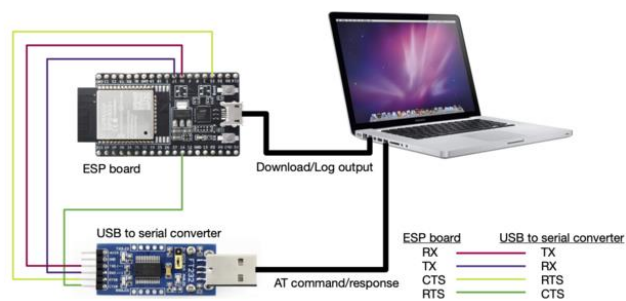


Figura18: Comunicación SERIAL

Fuente: https://docs.espressif.com/projects/esp-at/zh_CN/latest/_images/esp32-solo-hw-connection.png

La comunicación serial se utiliza en una amplia variedad de aplicaciones, desde la transferencia de datos entre computadoras y dispositivos periféricos, hasta la comunicación entre

microcontroladores y sensores. Hay diferentes tipos de protocolos de comunicación serial, como UART, SPI y I2C, cada uno con sus propias especificaciones y características.

Según Jain, N. K. (2010). La comunicación serial entre Python y ESP32 generalmente se realiza utilizando el protocolo UART (Universal Asynchronous Receiver/Transmitter), que es un protocolo de comunicación serial asíncrono comúnmente utilizado en microcontroladores y otros dispositivos electrónicos.

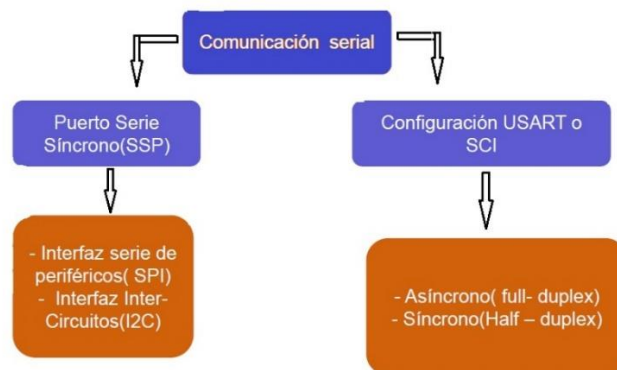


Figura19: clasificación de la comunicación Serial

Fuente:https://4.bp.blogspot.com/-k1TOXZwF82c/Vb_5NGX5aHI/AAAAAAAAA1A/ktJXqQ6FhRQ/s1600/Serial%2B1.jpg

En este tipo de comunicación, los datos se transmiten en serie, es decir, un bit a la vez, utilizando una línea de transmisión y otra de recepción. El protocolo UART utiliza un formato de trama de datos que incluye un bit de inicio, los datos que se están transmitiendo y uno o más bits de paridad para detectar errores de transmisión. y/o bits de parada para asegurar la integridad de la transmisión.

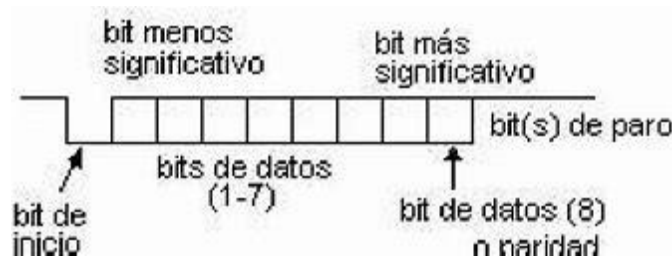


Figura20: Trama de la TX serial

Fuente:<https://th.bing.com/th/id/OIP.afpvL14yFJJ1V0ky6Ke52AHaCq?pid=ImgDet&rs=1>

La comunicación serial mediante el protocolo UART se caracteriza por su simplicidad, bajo costo y amplia disponibilidad en diferentes dispositivos electrónicos.

a) **Configurar la Comunicación Serial de Python a ESP32**

La comunicación serial de Python con un ESP32 es un proceso que permite enviar y recibir datos entre el microcontrolador y el ordenador a través de un puerto serial. Según Wu y Zhang (2021), "Python es uno de los lenguajes de programación más populares para la interacción con los microcontroladores debido a su simplicidad y facilidad de uso" (p. 66). Para lograr la comunicación serial, se utiliza la biblioteca "pyserial" de Python, que permite establecer una conexión serial y enviar y recibir datos a través de ella.

Para realizar la comunicación serial entre Python y ESP32, se deben seguir los siguientes pasos básicos:

- Importar la biblioteca "pyserial" de Python:
`import serial`
- Establecer la conexión serial con el ESP32:
`ser = serial.Serial('puerto_serial', baudrate=9600, timeout=1)`
- Enviar datos al ESP32 a través de la conexión serial:
`ser.write('datos_a_enviar'.encode())`
- Recibir datos del ESP32 a través de la conexión serial:
`datos_recibidos = ser.readline().decode().rstrip()`

Es importante tener en cuenta que el puerto_serial debe ser el puerto al que está conectado el ESP32 y que la velocidad de transmisión (baudrate) debe ser la misma en ambos dispositivos.

b) **Configurar la Comunicación Serial de ESP32 a Python**

Ahora las configuraciones básicas para recibir o enviar información en el esp32 serían las siguientes

```
void setup() { // Configuración del puerto serial
```

```

Serial.begin(9600);
}
void loop() { // Lectura de datos por el puerto serial
    if (Serial.available()) {
        String datosRecibidos = Serial.readStringUntil('\n');
        Serial.print("Datos recibidos: ");
        Serial.println(datosRecibidos);
        // Envío de datos al puerto serial
        Serial.println("Hello, Python!");
    }
    delay(1000);
}

```

Los ejemplos de configuración son básicos y el uso de la comunicación serial varía dependiendo de la aplicación específica. El uso de la detección de dedos levantados para personas con discapacidad es un poco más complejo en su recepción de datos desde Python y utilidad para realizar el control de iluminación y puertas.

2.4. ESP32

El WROOM-32 DEVKIT V1, es un potente módulo que integra WiFi y Bluetooth, ideal para desarrollar productos de IoT. Permite una amplia gama de aplicaciones, el uso de WiFi permite una comunicación de mediano alcance y conectarse a una red LAN y a través de un Modem Router con conexión a Internet, mientras que el Bluetooth nos permite conectarse directamente a otro dispositivo como un celular. El ESP32 es capaz de funcionar de forma fiable en entornos industriales, con una temperatura de funcionamiento que oscila entre -40 °C y +125 °C. Alimentado por circuitos de calibración avanzados, ESP32 puede eliminar dinámicamente las imperfecciones del circuito externo y adaptarse a los cambios en las condiciones externas como se muestra en la siguiente figura.



Figura21: modulo ESP32

Fuente:<https://www.majju.pk/assets/uploads/2020/07/download-4-2000x2000.png>

EL ESP WROOM-32 DEVKIT V1, trabaja con una corriente es inferior a 5 μ A, por lo que es adecuado para aplicaciones de electrónica portátiles con batería. En el núcleo de este módulo está el SoC ESP32-D0WDQ6. El chip integrado está diseñado para ser escalable y adaptado., y la frecuencia del reloj es ajustable de 80 MHz a 240 MHz. El usuario también puede apagar el CPU y utilizar el co-procesador de baja potencia para supervisar constantemente los periféricos para detectar cambios de estado.

El ESP WROOM-32 DEVKIT V1, integra un amplio conjunto de periféricos en los cuales se pueden conectar sensores táctiles capacitivos, sensores Hall, amplificadores de bajo nivel de ruido, interfaz para SD, Ethernet, SPI, UART, I2S e I2C. Para flashear el chip es necesario utilizar un módulo conversor USB a serial TTL como el Módulo CP2102.

El módulo ESP WROOM-32 DEVKIT V1, trabaja a 3.3V en alimentación y GPIO por lo que NO se debe alimentar con 5V. Se recomienda colocar un capacitor de 100uF en paralelo con la fuente de alimentación para filtrar los picos de corriente. Los pines de entradas/salidas (GPIO) trabajan a 3.3V por lo que para la conexión a sistemas de 5V es necesario utilizar convertidores de nivel como: Conversor de nivel 3.3-5V 4CH o Conversor de nivel bidireccional 8CH - TXS0108E. 20

La programación del ESP-WROOM-32 DEVKIT V1 se la realizar en el entorno de Arduino IDE por las ventajas que ofrece. Ya que podremos utilizar un lenguaje de programación conocido y hacer uso de un IDE sencillo de utilizar, además de hacer uso de toda la información sobre proyectos y librerías disponibles en internet. La comunidad de usuarios de Arduino es muy activa y da soporte a plataformas como el ESP32.

2.4.1.Descripción del Microcontrolador ESP32

Diagrama de bloques del ESP32

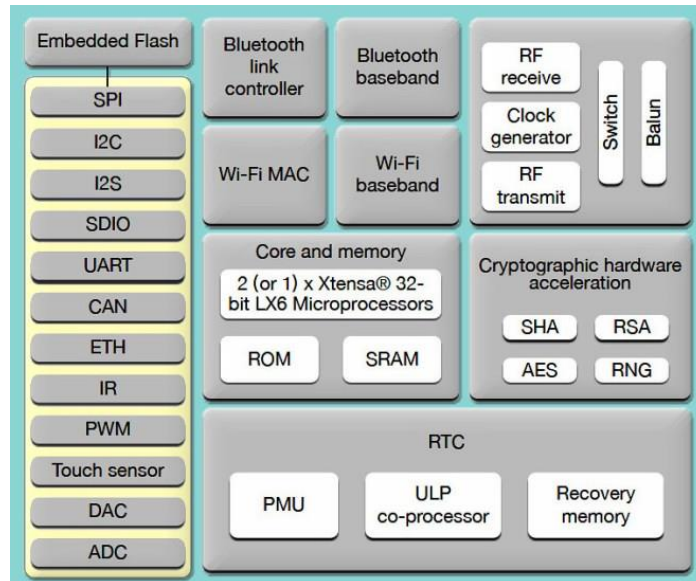


Figura22: DIAGRAM DE BLOQUES ESP32

Fuente:http://j-rpm.com/2020/09/transmisor-dcf77-con-esp32/esp32_bloques/

2.4.2.Características

- Conectividad inalámbrica:
- WiFi : velocidad de datos de 150,0 Mbps con HT40
- Bluetooth: BLE (Bluetooth de baja energía) y Bluetooth Classic
- Procesador: microprocesador Tensilica Xtensa Dual-Core de 32 bits LX6, que funciona a 160 o 240 MHz
- ROM: 448 KB y SRAM: 520 KB
- Bajo consumo: garantiza que se pueda usar en conversiones ADC, por ejemplo, durante el sueño profundo.
- Voltaje de alimentación es de 3.3 V, pero puede operar en un rango de (2.7 hasta 3.6 V)
- Corriente de operación aproximadamente de 80mA
- Entrada/salida periférica: voltaje lógico entradas/salidas (GPIO) es de 3.3 V
- Interfaz periférica con DMA que incluye toque capacitivo

- ADC (convertidor analógico a digital) tiene 2 (12 bit)
- DAC (convertidor de digital a analógico) tiene 2 de (8 bit)
- SPI (interfaz periférica en serie) tiene 3
- I²S (sonido interchip integrado)
- I²C (circuito interintegrado) tiene 2
- UART (receptor/transmisor asincrónico universal) tiene 2
- RMII (Interfaz independiente de medios reducida)
- PWM (modulación de ancho de pulso).
- Seguridad: aceleradores de hardware para AES y SSL / TLS
- Compatible con Arduino IDE: puede programar el ESP32 con Arduino IDE (instrucciones de instalación de Windows, Mac OS X y Linux).
- Compatible con MicroPython: puede programar el ESP32 con firmware MicroPython (comience con MicroPython en ESP32)

2.4.3. Memoria

En los ESP32 tiene más memorias que se suelen clasificar en internas y externas. Las memorias internas son aquellas que se encuentran ya incluidas en el SoC, y las externas son aquellas que se pueden adicionar para expandir la capacidad del sistema. Muchas placas de desarrollo basadas en ESP32 añaden memorias externas para lograr el buen funcionamiento del mismo.

Entre las memorias internas encontramos:

- Memoria ROM (448 KB): esta memoria es de solo escritura, es decir que no la puedes reprogramar. Aquí es donde se almacenan los códigos que manejan la pila Bluetooth, el control de la capa física de la Wifi, algunas rutinas de propósito general y el cargador de arranque (bootloader) para iniciar el código de la memoria externa.
- Memoria SRAM interna (520 KB): esta memoria es utilizada por el procesador para almacenar tanto datos como instrucciones. Su ventaja es que, para el procesador, es mucho más fácil acceder a esta que a la SRAM externa.
- RTC SRAM (16 KB): esta memoria es utilizada por el co-procesador cuando el dispositivo opera en modo deep sleep.

- Efuse (1 Kilobit): 256 bits de esta memoria son utilizados por el propio sistema y los 768 bits restantes están reservados para otras aplicaciones.
- Flash empotrado (Embedded flash): en esta memoria es donde se almacena el código de nuestra aplicación. La cantidad de memoria varía en dependencia del chip utilizado:
 - 0 MB (chips ESP32-D0WDQ6, ESP32-D0WD y ESP32-S0WD)
 - 2 MB (chip ESP32-D2WD)
 - 4 MB (módulo SP ESP32-PICO-D4)

Cuando la memoria es insuficiente para tu aplicación, es posible adicionar más memoria de forma externa:

- Se pueden agregar hasta 16 MB de memoria flash externa. De esta forma puedes desarrollar aplicaciones más complejas.
- También admite, hasta 8 MB de memoria SRAM externa. Por lo tanto, es difícil que te encuentres limitado en memoria al implementar una aplicación utilizando esta plataforma.

2.4.4.Pines disponibles en el ESP32

Hay 34 pines GPIO disponibles en el chip ESP32. Estos pines se nombran de a. ¿Pero eso no hace que la cuenta sea 40? No, porque las GPIO 20, 24, 28, 29, 30 y 31 no son accesibles. Además, no todos estos pines salen del módulo o de la placa

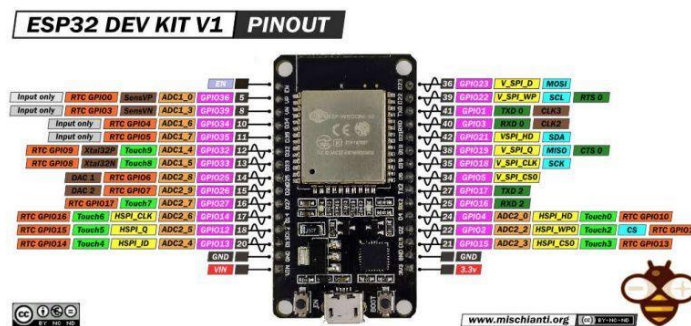


Figura23: PINES ESP32 Fuente: www.mischianti.org

2.4.4.1. Pines de Entradas y salidas

En la siguiente tabla se muestra cuales pines pueden trabajar como entrada y/o salida y sus funciones especiales de cada pin.

GPIO	¿Entrada?	¿Salida?	Notas
0	NO	SÍ	Tire de LOW para entrar en el modo de gestor de arranque.
1	NO	SÍ	TX0 de puerto serie para programar e imprimir mensajes de depuración.
2	SÍ	SÍ	Conectado al LED integrado, debe dejarse flotando o BAJO para entrar en modo intermitente.
3	SÍ	NO	RX0 de puerto serie para programar e imprimir mensajes de depuración.
4	SÍ	SÍ	
5	SÍ	SÍ	Pasador de flejado
6	NO	NO	Interfaz de memoria flash. No usar.
7	NO	NO	Interfaz de memoria flash. No usar.
8	NO	NO	Interfaz de memoria flash. No usar.
9	NO	NO	Interfaz de memoria flash. No usar.
10	NO	NO	Interfaz de memoria flash. No usar.

GPIO	¿Entrada?	¿Salida?	Notas
11	NO	NO	Interfaz de memoria flash. No usar.
12	SÍ	SÍ	Pasador de fleje. El arranque puede fallar si se tira de HIGH (para memorias de 3.3V) debido a una caída de tensión.
13	SÍ	SÍ	
14	SÍ	SÍ	
15	SÍ	SÍ	Al extraer LOW, se silencian los mensajes de depuración a través del puerto serie.
16	SÍ	SÍ	
17	SÍ	SÍ	
18	SÍ	SÍ	
19	SÍ	SÍ	
21	SÍ	SÍ	
22	SÍ	SÍ	
23	SÍ	SÍ	
25	SÍ	SÍ	
26	SÍ	SÍ	
27	SÍ	SÍ	

GPIO	¿Entrada?	¿Salida?	Notas
32	SÍ	SÍ	
33	SÍ	SÍ	
34	SÍ	NO	Sólo entrada
35	SÍ	NO	Sólo entrada
36	SÍ	NO	Sólo entrada
39	SÍ	NO	Sólo entrada

Figura24: DESCRIPCION DE PINES del ESP32

Fuente: <https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/>

2.4.4.2. Pines UART

ESP32 tiene tres UART en su interior (solo asíncrono) con control de flujo de hardware y software. Los UART se denominan UART0, UART1 y UART2. UART0 se utiliza para la programación en serie y para imprimir mensajes de depuración. Este es el UART que utilizamos con el puerto serie USB para imprimir mensajes desde el ESP32. UART1 es la instancia predeterminada en el entorno Arduino. Serial

Serial1 es UART1 y sus pines predeterminados se mezclan con la interfaz QSPI (pines GPIO 6-11). Por lo tanto, no se recomienda su uso sin reasignar los pines. La reasignación de los pines se puede hacer pasando los nuevos pines a la función. Dicho esto, aún puede usar con los GPIO predeterminados 9 (RX1, SD2) y 10 (TX1, SD3), pero solo podrá transmitir. La función de recepción no funcionará. Solo unos pocos tipos de placas ESP32 tienen GPIO 9 y 10 desglosados. Serial1begin () Serial1.

UART2 se asigna a en el boceto de Arduino. Todas las funciones UART se pueden asignar a cualquier pin GPIO que desee. Los predeterminados se enumeran a continuación. Serial2

Instancia de Arduino	UART	RX Pin	TX Pin	CTS	ST2
Serial	UART0	GPIO 3 (RX0)	GPIO 1 (TX0)	N/A	N/A
Serial1	UART1	GPIO 9 (RX1)	GPIO 10 (TX1)	GPIO 6	GPIO 11
Serial2	UART2	GPIO 16 (RX2)	GPIO 17 (TX2)	GPIO 8	GPIO 7

Figura25: Pines ESP32 UART

Fuente: <https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/>

2.4.4.3. ADC

Los convertidores analógicos a digitales (ADC) se utilizan para convertir voltajes analógicos a valores digitales. Hay dos ADC SAR de 12 bits disponibles en ESP32 con 18 canales de entrada. Pero solo 16 canales están disponibles en Arduino. A continuación, se muestra la lista de canales ADC y sus instancias de Arduino. No se utilizan canales ADC ADC1_CH1 y ADC1_CH2.

Arduino Pin	Canal ADC	GPIO	¿Utilizable?
A0	ADC1_CH0	36	SÍ
A3	ADC1_CH3	39	SÍ
A4	ADC1_CH4	32	SÍ

Arduino Pin	Canal ADC	GPIO	¿Utilizable?
A5	ADC1_CH5	33	SÍ
A6	ADC1_CH6	34	SÍ
A7	ADC1_CH7	35	SÍ
A10	ADC2_CH0	4	SÍ
A11	ADC2_CH1	0	NO
A12	ADC2_CH2	2	NO (LED conectado)
A13	ADC2_CH3	15	SÍ
A14	ADC2_CH4	13	SÍ
A15	ADC2_CH5	12	NO
A16	ADC2_CH6	14	SÍ
A17	ADC2_CH7	27	SÍ
A18	ADC2_CH8	25	SÍ
A19	ADC2_CH9	26	SÍ

Figura26: Pines ESP32 ADC

Fuente: <https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/>

2.4.4.4. PWM

ESP32 admite hasta 16 canales PWM (modulación de ancho de pulso) independientes con precisión de 16 bits. Las salidas PWM se pueden asignar a cualquier pin GPIO que admita el modo de salida.

2.4.5. IDE de Arduino y como Programar el ESP32

Arduino es una plataforma de prototipado electrónico que consta de un conjunto de hardware y software diseñados para facilitar el proceso de creación de proyectos electrónicos. La plataforma incluye una placa electrónica con entradas y salidas para conectarse a sensores, actuadores y otros dispositivos, así como un entorno de desarrollo integrado (IDE) que permite escribir, compilar y cargar el código en la placa.

El ESP32 es un microcontrolador de bajo costo y de bajo consumo de energía que es compatible con el entorno de desarrollo de Arduino. Es especialmente útil para proyectos que requieren conectividad inalámbrica, como Wi-Fi o Bluetooth.

La programación del ESP32 en el entorno de Arduino se realiza utilizando el lenguaje de programación C++. El IDE de Arduino proporciona una biblioteca de funciones predefinidas que facilitan la programación de tareas comunes, como la comunicación serial, la lectura de sensores y la activación de actuadores. Además, el entorno permite la integración de bibliotecas de terceros, lo que amplía las capacidades de la plataforma.

Para poder realizar la programación del esp32 en el IDE de Arduino se debe de seguir los siguientes pasos:

2.4.5.1. Descargar e instalar ESP32 en Arduino

Descargar Arduino desde este link <https://www.arduino.cc/en/software>, ahora ya se cuenta con una versión para computadoras de 64 bits y se trata del Arduino IDE 2.1.0 que es más atractivo y llamativo será importante realizar la exploración del mismo, pero aun las versiones anteriores son

funcionales y no tienen ningún problema por cual trabajaremos en nuestro caso trabajaremos con la versión 1.8.19.

Arduino Web Editor

Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

[CODE ONLINE](#) [GETTING STARTED](#)

Downloads

Arduino IDE 2.1.0

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

DOWNLOAD OPTIONS

Windows	Win 10 and newer, 64 bits
Windows	MSI installer
Windows	ZIP file
Linux	Applimage 64 bits (X86-64)
Linux	ZIP file 64 bits (X86-64)
macOS	Intel, 10.14: "Mojave" or newer, 64 bits

Figura27: ARDUINO

Fuente: <https://www.arduino.cc/en/software>

2.4.5.2. Configurar ESP32 en Arduino IDE

Este punto es el más importante ya que requiere de varios pasos para un buen funcionamiento del ESP32

a) Conectamos ESP32 vía USB a la computadora

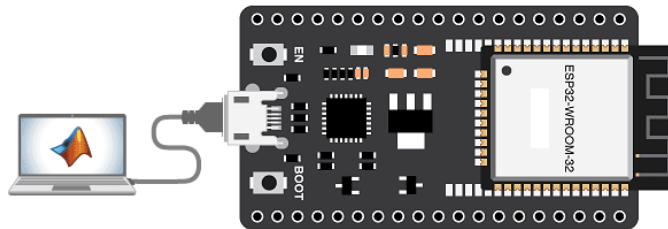


Figura28: Conectando ESP32 via USB

Fuente: <https://www.programadornovato.com/instalar-esp32-en-arduino-ide/>

b) Agregamos las librerías de ESP32

Para ello debemos de agregar el siguiente enlace haciendo click en Archivo -> Preferencias y pegamos este link https://dl.espressif.com/dl/package_esp32_index.json

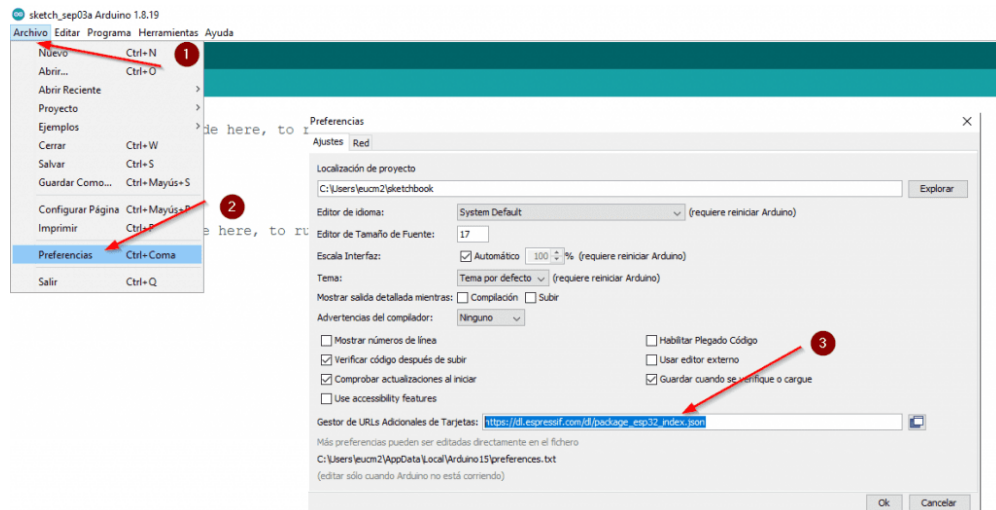


Figura29: LIBRERÍA ESP32

Fuente: <https://www.programadornovato.com/instalar-esp32-en-arduino-ide/>

Luego vamos a Herramientas -> Placa: "Arduino" -> Gestor de tarjetas

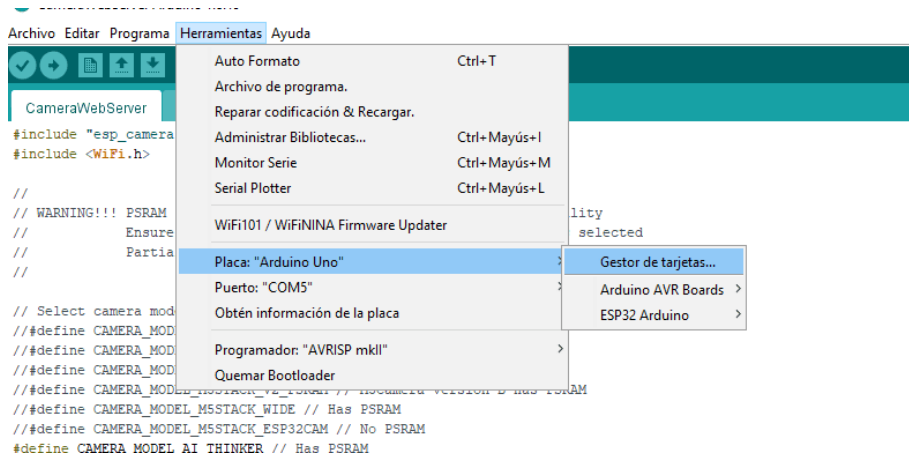


Figura30: Seleccionando la tarjeta en IDE Arduino

Fuente: <https://www.programadornovato.com/instalar-esp32-en-arduino-ide/>

Seguimos y Buscamos esp32 e instalamos la versión 1.0.4 y damos click en instalar

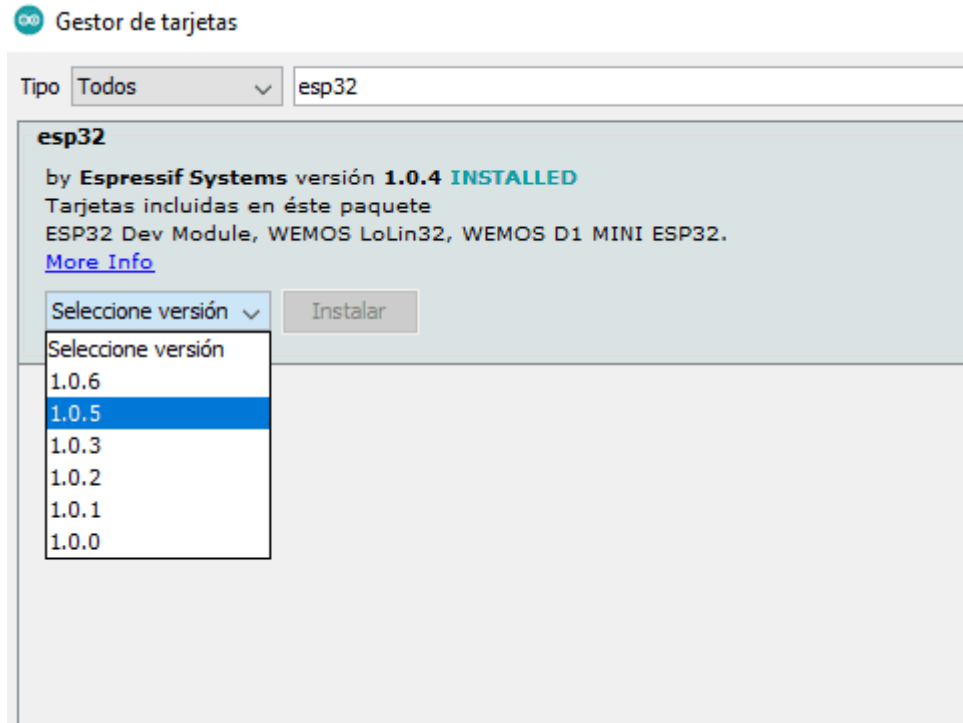


Figura31: Instalando ESP32

Fuente: <https://www.programadornovato.com/instalar-esp32-en-arduino-ide/>

Por ultimo vamos Ahora vamos a Herramientas-> Placa -> ESP32-> DOIT ESP32 DEVKIT V1

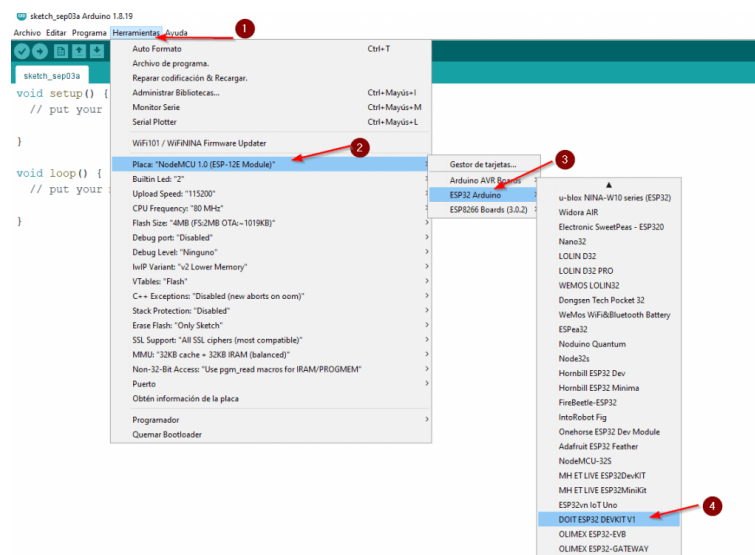


Figura32: Selección de Modulo ESP32

Fuente: <https://www.programadornovato.com/instalar-esp32-en-arduino-ide/>

2.4.5.3. Buscar el puerto de ESP32

Para realizar esta operación Vamos a Herramientas-> Puerto->COM5 en el ejemplo, pero esto puede cambiar en función a la computadora s utilizar y quizás pueda ser diferente.

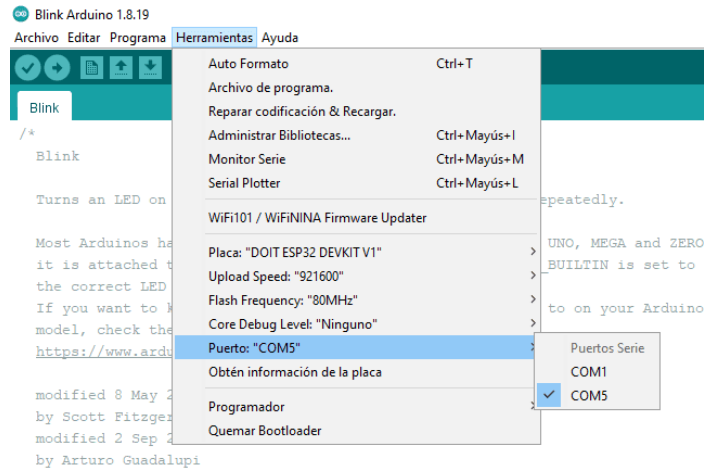


Figura33: Seleccionando Puerto COM

Fuente: <https://www.programadornovato.com/instalar-esp32-en-arduino-ide/>

2.4.5.4. Cargar un código de ejemplo

Para saber si está en buen funcionamiento el ESP32 podemos cargar un código de ejemplo llamado BLINK el cual enciende y apaga el led incorporado en el ESP32

Con todos estos pasos terminados estamos listos para realizar la programación de cualquier proyecto que se desee implementar en el IDE Arduino, en nuestro caso realizamos la programación del programa encargado de realizar el encendido de focos los cuales estarán conectados a relés, apertura de puertas con la ayuda de servomotores y sensor de movimiento con el LDR como sensor de luz, para encender la luz del patio y comunicación al usuario sobre la presencia de una persona

2.4.5.5. Estructura de un programa

La estructura de todo programa consta de dos funciones principales como la:

c) Función de configuración ()

La función `setup()`. Se usa para inicializar las variables, modos de pin, comenzar a usar bibliotecas, etc. La función de configuración solo se ejecutará una vez, después de cada encendido o reinicio de la placa Arduino. Aquí, se puede definir la velocidad de salida en el `Serial.begin()` (declaración que abre el puerto serie para permitir que la placa envíe salida para que la muestre el monitor serie).

d) Función `loop()`

Después de llamar a la función `setup()`, que inicializa y establece los valores iniciales, la función `loop()` hace precisamente lo que sugiere su nombre y se repite consecutivamente, lo que permite que su programa cambie y responda. Se utiliza para controlar activamente la placa Arduino.

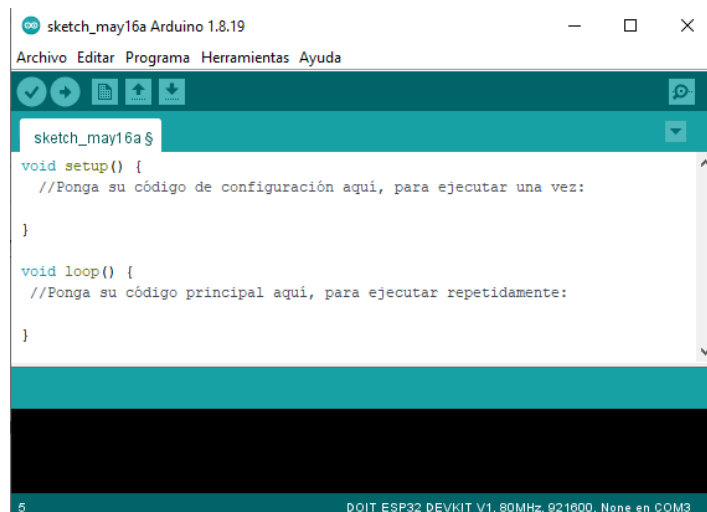


Figura34: Entorno IDE de Arduino

Fuente: Elaboración propia

2.4.6. Versiones de ESP32

Algunas de las versiones que existen en el mercado son:

Hasta mi conocimiento en septiembre de 2021, se conocen varias versiones del ESP32, que son las siguientes:

- ESP32-WROOM-32: Es la versión más común y ampliamente utilizada del ESP32. Viene en un formato de módulo con el ESP32 y otros componentes integrados, como la antena y los reguladores de voltaje.
- ESP32-WROOM-32D: Esta versión es similar al ESP32-WROOM-32, pero cuenta con una mejora en la sensibilidad de recepción de señal y un rango extendido de temperatura de funcionamiento.
- ESP32-WROVER: Esta versión del ESP32 incluye más pines de entrada/salida (E/S) y un módulo de memoria externa PSRAM para una mayor capacidad de almacenamiento.
- ESP32-WROVER-B: Es similar al ESP32-WROVER, pero con mejoras en la sensibilidad de recepción de señal y un rango extendido de temperatura de funcionamiento.
- ESP32-PICO-D4: Es una versión compacta del ESP32 que integra el chip ESP32 junto con otros componentes, como la antena y los reguladores de voltaje, en un formato de paquete pequeño.

Estas son algunas de las versiones más comunes del ESP32. Es posible que haya versiones adicionales o variantes desarrolladas por diferentes fabricantes o con características específicas para aplicaciones especializadas, pero en nuestro caso utilizaremos ESP32-WROOM-32.

2.4.7. Ventajas y Desventajas

ESP32 tiene varias ventajas y desventajas que vale la pena considerar al utilizarlo en proyectos:

2.4.7.1. Ventajas del ESP32

- Potencia y rendimiento: El ESP32 cuenta con un procesador dual-core de alto rendimiento y una velocidad de reloj de hasta 240 MHz. También tiene una memoria RAM suficientemente amplia para ejecutar aplicaciones complejas.
- Conectividad inalámbrica: El ESP32 ofrece conectividad Wi-Fi y Bluetooth integrada, lo que facilita la conexión a redes inalámbricas y la comunicación con otros dispositivos.

- Amplia gama de pines de E/S: El ESP32 proporciona una gran cantidad de pines de entrada/salida, lo que permite conectar una variedad de sensores, actuadores y otros componentes electrónicos.
- Versatilidad: El ESP32 es compatible con el entorno de desarrollo de Arduino, lo que facilita la programación y permite aprovechar una amplia gama de bibliotecas y recursos disponibles.
- Bajo consumo de energía: A pesar de su potencia y rendimiento, el ESP32 está diseñado para ser eficiente en cuanto al consumo de energía, lo que lo hace adecuado para aplicaciones que requieren una larga duración de la batería.

2.4.7.2 Desventajas del ESP32:

- Documentación en evolución: Aunque el ESP32 ha ganado popularidad, algunas partes de su documentación pueden estar en constante evolución o no estar tan consolidadas como en otras plataformas más establecidas.
- Soporte limitado en algunos entornos: Es posible que algunos entornos de desarrollo no ofrezcan un soporte completo para el ESP32, lo que puede dificultar su integración en ciertos proyectos.
- Complejidad para principiantes: Si bien el ESP32 es una plataforma poderosa, su amplio conjunto de características puede resultar abrumador para aquellos que recién comienzan en el mundo de la electrónica y la programación.

En general, el ESP32 es una opción sólida para nuestro trabajo de detección de señas de dedos para la activación de iluminación y puertas, aunque en este trabajo no requerimos la conectividad inalámbrica, pero podría ser integrada en una mejora del mismo como apoyo de control de los familiares a la persona con discapacidad, su potencia de procesamiento y flexibilidad de E/S nos permite crear un código de programación más amplio que el del Arduino Uno y más completo, con un funcionamiento rápido y eficiente. Sin embargo, es importante evaluar cuidadosamente las necesidades específicas de tu proyecto y considerar las ventajas y desventajas antes de elegir el ESP32 como la plataforma adecuada.

CAPITULO III

DESARROLLO DEL TRABAJO DE APLICACIÓN

3.1. DESCRIPCION DEL TRABAJO

En el presente capitulo se realizará el Diseño de prototipo de un sistema Control de iluminación y puertas, basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32, para mejorar la autonomía de las personas con discapacidad **el mismo que** Contará con relés que permitirán el control de encendido de focos; un foco del cuarto, un foco del patio y foco intermitente de aviso de personas, sensor PIR y LDR como sensor de luz para detectar la presencia de personas en la puerta y de esta manera comunicar a la persona con discapacidad la presencia de una persona en su puerta y por consiguiente decidir si abre la puerta o no, el sensor de Luz permitirá la activación del sistema de presencia de personas de manera automática encendiendo la luz del patio para el desplazamiento de la persona visitante y contara con dos servomotores que permitirán la apertura y cierre de la puerta de su habitación de la persona con discapacidad y de la puerta principal para el ingreso del visitante. Aquí iremos explicando cómo funciona el proyecto, cómo trabaja cada componente, el código de programación y el resultado del proyecto.

3.2. DIAGRAMA DE BLOQUES

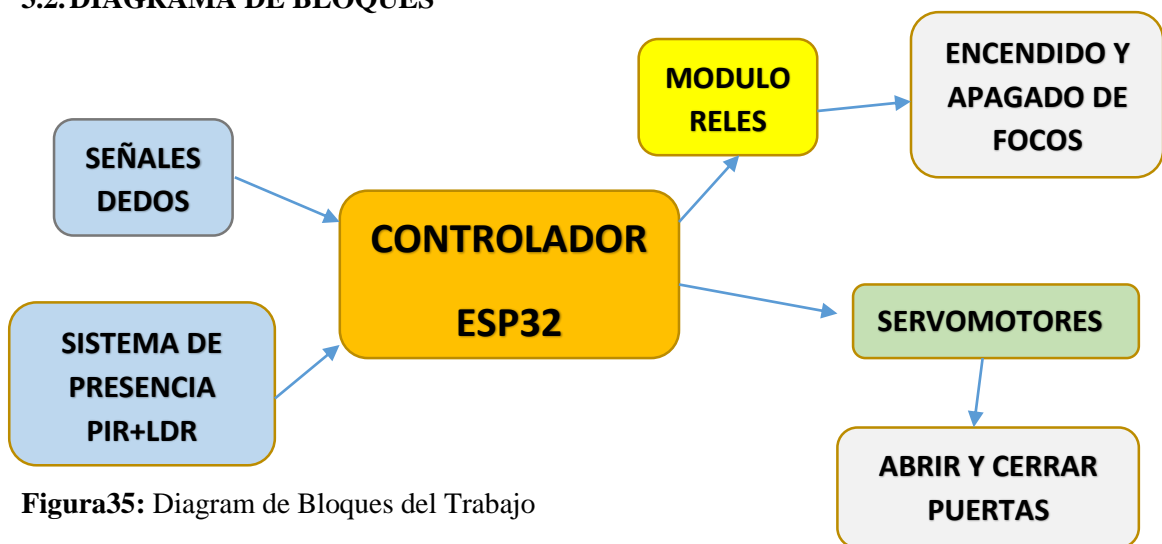


Figura35: Diagram de Bloques del Trabajo

Fuente: Elaboracion Propia

Señales de dedos: Este bloque representa la entrada de señales provenientes de un sensor o dispositivo que detecta las señales de los dedos, como un sensor de flexión o un guante con sensores táctiles. Estas señales indicarán los gestos realizados por los dedos para controlar el sistema en el presente trabajo será controlado por señas de dedos.

Sistema de presencia: Este bloque detecta la presencia de personas utilizando un sensor de movimiento por infrarrojos pasivo (PIR). Durante el día, activa un foco intermitente para indicar la presencia. Durante la noche, también utiliza un sensor de luz para encender dos focos adicionales.

Controlador: Este bloque es responsable de recibir las señales de dedos y procesarlas para determinar las acciones a realizar en el sistema. Puede estar compuesto por un microcontrolador o un dispositivo similar que ejecute el código necesario para interpretar las señales y tomar decisiones, además recibe información de las señales provenientes del sistema de presencia.

Focos: Este bloque representa los focos que se encienden o apagan en función de las señales de dedos recibidas. Pueden ser focos individuales o agrupados en un circuito de iluminación.

Puertas: Este bloque representa las puertas que se abren o cierran en función de las señales de dedos recibidas. Pueden ser puertas físicas controladas por motores o relés.

3.3. REPRESENTACIÓN DE CIRCUITO

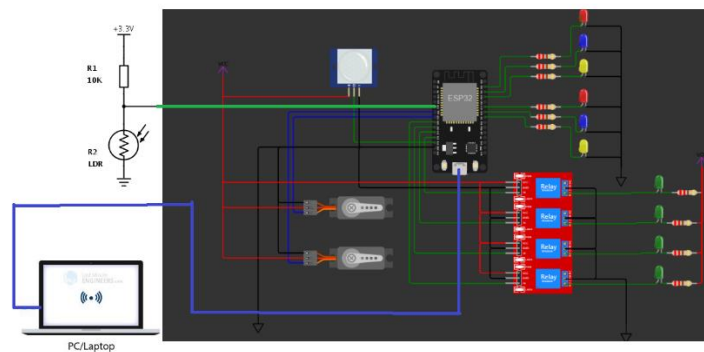


Figura36: Representación Gráfica del trabajo

Fuente: Elaboración Propia

3.4. PROGRAMACION DEL TRABAJO

Se presenta la programación utilizada para el funcionamiento sistema de control e iluminación para personas con discapacidad basado en señas específicas de los dedos

3.4.1. Arduino IDE

```
#include <ESP32Servo.h>

//NOMBRES A CADA PIN DE ENTRADA
#define LDR 35 //El LDR esta conectador en el pin A7
#define PIR 34 //El PIR esta conetado en el pin 34

//NOMBRES DE CADA PIN DE SALIDA
#define CUARTO 26 //foco del cuarto
#define PATIO 27 //foco del patio
#define PASILLO 14 //foco del pasillo
#define LAMPARA 25 //FOFO DE LA LAMPARA
#define PUERTA 13 //puerta del cuarto
#define PRINCIPAL 32 //puerta principal

//LUCES DE ACCIONES DE LAS PUERTAS
#define AZUL_CUARTO 4 //indicador de funcionamiento de la puerta CUARTO
#define VERDE_CUARTO 18 //indicador de puerta del CUARTO abierto
#define ROJO_CUARTO 19 //indicador de inicio de cierre de puerta CUARTO
#define AZUL_PRINCIPAL 21 //indicador de funcionamiento de la puerta PRINCIPAL
#define VERDE_PRINCIPAL 17 //indicador de puerta PRINCIÁL abierto
#define ROJO_PRINCIPAL 16 //indicador de inicio de cierre de puerta PRINCIÁL

//VARIABLES Y LIBRERIAS DE SERVOMOTOR
Servo servo1; //instancia de la clase Servo llamada "servo1" necesario para poder usar la librería Servo
Servo servo2; //instancia de la clase Servo llamada "servo2" necesario para poder usar la librería Servo

//declaracion de variables a ser utilizadas en el programa para almacenar datos o inicializacion de variables

char letra; //la variable letra nos permitira almacenar el dato proveniente de pyhton

int ESTADO = 0; //valor inicial del PIR
bool alarma_activado = false; //NOS PERMITIRA ACTIVAR O DESACTIVAR LA ALARMA
```

```

int luz = 800; //valor para garantizar el no funcionamiento
int valor_sensor = 4095; //valor inicial para garantizar el no funcionamiento
int valor_limite = 150; //valor que nos permitira comparar la luminosidad y garantizar el funcionamiento

//valores maximos de las puertas o cerrado y 90 abierto
int anguloAbierto = 90; // Ángulo de apertura de la puerta principal
int anguloCerrado = 0; // Ángulo de cierre de la puerta principal
int anguloAbierto2 = 90; // Ángulo de apertura de la puerta del cuarto
int anguloCerrado2 = 0; // Ángulo de cierre de la puerta del cuarto

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Uso de millis() como delay
bool bucle_activado = false; //estado inicial de la intermitencia
unsigned long actual = 0; //valor para realizar el delay con millis()
unsigned long anterior = 0; //valor para realizar el delay con millis()
bool estado1 = HIGH; //estado inicial de la luz intermitente
int i = 0; //nos permitira seguir con la intermitencia

//uso de millis() control de puerta principal
bool puerta = false; //estado inicial de la puerta cerrada
unsigned long actual1 = 0; //valor para realizar el delay con millis()
unsigned long anterior1 = 0; //valor para realizar el delay con millis()
unsigned long tiempoApertura = 5000; // Tiempo en milisegundos para abrir la puerta
unsigned long tiempoCierre = 5000; // Tiempo en milisegundos para cerrar la puerta
bool ejecucionCompletada=true; //permitira saber si la puerta se abrio y se cerro

//uso de millis() control de puerta del cuarto
bool puerta2 = false; //estado inicial de la puerta cerrada
unsigned long actual2 = 0; //valor para realizar el delay con millis()
unsigned long anterior2 = 0; //valor para realizar el delay con millis()
unsigned long tiempoApertura2 = 5000; // Tiempo en milisegundos para abrir la puerta
unsigned long tiempoCierre2 = 5000; // Tiempo en milisegundos para cerrar la puerta
bool ejecucionCompletada2=true; //permitira saber si la puerta se abrio y se cerro

bool espera = false; //no se inicia el tiempo para apagar el foco del patio
unsigned long tiempoEspera = 5000; // Tiempo en milisegundos para apagar el foco del patio
bool espera1 = false; //no se inicia el tiempo para apagar el foco del pasillo
unsigned long tiempoEspera1 = 5000; // Tiempo en milisegundos para aogar el foco del pasillo
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

```

//INICIALIZACION
void setup()
{
  Serial.begin(9600); //comunicacion serial para la lectura de datos de la vision artificial

  //configurar pines como entrada o salida

  pinMode(LDR,INPUT); //declaramos como entrada de lectura del LDR
  pinMode(PIR,INPUT); //declaramos como entrada de lectura del PIR

  pinMode(CUARTO, OUTPUT); //Configura como salida el pin donde esta conectado el foco del cuarto
  pinMode(PATIO, OUTPUT); //Configura como salida el pin donde esta conectado el foco del patio
  pinMode(PASILLO, OUTPUT); //configura como salida el pin donde esta conectado la pasillo
  pinMode(LAMPARA, OUTPUT); //configura como salida el pin donde esta conectado la LAMPARA

  pinMode(AZUL_CUARTO, OUTPUT); //Configura como salida indicador de funcionamiento de la puerta
CUARTO
  pinMode(VERDE_CUARTO, OUTPUT); //Configura como salida indicador de puerta del CUARTO abierto
  pinMode(ROJO_CUARTO, OUTPUT); //Configura como salida indicador de inicio de cierre de puerta CUARTO
  pinMode(AZUL_PRINCIPAL, OUTPUT); //Configura como salida indicador de funcionamiento de la puerta
PRINCIPAL
  pinMode(VERDE_PRINCIPAL, OUTPUT); //Configura como salida indicador de puerta PRINCIPAL abierto
  pinMode(ROJO_PRINCIPAL, OUTPUT); //Configura como salida indicador de inicio de cierre de puerta PRINCIPAL

  digitalWrite(CUARTO, HIGH); //foco del cuarto APAGADO
  digitalWrite(PATIO, HIGH); //foco del patio APAGADO
  digitalWrite(PASILLO, HIGH); //foco del pasillo APAGADO
  digitalWrite(LAMPARA, HIGH); //foco LAMPARA APAGADO

  digitalWrite(AZUL_CUARTO, LOW); //LED indicador de funcionamiento de la puerta CUARTO APAGADO
  digitalWrite(VERDE_CUARTO, LOW); //LED indicador de la puerta abierta del CUARTO APAGADO
  digitalWrite(ROJO_CUARTO, LOW); //LED indicador de inicio de cierre de la puerta del CUARTO APAGADO
  digitalWrite(AZUL_PRINCIPAL, LOW); //LED indicador de funcionamiento de la puerta CUARTO APAGADO
  digitalWrite(VERDE_PRINCIPAL, LOW); //LED indicador de la puerta abierta del CUARTO APAGADO
  digitalWrite(ROJO_PRINCIPAL, LOW); //LED indicador de inicio de cierre de la puerta del CUARTO APAGADO

  servo1.attach(PUERTA); //configura como salida el pin donde esta conectado el motor de control de la puerta del
cuarto

```

```
servo2.attach(PRINCIPAL); //configura como salida el pin donde esta conectado el motor de control de la puerta del principal
```

```
servo1.write(anguloCerrado); //inicializamos la puerta del cuarto cerrado  
servo2.write(anguloCerrado); //inicializamos la puerta principal cerrada  
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
void loop() //EJECUCION DEL PROGRAMA PRINCIPAL
```

```
{
```

```
    //LEER DATOS DEL PUERTO SERIAL
```

```
    if(Serial.available(>0) //verifica si hay datos disponibles para leer en el puerto serial
```

```
    {
```

```
        letra=Serial.read();//leer y guardar EL VALOR LEIDO EN LA VARIABLE LETRA
```

```
    }
```

```
//////////////////////////////////////////////////////////////////FUNCIONAMIENTO PRINCIPAL DEL CONTROL DE FOCOS
```

```
    if(letra=='A'){
```

```
        digitalWrite(CUARTO, LOW);//foco del cuarto encendido
```

```
    }
```

```
    if(letra=='B'){
```

```
        digitalWrite(CUARTO, HIGH);//foco del cuarto apagado
```

```
    }
```

```
    if(letra=='C'){
```

```
        digitalWrite(PATIO, LOW); //foco del patio encendido
```

```
    }
```

```
    if(letra=='D'){
```

```
        digitalWrite(PATIO, HIGH);//foco del patio apagado
```

```
    }
```

```
    if(letra=='E'){
```

```
        digitalWrite(LAMPARA, LOW); //foco de la LAMPARA encendido
```

```
    }
```

```
    if(letra=='F'){
```

```
        digitalWrite(LAMPARA, HIGH); //foco de la LAMPARA encendido
```

```
    }
```

```
//////////////////////////////////////////////////////////////////Funcionamiento principal para Encender la Alarma de Presencia
```

```
//////////////////////////////////////////////////////////////////SENSOR DE LUZ
```

```
valor_sensor = analogRead(LDR); //leer el valor analogico del LDR
```

```
luz = (3.3 * valor_sensor * 100.0)/4095.0; //convierte la lectura analógica del sensor en un valor de luz en lux
```

```
////////ACTIVACION O DESACTIVACION DE LA ALARMA
```

```
if(letra == '0')
{
    alarma_activado = true; //activar la alarma
    letra='w';           //salimos del bucle infinito para continuar con la apertura y cierre de la puerta
}else{
    if(letra == '1')
    {
        ESTADO = LOW;           //estado de lectura del PIR sin presencia
        bucle_activado = false; //desactivar el bucle
        digitalWrite(LAMPARA, HIGH); //apagar LA LAMPARA
        digitalWrite(ROJO_CUARTO, LOW); //apagar la luz DE PRESENCIA
        alarma_activado = false; //desactivar la alarma
    }
}
```

```
////////FUNCIONAMIENTO DE LA LUZ INTERMITENTE Y LUZ DEL PATIO
```

```
if(alarma_activado)
{
    ESTADO=digitalRead(PIR); //leer en que estado esta el PIR
    if (luz > valor_limite && ESTADO==HIGH) //es dis y se detecto presencia de una persona
    {
        i=0;           //condicion para seguir con la intermitencia
        bucle_activado = true; //activar el bucle para seguir la intermitencia
        alarma_activado = true; //alarma aun sigue activada
        if(letra == '1') //permite desactivar la alarma
        {
            alarma_activado=false; //desactivar la alarma
            bucle_activado = false; //apagar la alerta
        }
    }
}
```

```
if(luz <= valor_limite && ESTADO==HIGH)//noche; // Si se detecta presencia, el bucle de parpadeo
```

continúa

```
{
    i=0;           //condicion para seguir con la intermitencia
    bucle_activado = true; //continuar con el bucle de intermitencia
    alarma_activado = true; //alarma aun sigue activada
    digitalWrite(LAMPARA, LOW); //ENCENDER LAMPARA
```

```

//digitalWrite(ROJO_PRINCIPAL, HIGH); //encender la luz del patio mientras dure la
intermitencia
if(letra == '1') //permite desactivar la alarma
{
    alarma_activado=false; //desactivar la alarma
    bucle_activado = false; //apagar la alerta
}
}
}

////////////////////////encendido del foco intermitencia
actual = millis();//iniciar en contador de tiempo
// Verificar si ha pasado el tiempo de parpadeo y el bucle está activado
if (actual - anterior >= 500 && bucle_activado)//durara el cambio medio segundo cada
intermitencia
{
    anterior = actual; // Actualizar el tiempo anterior
    if (i < 2) { //se repetira 10 veces el encendido y apagado de la luz intermitente
        estado1 = !estado1; // Cambiar el estado del foco intermitente
        digitalWrite(ROJO_CUARTO, estado1);
        i++; // Incrementar el contador de parpadeos
    } else {
        bucle_activado = false; // Desactivar el bucle después de diez parpadeos
        digitalWrite(ROJO_CUARTO, LOW); //apagar la luz del patio
        digitalWrite(LAMPARA, HIGH); //apagar LA LAMPARA
    }
}

////////////////////////FUNCIONAMIENTO PRINCIPAL DE LAS PUERTAS
////////////////////////CONTROL DE LA PUERTA PRINCIPAL
if(letra=='2')//PERMITE REALIZAR LA APERTURA Y CIERRE DE LA PUERTA
{
    digitalWrite(AZUL_CUARTO, HIGH); //LED indicador de inicio de apertura de la puerta CUARTO
    ejecucionCompletada=false; //VAMOS A REALIZAR EL CONTROL AUTOMATICO DE LA PUERTA
    puerta = false; //REALIZAR LA APERTURA DE LA PUERTA
    anterior1=millis(); //CONTADOR QUE PERMITIRA INICIAR EL CONTEO DE TIEMPO DE
EJECUCION
    letra='w'; //salimos del bucle infinito para continuar con la apertura y cierre de la puerta
}
if(letra=='3')//PERMITE REALIZAR LA APERTURA Y CIERRE DE LA PUERTA
{

```

```

digitalWrite(AZUL_CUARTO, LOW);
ejecucionCompletada=false; //VAMOS A REALIZAR EL CONTROL AUTOMATICO DE LA PUERTA
puerta = true;           //REALIZAR CIERRE DE LA PUERTA
letra='w';               //salimos del bucle infinito para continuar con la apertura y cierre de la puerta
}
////////////////////

////////////////////CONTROL DE LA PUERTA DEL CUARTO
if(letra=='4')// apertura y cierre de la puerta del cuarto
{
digitalWrite(PATIO, LOW);           //encender la luz del patio mientras este en funcionamiento la puerta
digitalWrite(AZUL_PRINCIPAL, HIGH); //LED indicador de inicio de la apertura de la puerta PRINCIPAL
ejecucionCompletada2=false;        //INDICA QUE SE DESEA ABRIR Y CERRAR LA PUERTA
puerta2 = false;                   //REALIZAR LA APERTURA DE LA PUERTA
anterior2=millis();                //CONTADOR QUE PERMITIRA INICIAR EL CONTEO DE TIEMPO DE
EJECUCION
letra='w';                          //salimos del bucle infinito para continuar con la apertura y cierre de la puerta
}
if(letra=='5')// PERMITE CERRAR LA PUERTA
{
digitalWrite(AZUL_PRINCIPAL, LOW); //LED indicador de inicio de la apertura de la puerta PRINCIPAL
ejecucionCompletada2=false;        //NOS PERMITIRA CONTROLAR QUE SE TERMINO EL PROCESO
puerta2 = true;                   //REALIZAR EL CIERRE DE LA PUERTA
letra='w';                          //salimos del bucle infinito para continuar con la apertura y cierre de la puerta
}
////////////////////

if(!ejecucionCompletada) //AMBAS CONDICIONES DEBEN CUMPLIRSE PARA CONTINUAR
{
actual1=millis(); //INICIAMOS EL CONTADOR DE TIEMPO

if (!puerta)//se debe de cumplir el tiempo y la condicion para continuar
{
int anguloActual = map(actual1 - anterior1, 0, tiempoApertura, anguloCerrado,
anguloAbierto);//compara el tiempo para abrir luego de 5 segundos
servo1.write(anguloActual); //movemos el servomotor para abrir
if (anguloActual >= anguloAbierto) //si el angulo de apertuta es 90 grados
{
digitalWrite(VERDE_CUARTO, HIGH); //LED indicador de la puerta esta abierta CUARTO
puerta = true; // Actualizar el estado de la puerta para luego cerrarla
}
}
}

```



```

        anterior1 = actual1;          // Reiniciar el tiempo de referencia para continuar con el cierre
        digitalWrite(AZUL_CUARTO, LOW); //LED indicador Se inicio el cierre de la puerta del
CUARTO ENCENDIDO
    }
}
// Cerrar la puerta después de tiempoCierre
if (puerta) //se debe de cumplir el tiempo y la condicion para continuar
{
    int anguloActual = map(actual1 - anterior1, 0, tiempoCierre, anguloAbierto, anguloCerrado); //compara
el tiempo para abrir luego de 5 segundos
    servo1.write(anguloActual); //movemos el servomotor para cerrar
    if (anguloActual <= anguloCerrado) //si el angulo de apertura es 0 grados
    {
        puerta = false; // Actualizar el estado de la puerta para luego cerrarlo
        anterior1 = actual1; // Reiniciar el tiempo de referencia
        ejecucionCompletada=true; // la puerta termino su ciclo de cierre y apertura
        digitalWrite(VERDE_CUARTO, LOW); //LED indicador LA PUERTA SE CERRO
    }
}
}
////////////////////////////////////

////////////////////////////////////

if(!ejecucionCompletada2)//INICIAMOS LA APERTURA Y RESPECTIVO CIERRE DE LA PUERTA
{
    actual2=millis();//INICIAMOS EL CONTADOR DE TIEMPO

    if (!puerta2) //CONDICIN PARA ABRIR LA PUERTA
    {
        // Cerrar la puerta después de tiempoCierre
        int anguloActual2 = map(actual2 - anterior2, 0, tiempoApertura2, anguloCerrado2,
anguloAbierto2); //compara el tiempo para abrir luego de 5 segundos
        servo2.write(anguloActual2); //movemos el servomotor para abrirlo
        if (anguloActual2 >= anguloAbierto2) //si el angulo de apertura es 90 grados
        {
            digitalWrite(VERDE_PRINCIPAL, HIGH); //LED indicador de la puerta PRINCIPAL abierta
            puerta2 = true; //iniciamos el cierre de la puerta PRINCIPAL
            anterior2 = actual2; //reiniciamos en tiempo de referencia

```

```

        digitalWrite(AZUL_PRINCIPAL, LOW); //LED indicador de INICIO de cierre de la puerta
PRINCIPAL
        //digitalWrite(ROJO_PRINCIPAL, HIGH); //LED indicador de inicio de cierre de la puerta
PRINCIPAL
    }
}
if (puerta2) //se debe de cumplir el tiempo y la condicion para continuar
{
    int anguloActual3 = map(actual2 - anterior2, 0, tiempoCierre2, anguloAbierto2,
anguloCerrado2); //compara el tiempo para abrir luego de 5 segundos
    servo2.write(anguloActual3); //movemos el servomotor para cerrar
    if (anguloActual3 <= anguloCerrado2) //si el angulo de apertura es 0 grados
    {
        puerta2 = false; // Actualizar el estado de la puerta ya se cerro
        anterior2 = actual2; // Reiniciar el tiempo de referencia
        ejecucionCompletada2=true; // la puerta termino su ciclo de cierre y apertura
        espera=true; //EMPEZAR EL CONTEO DE TIEMPO PARA APAGAR LA LUS
DEL PATIO
        digitalWrite(VERDE_PRINCIPAL, LOW); //LED indicador LA PUERTA SE CERRO
        digitalWrite(ROJO_PRINCIPAL, HIGH); //LED indicador LA PUERTA SE CERRO
        digitalWrite(PASILLO, LOW); //apagar la luz del patio
    }
}
}
////////////////////////////////////

//////////tiempo de espera para el apagado de las luces.
actual2=millis();
if(espera){
    if(actual2 - anterior2>=tiempoEspera)
    {
        digitalWrite(PATIO, HIGH); //apagar la luz del patio
        digitalWrite(ROJO_PRINCIPAL, LOW); //LED indicador LA PUERTA SE CERRO Y LUZ DE PATIO
APAGADO
        espera=false; //terminar el tiempo de espera
        anterior2 = actual2; //Reiniciar el tiempo de referencia
        espera1=true; //PASAMOS TIEMPO DE APAGADO DEL FOCO DEL PATIO
    }
}
}

```

```

if(espera1)
{
    if(actual2 - anterior2>=tiempoEspera1)
    {
        puerta2 = false;           //Actualizar el estado de la puerta ya se cerro
        espera1=false;             //terminar el tiempo de espera1
        digitalWrite(PASILLO, HIGH); //apagar la luz del pasillo
    }
}
}
}

```

3.4.2. Python

```

import cv2 #libreria para el procesamiento del video
import mediapipe as mp #libreria para el seguimiento de manos
import numpy as np #libreria para realizar calculos numericos
from math import acos, degrees #util para calcular el arco coseno y convertir angulo en radianes

#LIBRERIAS PARA LA COMUNICACION SERIAL
import serial #libreria para la comunicacion serial con el ESP32
import time #para la comunicacion serial y retardos

R=0 # NOS PERMITIRA RESTINGIR EL USO Y EL ERROR DE LECTURA DE DATOS
t=texto=texto1=texto2=texto3=texto4=texto5=""#nos permitira utilizar para escribir texto en pantalla
texto6=texto7=texto8=texto9=texto10=texto11=""#nos permitira utilizar para escribir texto en pantalla
m = [2, 2, 2, 2, 2, 2] #define el tamaño del borde de los cuadros que se muestran en pantalla

esp = serial.Serial('COM3', 9600, timeout =1) #definicion de los parametros de la comunicacion serial
espera un segundo antes de recibir datos
time.sleep(2) #tiempo de espera 2 segundos

#calculamos en centro de la palma
def palm_centroid(coordinates_list):
    coordinates = np.array(coordinates_list) #convertimos en un arreglo
    centroid = np.mean(coordinates, axis=0) #eje 0 como media para calcular la media de las coordenadas

```

```

centroid = int(centroid[0]), int(centroid[1]) #coordenadas en enteros
return centroid #resultado de la funcion del centroide calculado

mp_drawing = mp.solutions.drawing_utils #funcion para dibujar o visualizar el video
mp_drawing_styles = mp.solutions.drawing_styles #permite definir colores y espesores de lineas
mp_hands = mp.solutions.hands #modelos y metodos para el seguimiento de manos en el video

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW) #TOMAMOS LA CAPTURA CON 0 DE LA CAMARA DE LA
COMPUTADORA CON 1 DEL PUERTO USB

#DEFINIR QUE DEDOS ESTAN LEVANTADOS

ONCUARTO=np.array([True,True,True,True,False])#ENCENDER EL FOCO DEL CUARTO
OFFCUARTO=np.array([True,True,True,False,True])#APAGAR EL FOCO DEL CUARTO
ONPATIO=np.array([True,True,True,False,False])#ENCENDER EL FOCO DEL PATIO
OFFPATIO=np.array([True,False,True,True,True])#APAGAR EL FOCO DEL PATIO
ONLAMPARA=np.array([True,True,False,True,True])#ENCENDER LA LAMPARA
OFFLAMPARA=np.array([False,False,False,False,False])#APAGAR LA LAMPARA
ONALARMA=np.array([False,True,True,True,False])#ENCENDER LA ALARMA
OFFALARMA=np.array([False,True,True,True,True])#APAGAR LA ALARMA
OPENPUERTA=np.array([True,False,False,False,False])#ABRIR LA PUERTA DEL CUARTO
CLOSEPUERTA=np.array([True,False,False,False,True])#CERRAR LA PUERTA DEL CUARTO
OPENPRINCIPAL=np.array([True,True,False,False,False])#ABRIR LA PUERTA PRINCIPAL
CLOSEPRINCIPAL=np.array([True,True,False,False,True])#CERRAR LA PUERTA PRINCIPAL
MANOABIERTA=np.array([True,True,True,True,True])#LA MANO ESTA ABIERTA

# Pulgar
thumb_points = [1, 2, 4]#PUNTOS DE REFERENCIA DEL PULGAR

# Índice, medio, anular y meñique
palm_points = [0, 1, 2, 5, 9, 13, 17]#PUNTOS DE REFERENCIA DE LA PALMA
fingertips_points = [8, 12, 16, 20]#puntos de las yemas de los dedos
#finger_base_points =[7, 11, 15, 19]#puntos de base de referencia de los dedos

```

```

finger_base_points = [6, 10, 14, 18] #puntos de base de referencia de los dedos

# Colores GBR
VERDE = (48, 255, 48) #DEFINE EL COLOR VERDE CON GBR
AZUL = (192, 101, 21) #DEFINE EL COLOR AZUL CON GBR
AMARILLO = (0, 204, 255) #DEFINE EL COLOR AMARRILLO CON GBR
MORADO = (128, 64, 128) #DEFINE EL COLOR MORADO CON GBR
ROSADO = (180, 229, 255) #DEFINE EL COLOR ROSADO CON GBR
NARANJA = (0, 69, 255) #DEFINE EL COLOR NARANJA CON GBR
ROJO = (0, 0, 255) #DEFINE EL COLOR ROJO CON GBR
NEGRO = (0, 0, 0) #DEFINE EL COLOR NEGRO CON GBR

with mp_hands.Hands(
    model_complexity=1, #presicion del seguimiento de manos sin dejar de lado la velocidad
    max_num_hands=1, #número máximo de manos a detectar
    min_detection_confidence=0.5, #confianza de detección de manos
    min_tracking_confidence=0.5) as hands: #confianza para el seguimiento de manos

    while True:
        ret, frame = cap.read() #leemos el video se le asigna una trama
        if ret == False: #comprueba si fallo la lectura
            break #se interrumpe si sale del bucle
        frame = cv2.flip(frame, 1) #reflejar el video
        height, width, _ = frame.shape #obtener las dimensiones de altura y ancho del fotograma
        #print("Dimensiones del fotograma: {} x {}".format(width, height))
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) #cambiamos a RGB para el trabajo con
mediapipe

        results = hands.process(frame_rgb) #resultado del seguimiento de manos
        fingers_counter = "_" #contador de dedos inicializado

        if results.multi_hand_landmarks:
            coordinates_thumb = [] #iniciamos lista vacio para recibir puntos de referencia del pulgar
            coordinates_palm = [] #iniciamos lista vacio para recibir puntos de referencia de la palma

```

```

coordinates_ft = [] #iniciamos lista vacio para recibir puntos de referencia de los dedos
coordinates_fb = [] #iniciamos lista vacio para recibir puntos de referencia de la base de los
dedos
for hand_landmarks in results.multi_hand_landmarks:
    for index in thumb_points: #lista de coordenadas de pulgar
        x = int(hand_landmarks.landmark[index].x * width) #obtenemos la posicion relativa en el
eje x
        y = int(hand_landmarks.landmark[index].y * height) #obtenemos la posicion relativa en el
eje y
        coordinates_thumb.append([x, y]) #coodenadas de cada punto de referencia del pulgar

    for index in palm_points: #lista de coordenadas de la palma
        x = int(hand_landmarks.landmark[index].x * width) #obtenemos la posicion relativa en el
eje x
        y = int(hand_landmarks.landmark[index].y * height) #obtenemos la posicion relativa en el
eje y
        coordinates_palm.append([x, y]) #coodenadas de cada punto de referencia de la palma

    for index in fingertips_points: #lista de coordenadas de los dedos
        x = int(hand_landmarks.landmark[index].x * width) #obtenemos la posicion relativa en el
eje x
        y = int(hand_landmarks.landmark[index].y * height) #obtenemos la posicion relativa en el
eje y
        coordinates_ft.append([x, y]) #coodenadas de cada punto de referencia de los dedos

    for index in finger_base_points: #lista de coordenadas de la base de los dedos
        x = int(hand_landmarks.landmark[index].x * width) #obtenemos la posicion relativa en el
eje x
        y = int(hand_landmarks.landmark[index].y * height) #obtenemos la posicion relativa en el
eje y
        coordinates_fb.append([x, y]) #coodenadas de cada punto de referencia de la base de los
dedos

#####

```

```

# Pulgar
p1 = np.array(coordinates_thumb[0]) #arreglo del primer punto de referencia del pulgar
p2 = np.array(coordinates_thumb[1]) #arreglo del segundo punto de referencia del pulgar
p3 = np.array(coordinates_thumb[2]) #arreglo del tercer punto de referencia del pulgar

l1 = np.linalg.norm(p2 - p3) #calcula la longitud entre p2 y p3
l2 = np.linalg.norm(p1 - p3) #calcula la longitud entre p2 y p3
l3 = np.linalg.norm(p1 - p2) #calcula la longitud entre p2 y p3

# Calcular el ángulo
angle = np.degrees(np.arccos((l1**2 + l3**2 - l2**2) / (2 * l1 * l3)))
#calculo del angulo y convertir el valor en grados
thumb_finger = np.array(False) #indica que el pulgar no esta extendido
if angle > 150: #verifica el angulo formado por el pulgar
    thumb_finger = np.array(True) #el pulgar esta extendido

#####
# Índice, medio, anular y meñique
nx, ny = palm_centroid(coordinates_palm) #calcula el centro de la palma
cv2.circle(frame, (nx, ny), 3, (0, 255, 0), 2) #dibuja un punto en el centro de la palma
coordinates_centroid = np.array([nx, ny]) #arreglo de coordenadas de la palma
coordinates_ft = np.array(coordinates_ft) #arreglo de coordenadas de referencia de los
dedos sin el pulgar
coordinates_fb = np.array(coordinates_fb) #arreglo de coordenadas de la base de los dedos

# Distancias
d_centrid_ft = np.linalg.norm(coordinates_centroid - coordinates_ft, axis=1) #distancia entre
el centro de la palma y los puntos de referencia de los dedos
d_centrid_fb = np.linalg.norm(coordinates_centroid - coordinates_fb, axis=1) #distancia entre
la base y los puntos de referencia de los dedos
dif = d_centrid_ft - d_centrid_fb #resultado de la diferencia de las distancias
fingers = dif > 0 # comparara para colocar True al dedo levantado y False si no
fingers = np.append(thumb_finger, fingers) #nos permite saber que dedo esta levantado

```

```

mp_drawing.draw_landmarks( #dibuja lineas de conexion en los dedos
frame, #donde se dibujarn las lineas
hand_landmarks, #representa los puntos claves en la mano
mp_hands.HAND_CONNECTIONS, #conexiones para las manos
mp_drawing_styles.get_default_hand_landmarks_style(), #permite dibujar objetos
mp_drawing_styles.get_default_hand_connections_style()) #estilo de dibujo de la conexion
de la mano

```

```
#####
```

```
# CODIGO DE FUNCIONAMIENTO DE INTERFAZ Y COMUNICACION SERIAL
```

```
if (fingers == MANOABIERTA).all() #CONDICION PARA PODER CONTROLAR UN DISPOSITIVO
```

```
R = 1 #VALOR A COMPARAR PARA RESPONDER UNA PETICION
```

```
if R == 1 and (fingers == ONCUARTO).all(): #ENCENDER EL FOCO DEL CUARTO
```

```
esp.write(b'A') #mandamos un caracter al ESP32
```

```
texto1="" #BORRAR DE PANTALLA EL TEXTO1
```

```
texto="ON" #MOSTRAR EN PANTALLA ON ENCENDIDO EL CUARTO
```

```
m[0]=-1 # PINTAR DE COLOR RASADO EL CUADRO COMPLETO
```

```
R = 0
```

```
if R==1 and (fingers == OFFCUARTO).all() #APAGAR EL FOCO DEL CUARTO
```

```
esp.write(b'B')#mandamos un caracter al ESP32
```

```
texto=""#BORRAR DE PANTALLA
```

```
texto1="OFF"#MOSTRAR EN PANTALLA
```

```
m[0]=2# BORRAR EL CUADRO COMPLETO
```

```
R = 0
```

```
if R==1 and (fingers == ONPATIO).all():#ENCENDER EL FOCO DEL PATIO
```

```
esp.write(b'C')#mandamos un caracter al ESP32
```

```
texto3=""#BORRAR DE PANTALLA
```

```
texto2="ON"#MOSTRAR EN PANTALLA
```

```
m[1]=-1#PINTAR EL CUADRO COMPLETO
```

```
R = 0
```

```
if R==1 and (fingers == OFFPATIO).all() #APAGAR EL FOCO DEL PATIO
```

```
esp.write(b'D')#mandamos un caracter al ESP32
```

```
texto2=""#BORRAR DE PANTALLA
```



```

texto3="OFF"#MOSTRAR EN PANTALLA
m[1]=2# BORRAR EL CUADRO COMPLETO
R = 0
if R==1 and (fingers == ONLAMPARA).all():#ENCENDER LA LAMPARA
    esp.write(b'E')#mandamos un caracter al ESP32
    texto5=""#BORRAR DE PANTALLA
    texto4="ON"#MOSTRAR EN PANTALLA
    m[2]=-1#PINTAR EL CUADRO COMPLETO
    R = 0
if R==1 and (fingers == OFFLAMPARA).all() #APAGAR LA LAMPARA
    esp.write(b'F')#mandamos un caracter al ESP32
    texto4=""#BORRAR DE PANTALLA
    texto5="OFF"#MOSTRAR EN PANTALLA
    m[2]=2# BORRAR EL CUADRO COMPLETO
    R = 0
if R==1 and (fingers == ONALARMA).all():#ENCENDER LA ALARMA
    esp.write(b'O')#mandamos un caracter al ESP32
    texto7=""#BORRAR DE PANTALLA
    texto6="ON"#MOSTRAR EN PANTALLA
    m[3]=-1#PINTAR EL CUADRO COMPLETO
    R = 0
if R==1 and (fingers == OFFALARMA).all() #APAGAR LA ALARMA
    esp.write(b'1')#mandamos un caracter al ESP32
    texto6=""#BORRAR DE PANTALLA
    texto7="OFF"#MOSTRAR EN PANTALLA
    m[3]=2# BORRAR EL CUADRO COMPLETO
    R = 0
if R==1 and (fingers == OPENPUERTA).all():#ABRIR LA PUERTA DEL CUARTO
    esp.write(b'2')#mandamos un caracter al ESP32
    texto9=""#BORRAR DE PANTALLA
    texto8="ON"#MOSTRAR EN PANTALLA
    m[4]=-1#PINTAR EL CUADRO COMPLETO
    R = 0

```

```

if R==1 and (fingers == CLOSEPUERTA).all():#CERRAR LA PUERTA DEL CUARTO
    esp.write(b'3')
    texto8=""
    texto9="OFF"
    m[4]=2
    R = 0
if R==1 and (fingers == OPENPRINCIPAL).all():#ABRIR LA PUERTA PRINCIPAL
    esp.write(b'4')#mandamos un caracter al ESP32
    texto11=""#BORRAR DE PANTALLA
    texto10="ON"#MOSTRAR EN PANTALLA
    m[5]=-1#PINTAR EL CUADRO COMPLETO
    R = 0
if R==1 and (fingers == CLOSEPRINCIPAL).all():#CERRAR LA PUERTA PRINCIPAL
    esp.write(b'5')
    texto10=""
    texto11="OFF"
    m[5]=2
    R = 0
else:
    print(R)
break

```

Visualización en pantalla de encendido o apagado

```

if R==1:
    t="SI"
elif R==0:
    t="NO"

```

#INDICACION DE QUE SE PUEDE ACTIVAR O NO ALGUN FOCO O PUERTA O ALARMA

```

cv2.rectangle(frame, (0, 340), (80, 420), (255, 255, 51), -1)
cv2.putText(frame, t,(10, 400), 1, 3 , NEGRO, 5, cv2.LINE_AA)
cv2.putText(frame, "PUEDE", (10, 440), 1, 1, ROJO, 2,cv2.LINE_AA)
cv2.putText(frame, "ACTIVAR", (5, 460), 1, 1, ROJO, 2,cv2.LINE_AA)

```

#indicacion de FOCO DE CUARTO dibuja un cuadro de tamaño, grosor, color y estilo de letra

```

cv2.rectangle(frame, (100, 370), (150, 420), ROSADO, m[0])

```

```

cv2.putText(frame, texto, (110, 400), 1, 1, NEGRO, 2,cv2.LINE_AA)
cv2.putText(frame, texto1, (110, 400), 1, 1, ROJO, 2,cv2.LINE_AA)
cv2.putText(frame, "FOCO", (100, 440), 1, 1, ROSADO, 2,cv2.LINE_AA)
cv2.putText(frame, "CUARTO", (80, 460), 1, 1, ROSADO, 2,cv2.LINE_AA)
#indicacion de FOCO DE PATIO dibuja un cuadro de tamaño, grosor, colro y estilode letra
cv2.rectangle(frame, (160, 370), (210, 420), MORADO, m[1])
cv2.putText(frame, texto2, (170, 400), 1, 1, NEGRO, 2,cv2.LINE_AA)
cv2.putText(frame, texto3, (170, 400), 1, 1, ROJO, 2,cv2.LINE_AA)
cv2.putText(frame, "FOCO", (160, 440), 1, 1, MORADO, 2,cv2.LINE_AA)
cv2.putText(frame, "PATIO", (160, 460), 1, 1, MORADO, 2,cv2.LINE_AA)
#indicacion de PASILLO dibuja un cuadro de tamaño, grosor, colro y estilode letra
cv2.rectangle(frame, (220, 370), (270, 420), AMARILLO, m[2])
cv2.putText(frame, texto4, (230, 400), 1, 1, NEGRO, 2,cv2.LINE_AA)
cv2.putText(frame, texto5, (230, 400), 1, 1, ROJO, 2,cv2.LINE_AA)
cv2.putText(frame, "LAMPARA", (210, 440), 1, 1, AMARILLO, 2,cv2.LINE_AA)
#indicacion de ALARMA dibuja un cuadro de tamaño, grosor, colro y estilode letra
cv2.rectangle(frame, (280, 370), (330, 420), VERDE, m[3])
cv2.putText(frame, texto6, (290, 400), 1, 1, NEGRO, 2,cv2.LINE_AA)
cv2.putText(frame, texto7, (290, 400), 1, 1, ROJO, 2,cv2.LINE_AA)
cv2.putText(frame, "ALARMA", (275, 440), 1, 1, VERDE, 2,cv2.LINE_AA)
#indicacion de PUERTA DE CUARTO dibuja un cuadro de tamaño, grosor, colro y estilode letra
cv2.rectangle(frame, (340, 370), (390, 420), AZUL, m[4])
cv2.putText(frame, texto8, (350, 400), 1, 1, NEGRO, 2,cv2.LINE_AA)
cv2.putText(frame, texto9, (350, 400), 1, 1, ROJO, 2,cv2.LINE_AA)
cv2.putText(frame, "PUERTA", (340, 440), 1, 1, AZUL, 2,cv2.LINE_AA)
cv2.putText(frame, "CUARTO", (340, 460), 1, 1, AZUL, 2,cv2.LINE_AA)
#indicacion de PUERTA PRINCIPAL dibuja un cuadro de tamaño, grosor, colro y estilode letra
cv2.rectangle(frame, (400, 370), (450, 420), NARANJA, m[5])
cv2.putText(frame, texto10, (410, 400), 1, 1, NEGRO, 2,cv2.LINE_AA)
cv2.putText(frame, texto11, (410, 400), 1, 1, ROJO, 2,cv2.LINE_AA)
cv2.putText(frame, "PUERTA", (410, 440), 1, 1, NARANJA, 2,cv2.LINE_AA)
cv2.putText(frame, "PRINCIPAL", (420, 460), 1, 1, NARANJA, 2,cv2.LINE_AA)
cv2.putText(frame, "WILSON CONDORI APAZA", (5, 30), 1, 1, (160,122,255), 2,cv2.LINE_AA)

```

```

cv2.imshow("INTERFAZ DE USUARIO", frame) #mostrar el fotograma en una ventana
if cv2.waitKey(1) & 0xFF == 27: #si se presiona ESC rompe el bucle se cierre la captura de video
    break
cap.release() #cierra la conexion con el video
cv2.destroyAllWindows() #cierra todas las ventanas visuales
#esp.close()#cerrar la comunicacion serial

```

3.5. COSTOS DEL TRABAJO Y DURACION

Se realiza una breve cotización de los costos realizados para la implementación del prototipo.

3.5.1. Costo de Materiales

Se toma en cuenta los materiales de mayor costo y necesarios para el funcionamiento del prototipo de los cuales se realiza el siguiente listado:

- 1Modulo ESP32	85Bs.
- 1 Shield Relé de 5 V, 4 canales	40 Bs.
- 2 Servomotores	30 BS.
- 1 Protoboard	30 BS.
- 1Sensor PIR HC-SR501	18 Bs.
- 1 LDR	2 Bs.
- 5 resistencia de 220 Ohmios	3 Bs.
- TOTAL	208Bs.

3.5.2. Costo de Mano de Obra

Para poder definir el costo de la mano de obra tomando en cuenta la recopilación de datos, programación de todos los sistemas, diseño de simulación y el ensamblado de maqueta, como profesional es imprescindible poder darle un valor al trabajo profesional que se brinda para poder resolver algún problema y entregar un equipo que funcione de la manera como lo desee el usuario,

sin más esta es una estimación de lo que considero que cuesta la mano de obra todo lo mencionado anteriormente.

- Recopilación de datos 20Bs
- Diseño de simulación 20Bs
- Programación de todos los sistemas 100Bs
- Ensamblado de maqueta 30Bs.
- TOTAL 170Bs

3.5.3. Tiempo Empleado

Debido a la simplicidad que puede presentar el presente trabajo o complejidad como quiera verlo el lector tiene como consecuencia un costo debido al tiempo empleado a la implementación desde el boceto de la idea, pasando por la programación y ensamblado del prototipo todo esto implica tiempo y el tiempo también es un factor económico variable que influye en el producto final, pero lo que se menciona de manera breve un resumen del tiempo utilizado para el presente trabajo.

ACTIVIDAD	Semana 1					Semana 2					Semana 3					Semana 4				
	L	M	M	J	V	L	M	M	J	V	L	M	M	J	V	L	M	M	J	J
Recopilación de datos	■	■	■	■	■	■	■	■	■	■										
Diseño de simulación				■	■	■	■	■	■	■										
Programación de todo el sistema								■	■	■	■	■	■	■	■					
Ensamblado de maqueta																■	■	■	■	■

CAPITULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

A finalizar el desarrollo del trabajo de aplicación se llegó a las siguientes conclusiones:

Se pudo describir la situación actual de las necesidades y limitaciones de las personas con discapacidad en relación con el control de iluminación y puertas, explicando como el sistema de control de iluminación y puertas, puede brindarles la autonomía y dejar de depender de terceros para realizar tareas sencillas como el encendido de luces y apertura de puertas.

Se logró describir las características técnicas que tendrá el Diseño del prototipo de un sistema Control de iluminación y puertas, basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32, analizando los componentes electrónicos que lo conforman, el funcionamiento para controlar la luces y la apertura de las puertas, el cómo se controlara todo el sistema de control de iluminación y puertas con señas de los dedos.

Se logró Implementar un modelo a escala del prototipo de un sistema Control de iluminación y puertas, basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32. El mismo ha permitido materializar y demostrar de manera tangible el funcionamiento del sistema de control de iluminación y puertas, permitiendo evaluar las posibles mejoras, así como la optimización del diseño y la funcionalidad, para poder implementarlo a gran escala en un domicilio real.

Se logró diseñar un prototipo de un sistema Control de iluminación y puertas, basado en la detección de gestos de los dedos levantados mediante visión artificial y esp32, para mejorar la autonomía de las personas con discapacidad, el prototipo se plasma en una maqueta para observar su funcionamiento a pequeña escala.

Este sistema es una solución innovadora y accesible que les permitirá a las personas con discapacidad controlar sus dispositivos de forma autónoma y sin la necesidad de utilizar dispositivos tradicionales de control, como interruptores y cerraduras.

La tecnología utilizada es económica y fácilmente adaptable, lo que permite una implementación en gran escala.

El sistema puede ser utilizado no solo por personas con discapacidad, sino también por cualquier persona que quiera tener un control más intuitivo y fácil de sus dispositivos de iluminación y puertas.

La visión artificial y el reconocimiento de gestos de los dedos levantados es una tecnología eficaz y accesible para el control de dispositivos en un entorno doméstico, pero además, el sistema desarrollado es capaz de controlar varios dispositivos de forma simultánea, lo que aumenta su funcionalidad y accesibilidad.

El sistema puede ser mejorado en el futuro, por ejemplo, para agregar soporte para más gestos o para integrarlo con otros dispositivos inteligentes en un hogar.

El proyecto demuestra el potencial de la tecnología para mejorar la calidad de vida de las personas con discapacidad y su autonomía en la vida diaria.

4.2. RECOMENDACIONES

Se recomienda realizar una planificación cuidadosa del proyecto y definir claramente los requisitos y objetivos del sistema antes de comenzar con la implementación.

Es importante realizar una investigación detallada de los componentes electrónicos que se van a utilizar, asegurándose de que sean compatibles y de buena calidad.

Se debe prestar especial atención a la programación del sistema, utilizando buenas prácticas de programación y documentando el código de manera adecuada para facilitar su mantenimiento y evolución.

Se recomienda hacer pruebas exhaustivas del sistema antes de su implementación, utilizando herramientas de simulación y verificación para asegurar su correcto funcionamiento y detectar posibles problemas antes de su puesta en marcha.

En cuanto a la programación en Python, se recomienda utilizar librerías bien documentadas y probadas por la comunidad para evitar errores y aumentar la eficiencia del código.

En el caso del ESP32, se recomienda prestar especial cuidado al manejo de la memoria y la eficiencia del código para asegurar un funcionamiento estable y rápido del sistema.

En la utilización de componentes electrónicos como reles, LDR y PIR, se recomienda leer detenidamente las hojas de especificaciones y seguir las recomendaciones del fabricante para evitar problemas de compatibilidad y mal funcionamiento.

Por último, se recomienda prestar atención a la fuente de alimentación del sistema, asegurándose de que sea suficiente para los componentes electrónicos utilizados y de que se utilicen reguladores de voltaje adecuados para evitar daños en los dispositivos.

BIBLIOGRAFIA

Alonso, A., Rodríguez, J., & Martínez, J. (2019). Sistemas de control de iluminación para la eficiencia energética en edificios. *Revista Electrónica de Tecnología Energética*, (3), 51-60.

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc.

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc.

Cao, Z., Simon, T., Wei, S.-E., & Sheikh, Y. (2017). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7291-7299. MediaPipe. (2021).

Chen, S., Qian, F., Guo, L., Zhang, D., Liu, X., & Yang, Y. (2020). A Review on Passive Infrared Sensor Based Occupancy Detection for Building Energy Efficiency. *Applied Sciences*, 10(18), 6506. <https://doi.org/10.3390/app10186506>

Chiang, Y. (2020). *Building Automation: Control Devices and Applications* (2nd ed.). CRC Press.

Chilamkurti, N. (2020). *Smart Home Systems and Automation: IoT Enabled Design and Implementation* (1ra ed.). Boca Raton, FL: CRC Press.

Diwo BQ. (2023). Descubre el LDR. Recuperado de <http://diwo.bq.com/descubre-el-ldr/> el 14 de mayo de 2023.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6), 2280-2292.

Gómez-Expósito, A., & Suárez-Álvarez, M. M. (2018). Domótica: Diseño e implementación de sistemas domóticos e inmóticos. Paraninfo.

González, R. C., & Woods, R. E. (2018). Procesamiento digital de imágenes. México: Pearson Educación.

González, R., & Pérez, A. (2019). Visión por Computador y Reconocimiento de Patrones. Ediciones Paraninfo.

Google. (2021). MediaPipe Hands. Recuperado el 14 de mayo de 2023, de <https://developers.google.com/mediapipe/solutions/hands>.

Google. (2021). MediaPipe: Cross-platform, customizable ML solutions for live and streaming media. Recuperado el 14 de mayo de 2023, de <https://mediapipe.dev/>

Hernández, J. J. (2013). Fundamentos de electrónica: Diapositivas del curso. Universidad Autónoma de Zacatecas. Recuperado de <http://www.fcfm.buap.mx/ACADEMIA/Cursos/Electronica/Notas/Fundamentos%20de%20Electronica.pdf>

Hughes, J. (2005). Ingeniería de Control: Teoría y Práctica. México: McGraw-Hill Interamericana.

Jain, N. K. (2010). Microcontroller-Based Projects. New Delhi: Tata McGraw-Hill Education.

Karlen, M. y Spangler, C. (2017). Lighting Design Basics (3ra ed.). Hoboken, NJ: John Wiley & Sons.

Machado, A. (2019). Tipos de relés. Grupo Electrón. Recuperado de <https://www.grupoelectron.com/tipos-de-reles/>

Malvino, A. y Bates, D. (2007). Electronic Principles (8va ed.). Nueva York, NY: McGraw-Hill.

Mediapipe. (2021). Hands. Recuperado el 14 de mayo de 2023, de <https://google.github.io/mediapipe/solutions/hands.html>

Ogata, K. (2010). Ingeniería de Control Moderna. México: Pearson Educación.

OpenCV. (2021). Video Input with OpenCV and similarity measurement. Recuperado el 14 de mayo de 2023, de https://docs.opencv.org/master/dd/d43/tutorial_py_video_display.html.

Organización Mundial de la Salud. (2011). Clasificación internacional del funcionamiento, de la discapacidad y de la salud: CIF. Ginebra, Suiza: OMS.

WHO. (2001). Clasificación Internacional del Funcionamiento, la Discapacidad y la Salud (CIF). Ginebra, Suiza: OMS. Recuperado de <https://www.who.int/classifications/icf/es/>

WHO. (2011). Informe mundial sobre la discapacidad. Ginebra, Suiza: OMS. Recuperado de https://www.who.int/disabilities/world_report/2011/es/

Wu, Y., & Zhang, J. (2021). Python Programming for Arduino and IoT. Springer.

ANEXOS

A) Prototipo elabora en forma de maqueta para visualizar su funcionamiento





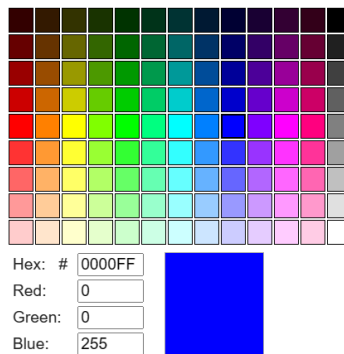
B) Colores en Programación

Para poder saber el valor de los colores podemos utilizar Word para saber el valor o ingresar a la página:

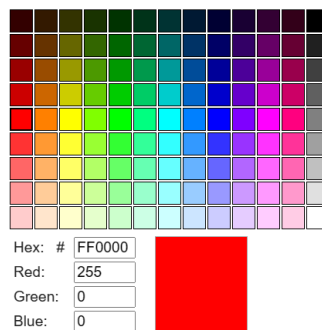
[RGB Color Codes Chart !\[\]\(bd1a142de767a21e5362c595f844a4ff_img.jpg\) \(rapidtables.com\)](https://www.rapidtables.com/web/color/RGB_Color.html)

https://www.rapidtables.com/web/color/RGB_Color.html

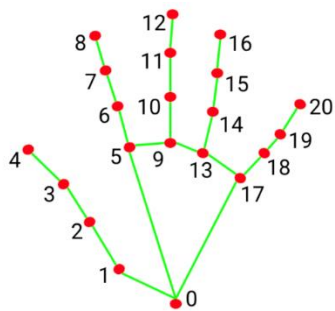
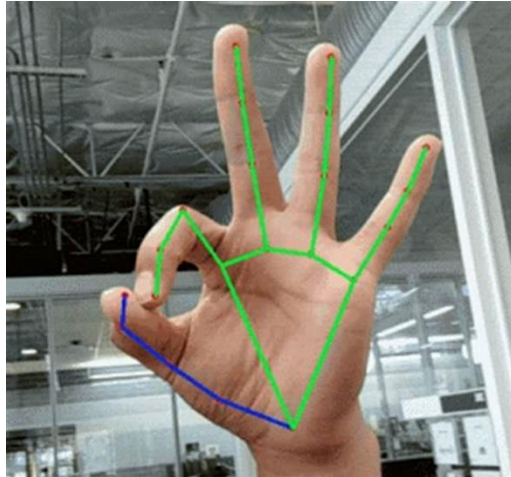
COLOR AZUL



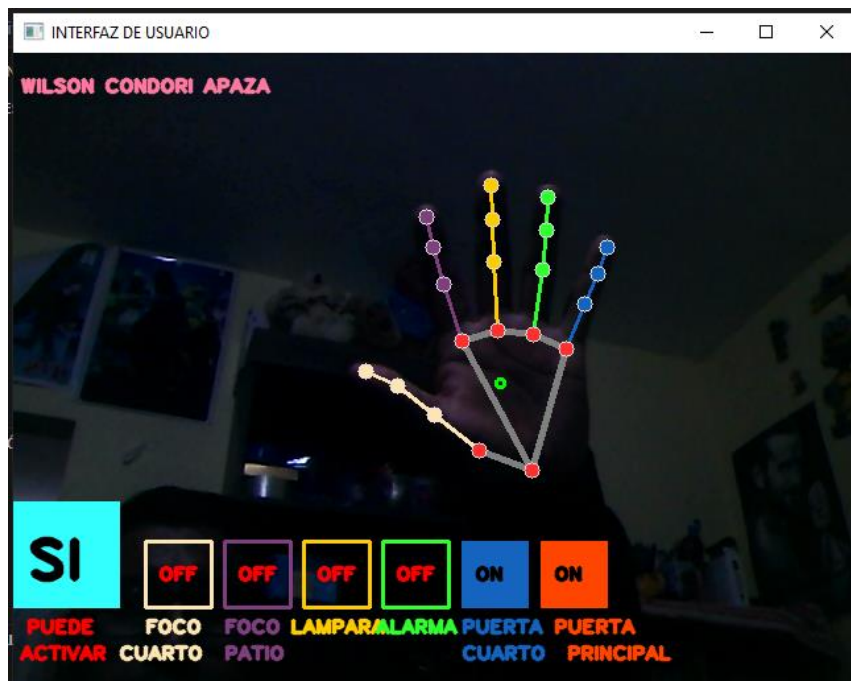
COLOR ROJO



C) PUNTOS DE REFERENCIA MANOS



- | | |
|-----------------------|-----------------------|
| 0. WRIST | 11. MIDDLE_FINGER_DIP |
| 1. THUMB_CMC | 12. MIDDLE_FINGER_TIP |
| 2. THUMB_MCP | 13. RING_FINGER_MCP |
| 3. THUMB_IP | 14. RING_FINGER_PIP |
| 4. THUMB_TIP | 15. RING_FINGER_DIP |
| 5. INDEX_FINGER_MCP | 16. RING_FINGER_TIP |
| 6. INDEX_FINGER_PIP | 17. PINKY_MCP |
| 7. INDEX_FINGER_DIP | 18. PINKY_PIP |
| 8. INDEX_FINGER_TIP | 19. PINKY_DIP |
| 9. MIDDLE_FINGER_MCP | 20. PINKY_TIP |
| 10. MIDDLE_FINGER_PIP | |



D) SEÑAS PARA CONTROLAR EL SISTEMA
PERMITE REALIZAR LA SELECCIÓN
DE UNA ACCION



ENCENDER CUARTO



ENCENDER PATIO



APAGAR CUARTO



APAGAR PATIO



ENCENDER LAMPARA



APAGAR LAMPARA



ENCENDER ALARMA



APAGAR ALARMA



ABRIR PUERTA



CERRAR PUERTA



ABRIR PRINCIPAL



CERRAR PRINCIPAL

