UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN

“Desarrollo de aplicaciones domóticas en MATLAB App Designer, implementadas en una tarjeta de desarrollo Arduino”

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN TELECOMUNICACIONES, SISTEMAS Y ELECTRÓNICA

PRESENTA:
JONATHAN FUENTES EUAN

ASESOR: DR. FERNANDO GUDIÑO PEÑALOZA

UNAM
CUAUTITLÁN

Cuautitlán Izcalli, Estado de México, 2021

VNIVERSIDAD NACIONAL AVPNOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
SECRETARÍA GENERAL
DEPARTAMENTO DE TITULACIÓN

U.N.A.M.
FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
DEPARTAMENTO DE EXÁMENES PROFESIONALES

ASUNTO: VOTO APROBATORIO

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLAN
PRESENTE

ATN: I.A. LAURA MARGARITA CORTAZAR FIGUEROA
Jefa del Departamento de Titulación
de la FES Cuautitlán.

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos el trabajo de: **Tesis y examen profesional.**

Desarrollo de aplicaciones domóticas en MATLAB App Designer, implementadas en una tarjeta de desarrollo Arduino.

Que presenta el pasante: **Jonathan Fuentes Euan.**
Con número de cuenta: **308009288** para obtener el Título de: **Ingeniero en Telecomunicaciones, Sistemas y Electrónica.**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**.

ATENTAMENTE
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Méx. a 07 de septiembre de 2021.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	Mtro. Jorge Buendía Gómez	
VOCAL	Ing. Noemi Hernández Domínguez	
SECRETARIO	Dr. Fernando Gudiño Peñaloza	
1er. SUPLENTE	Mtro. Leopoldo Martín del Campo Ramírez	
2do. SUPLENTE	Dr. David Tinoco Varela	

NOTA: los sinodales suplentes están obligados a presentarse el día y hora del Examen Profesional.

LMCF/cnn*

La locura puede ser un escape.
Si las cosas no están bien, entonces tú puedes querer imaginar algo mejor.
En la locura, pensaba que yo era la persona más importante del mundo.

John Nash.

AGRADECIMIENTOS

A Dios:

Porque ha mantenido a mis seres queridos y a mí con salud, porque me ha dado la paciencia necesaria cuando más la necesitaba, porque puso en mi camino a gente maravillosa a la cual también tengo mucho que agradecer y sobre todo porque me ha otorgado el regalo más grande de todos: “vida”.

A mis padres Claudia Euan Estrada y José Gabriel Fuentes Mendoza:

Por todo el apoyo que me han brindado a lo largo de mi vida, hoy en este trabajo encuentro un pequeño espacio para poder agradecerles todo lo que hicieron, hacen y sé que seguirán haciendo por mí y hoy en día por mis 4 mujercitas. Porque me enseñaron que la vida puede tirarte y/o pisotearte, pero depende de uno levantarse y seguir adelante, porque han sido desde mi niñez hasta estos momentos los mejores maestros de vida, aunque en lo académico llegó un momento en el que ustedes ya no pudieron enseñarme, siempre encontraron la manera de que pudiera aprender las cosas; porque gracias a sus valores y enseñanzas siento que soy un hombre de bien en todo sentido.

A mi pareja Gabriela Sarahí:

Por ser mi compañera de vida, por apoyarme a encontrar los tiempos suficientes para que pudiera realizar este trabajo de tesis, por alimentar mi alma con sueños y metas, porque me has demostrado que nunca es tarde para seguir adelante, gracias por esa forma tan hermosa que tienes de sorprenderme cada día, ya sea con una palabra, con una mirada, con una sonrisa o con unos platanitos fritos en mi barriga. Sobre todo, muchas gracias por ayudarme en ir mejorando en mi redacción y ortografía, así como, por darme tus puntos de vista en la elaboración de este trabajo.

A mi hija Lía Regina:

Porque tu llegada a mi vida generó grandes cambios, eres mi mayor motivo para tratar de ser mejor persona cada día.

A mi hermana Gabriela Leticia:

Gracias gordita preciosa, por acompañarme en mis locuras desde pequeños. Eras la única persona que sabía que quería estudiar I. T. S. E. y que la Facultad se encontraba a más de 35 km de nuestra casa y en lugar de desanimarme encontré apoyo en ti, guardaste el secreto hasta que fue un hecho mi admisión a la FESC y tuve que decirlo, porque me apoyaste económica y psicológicamente cuando notabas que necesitaba algo.

A Sarah Viviana y Valeria Sofía:

Porque han soportado mi mal humor y aun así me llenan de cariño cada día, por esas porras que me dan cuando lo he necesitado, por haber llegado a mi vida.

A mis mejores amigos:

Francisco Mejía Tapia, porque he contado contigo en cada momento de mi vida, me enseñaste que lo más importante para invertir es en la educación, que hay amigos que se vuelven parte de tu familia y que jamás se debe perder la humildad.

Antonio Guerrero Juárez, David Alejandro Hernández Garduño y Octavio Roa Saavedra, porque estuvieron en todo mi proceso de aprendizaje, porque a pesar del tiempo sin vernos y grandes distancias de nuestros hogares, seguimos en contacto.
Porque con ustedes hacen honor al viejo dicho italiano: "Chi trova un'amico, trova un tesoro (Quien encuentra un amigo, encuentra un tesoro)"

A mi jefe y padre académico:

Ing. Marcelo Bastida Tapia, gracias por todo el apoyo y confianza que ha puesto en mí, por ser mi jefe, maestro, amigo y padre académico, porque siempre me ayudó a entender temas que en ocasiones nadie más podía explicarme, porque después de ser mi maestro me siguió enseñando que uno debe seguirse preparando no solo en lo académico.

A mi asesor de tesis:

Dr. Fernando Gudiño Peñaloza, gracias por el tiempo que dedicó a la elaboración de este trabajo de tesis, por aceptarme como su tesista, por las enseñanzas brindadas durante y después de la carrera, porque a pesar de la situación que estamos viviendo el día de hoy se tomó su tiempo para tutorarme y como es su costumbre por seguirme enseñando.

A mis sinodales:

Mtro. Jorge Buendía Gómez
Ing. Noemi Hernández Domínguez
Mtro. Leopoldo Martín del Campo Ramírez
Dr. David Tinoco Varela

Gracias por ser parte de este trabajo, y más aún, por brindarme la formación académica durante la carrera para tener las bases necesarias para desarrollar el tema planteado.

A mi amada UNAM - FES Cuautitlán

Porque me enseño que, no se necesitan de lujosas instalaciones para poder aprender cosas nuevas, por ser mi segundo hogar, por brindarme una buena preparación académica, darme buenos amigos y llenarme de constantes aprendizajes.

Por mi raza hablará el espíritu.

ÍNDICE

INTRODUCCIÓN

En el año 1966, se da a conocer el ordenador doméstico ECHO IV (Fig. I), el primer dispositivo desarrollado en Pittsburgh por el ingeniero James Shutherland para realizar una casa inteligente, este dispositivo podía controlar el televisor, programar el despertador y los relojes digitales, por mencionar algunas de sus aplicaciones, la desventaja de este ordenador era el tamaño, ya que ocupaba toda una habitación y se tenía la necesidad de colocar varios teclados por toda la casa.

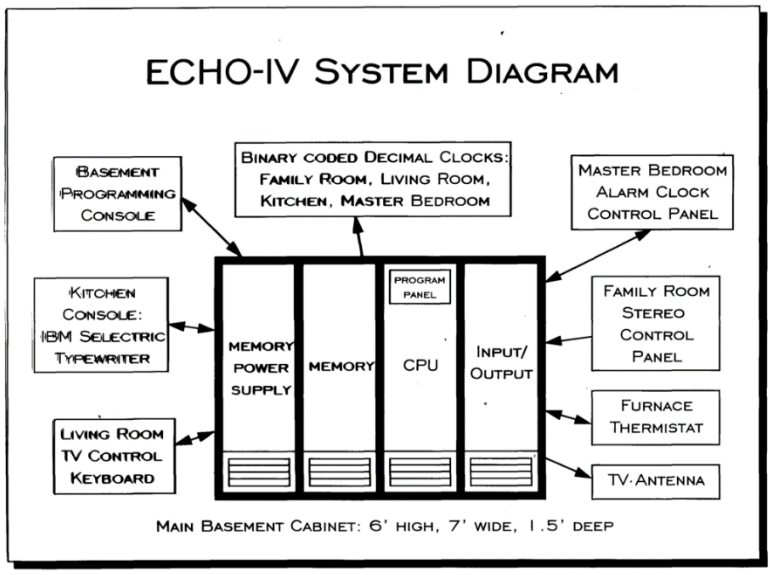


Fig. I Diagrama del sistema principal ECHO IV. [1]

Desafortunadamente el ECHO IV no fue considerado como parte de las primeras tecnologías domóticas, en su lugar varios autores mencionan que el protocolo de comunicación X-10 fue el primer sistema domótico, sin embargo, fue desarrollado hasta el año de 1978. El primer módulo podía controlar cualquier dispositivo a través de la red eléctrica doméstica (120 o 220 V y 60 o 50 Hz) modulando pulsos de 120 KHz (0 = sin pulso, 1 = pulso). [2]

Posteriormente, en 1983 Motorola desarrolla el teléfono móvil *Dynatac 8000x*, catalogándolo como el primer teléfono móvil del mundo que puede ser sostenido con una sola mano. Podríamos decir que es considerado el predecesor de los teléfonos móviles actuales. [3]

Mirar al pasado permite recordar los errores y aciertos que se han tenido, consideremos el ECHO IV, su mayor problema fue el tamaño de la central de operación y los múltiples dispositivos de control (teclados) que tenían que estar distribuidos en toda la casa, la solución hoy en día sería ocupar sistemas de control más discretos como una PC, un celular, una Tablet, sensores de movimiento o voz.

El ECHO IV, el protocolo de comunicación X-10 y el teléfono móvil *Dynatac 8000x* son inventos que permitirían el auge y la mejora de los sistemas de automatización. En los recursos de consulta impresos y digitales será común encontrar los términos de domótica y en menor medida inmótica, si bien tienen muchas similitudes en las aplicaciones y sistemas utilizados, el sector al que va dirigido es distinto, es por esto por lo que comúnmente encontraremos empleada la palabra domótica refiriéndose a las aplicaciones desarrolladas para la inmótica.

Para adentrarse al mundo de la automatización del hogar, hay que tener presente los conceptos muy claros; por tal motivo, se mencionará la definición y diferencias que existen entre la domótica y la inmótica. Las definiciones pueden cambiar un poco entre cada autor, para este trabajo se tomará en cuenta las definiciones que brinda la Asociación Española de Domótica, anteriormente nombrada Comité Español de la Domótica (CEDOM[1]).

¿Qué son la domótica y la inmótica?

La domótica es el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía, que aporta seguridad y confort, además de comunicación entre el usuario y el sistema.
Un sistema domótico es capaz de recoger información proveniente de sensores o entradas, procesarla y emitir órdenes a actuadores o salidas, asimismo, puede acceder a redes exteriores de comunicación o información. [4]

[1] Aunque la Asociación Española de Domótica cambió su nombre sigue conservando sus iniciales como CEDOM.

La palabra inmótica no está incorporada al diccionario de la Real Academia Española (RAE), sin embargo, la Asociación Española de Domótica e Inmótica dice que es el conjunto de tecnologías aplicadas al control y la automatización inteligente de edificios no destinados a la vivienda [5], como son:

- Hoteles.
- Centros comerciales.
- Universidades.
- Hospitales.
- Aeropuertos.
- Plantas industriales.
- Etc.

Es posible referirse a una escuela como un edificio no destinado a la vivienda, que permite mejorar el control en las áreas donde se generan gastos de energía innecesarios, por ejemplo, puede generarse un sistema capaz de detectar la presencia de alumnos y/o profesores en un aula, generalmente las aulas se encuentran alumbradas todo el tiempo, sin embargo, la luminosidad requerida por la mañana no es la misma que se requiere por la tarde, por la noche o cuando ya no se encuentra alguien.

Las principales aportaciones que debe brindar la domótica y la inmótica a los usuarios son:

- *Confort*[2].
- Seguridad.
- Eficiencia eléctrica.
- Accesibilidad.
- Sistema fácil de manejar.

[2] De acuerdo con la RAE, la palabra *"Confort"* es empleada para considerar el bienestar y comodidad. En las fuentes de consulta referente a la domótica es común encontrar este término.

Aunque parece haber muchas ventajas en los sistemas domóticos, también se encuentran grandes desventajas, por ejemplo:

- El costo de instalación en los servicios puede resultar ser elevado para algunos sistemas de automatización.
- La carencia de personal que se encuentre realmente especializado para llevar a cabo el desarrollo y las instalaciones de los sistemas domóticos es una de las grandes desventajas.

Se espera a futuro que cada vivienda pueda ser o estar domotizada, con ello se podrá reducir gastos en consumos de energía, mejorar la calidad de vida para las personas (sobre todo si se cuenta con alguna capacidad diferente), aportar mayor comodidad y accesibilidad a los aparatos o áreas del hogar, aprovechando las ventajas que se cuentan en la actualidad, como la mejoría en los servicios de internet, computadoras y celulares con mayor capacidad de almacenamiento y menores tiempos de respuesta, así como, un conocimiento más amplio respecto al tema de la domótica y los dispositivos electrónicos.

JUSTIFICACIÓN

El plan de estudios de la carrera de I. T. S. E. se compone por 48 asignaturas distribuidas en nueve semestres, de las cuales 43 son obligatorias, 3 obligatorias de elección pertenecientes a uno de los 5 campos disciplinarios terminales (Comunicaciones, Ingeniería de Control y Mecatrónica, Sistemas Digitales, Sistemas Analógicos y Sistemas de Información) y 2 optativas (el alumno puede elegir de un total de 28 asignaturas ofertadas). Las asignaturas que se encuentran relacionadas con este trabajo son: Simulación de Sistemas, Sistemas Digitales, Microprocesadores, Microcontroladores, Sistemas Inteligentes, Diseño de Sistemas Digitales Avanzados y principalmente Domótica. [6]

La asignatura de domótica tiene como objetivo general que al finalizar el curso el alumno comprenda y aplique los conceptos fundamentales de la ingeniería domótica, asimismo, que conozca los equipos más representativos de los edificios inteligentes y hogares automatizados. Actualmente, la asignatura no cuenta con un manual de prácticas de laboratorio, por ello se pretende desarrollar e implementar aplicaciones orientadas a la domótica, las cuales se presentarán en forma de manual, esperando que sea de utilidad para aquel alumno que curse la asignatura pueda complementar su formación académica, reforzando y relacionando los conocimientos teóricos con la práctica.

Este trabajo de tesis pretende ser un apoyo para los estudiantes de semestres más avanzados que quieran adentrarse al mundo del desarrollo de aplicaciones implementadas a la domótica. De la misma manera, se espera aportar a la sociedad ingenieros mejor preparados, capaces no solo de instalar una bombilla inteligente o un asistente personal, sino que tengan la habilidad de comprender el funcionamiento interno y externo, ventajas y desventajas para la mejora en la instalación e implementación de los sistemas domóticos.

OBJETIVOS

GENERAL

- Desarrollar en App Designer de MATLAB aplicaciones orientadas a la domótica e implementarlas en la tarjeta de desarrollo Arduino UNO.

ESPECÍFICOS

- Desarrollar e implementar diferentes aplicaciones relacionadas con los temas de seguridad, automatización y confort.

- Presentar las aplicaciones en forma de manual permitiendo al lector adquirir los conocimientos necesarios para implementar sistemas domóticos básicos.

- Despertar el interés del lector en seguir investigando y desarrollando sistemas domóticos con los elementos mencionados en este trabajo o con aquellos que surjan de su propia investigación.

ESTRUCTURA DE LA TESIS

Este documento se encuentra estructurado en cuatro capítulos, conclusiones del trabajo de tesis, referencias bibliográficas y los anexos.

En el capítulo uno se abordan los aspectos relevantes de las arquitecturas domóticas, los componentes de los sistemas, el software y elementos necesarios para el desarrollo de las aplicaciones propuestas.

En el capítulo dos se describe el proceso necesario para el desarrollo de las aplicaciones domóticas, desde la inicialización de nuestros sistemas, hasta su relación con los componentes electrónicos, eléctricos y demás necesarios.

En los capítulos tres y cuatro se desarrolla y demuestra la funcionalidad de los diversos sistemas empleados en ambientes domóticos, haciendo énfasis en aquellos que sirven para mejorar las condiciones de confort y seguridad física de los habitantes de una casa habitación. Se ha decidido solo enfocarse en estas áreas de la domótica, dado que el universo de aplicaciones a desarrollar es vasto, por tanto, abarcar todas ellas está más allá del enfoque de esta tesis.

En las conclusiones obtenidas del desarrollo de este proyecto se indican las observaciones pertinentes para la continuación de este en un trabajo a futuro.

Las referencias bibliográficas mencionadas en este trabajo son las páginas web, libros y artículos adecuados para abordar los temas incluidos en esta tesis.

Por último, se incluyen los anexos en donde se muestran los códigos generados para el desarrollo de las aplicaciones y la comunicación establecida con los dispositivos electrónicos utilizados.

CAPÍTULO 1

MARCO TEÓRICO

1.1. Elementos de una arquitectura domótica

La clasificación de los sistemas de control dependerá del tipo de soporte que sea utilizado como vía de comunicación en un sistema domótico y/o según sea el tipo de tecnología de control. A continuación, se describen las características de cada tipo de sistema.

1.1.1. Tipos de soportes utilizados como vía de comunicación

- **Sistemas punto a punto (conexión estrella):** Cada elemento sensor o actuador se conecta al sistema de control central mediante dos conductores, son recomendados para instalaciones en viviendas particulares o en pequeños edificios del sector terciario.

- **Sistemas mediante bus:** Se basan en un soporte físico cableado (bus) y dedicado exclusivamente a la comunicación entre los diferentes sensores, elementos de mando, actuadores, central, etc. Habitualmente están formados por 2 conductores de 0.5 mm^2 (20 AWG American Wire Gauge), permitiendo enviar las señales a velocidades de hasta 10 megabits/s. Resultan adecuados para el control de grandes edificios, instalaciones de climatización, iluminación, gestión de accesos, automatismos de socorro, gestión de la seguridad del edificio, gestión de la alarma contra incendios, mantenimiento asistido por ordenador, etc.

- **Sistemas mixtos de comunicación (bus y punto a punto):** Utilizan un bus para establecer la comunicación entre la unidad central de control (en su mayoría son sistemas centralizados) y los módulos de distribución, desde los cuales se realiza la conexión punto a punto con los distintos sensores y actuadores de la instalación.

- **Sistemas sobre la red eléctrica (onda portadora):** Se trata de una transmisión analógica de la señal, en la cual esta señal hace variar una onda portadora (de ahí su nombre), modulándola según una amplitud y una frecuencia determinadas. En la

recepción, esta modulación se filtra separándola de la onda portadora para reconstruir la señal original enviada.

- **Sistemas sobre otros soportes (vía radio, wifi, bluetooth[3], línea telefónica, infrarrojos y fibra óptica):** La transmisión de instrucciones es inalámbrica, se requiere de los dispositivos y la configuración adecuada para poder hacer uso de este sistema. Cada uno de estos medios de transmisión de información cuenta con su propia peculiaridad:
    - Distancias mínimas o máximas de transmisión.
    - Diferentes costos al implementarlos.
    - Velocidades de transmisión de datos.
    - Sistemas de seguridad que se les pueden implementar.

1.1.2. Tipos de tecnología de control

- **Sistemas centralizados:** Como criterio básico pueden catalogarse dentro de este grupo aquellos sistemas en los que haya un solo microcontrolador, encargado de la gestión de la instalación, que será el encargado de ejecutar el programa que rige el funcionamiento de esta.

  Si existe un fallo en la central, y esta queda temporalmente fuera de servicio, el sistema global queda inutilizado por completo. Esta central de control puede ser un autómata programable, un ordenador, o una central domótica especialmente diseñada para controlar un determinado conjunto de elementos (circuito electrónico secuencial).

[3] Bluetooth es una especificación industrial para redes inalámbricas de área personal (WPAN) creado por Bluetooth *Special Interest Group, Inc*. que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz.

Para más información visita la página: https://www.bluetooth.com/

- **Sistemas descentralizados:** Entran en esta categoría los sistemas en los que el control de la instalación se realiza de forma parcial e independiente mediante varios microcontroladores repartidos por ella.
  La ventaja de estos sistemas es que en caso de estropearse alguno de los controladores solo se ve afectada la parte de la instalación que gobierna. El inconveniente es que no pueden interactuar los distintos sistemas, al ser estos totalmente independientes.

- **Sistemas mixtos de gestión:** Son aquellos sistemas en los que, siendo descentralizados, se puede actuar sobre la instalación desde un ordenador situado en un nivel de gestión superior. Desde él se puede visualizar y variar el estado de las entradas y salidas en los distintos microcontroladores que intervienen en el sistema o en sistemas distintos (no tienen por qué estar gobernando la misma instalación) sin que realmente estos se hayan activado.

  Normalmente el ordenador que se decida ocupar se encontrará conectado con los distintos autómatas u ordenadores que gobiernan los sistemas localmente, mediante un bus que puede ser local o Ethernet (se podría trabajar mediante hojas de internet en donde los autómatas principales incorporen su propia web). [7]

1.2. Entorno de desarrollo MATLAB

Se utilizará MATLAB® para realizar la comunicación y programación con la tarjeta de desarrollo, mientras que con la herramienta App Designer se generará la interfaz de usuario de las aplicaciones.

MATLAB® es una plataforma de programación y cálculo numérico utilizada por millones de ingenieros y científicos para analizar datos, desarrollar algoritmos y crear modelos. Combina un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de *arrays* directamente. [8]

Puede ser utilizado en:

- Sistemas de control.
- Aprendizaje Automático (*Machine Learning*).
- Procesamiento de señales.
- Aprendizaje Profundo (*Deep Learning*).
- Mantenimiento predictivo.
- Prueba y medición.
- Procesamiento de imágenes y visión artificial.
- Robótica.
- Comunicaciones inalámbricas.
- Sistemas embebidos.
- Etc.

1.2.1. Requisitos de operación para el entorno MATLAB

Para poder llevar a cabo la instalación del software de MATLAB y sus complementos se deben considerar los requisitos mínimos y recomendados del sistema y del equipo de cómputo a utilizar.

Como MATLAB fue instalado en un equipo con sistema operativo Windows solo se considerará en este trabajo las recomendaciones[4] descritas para este sistema operativo.

Sistema Operativo:

- Mínimo: Windows 7 *Service Pack* 1.
- Recomendado: Windows 10 (versión 1803 o superior).

[4] Para conocer los requerimientos necesarios para la instalación en Linux o MAC pueden consultarse en: https://la.mathworks.com/support/requirements/matlab-system-requirements.html

Memoria RAM:

- Mínimo: 4 GB.
- Recomendada: 8 GB o superior.

Memoria en disco físico:

- Mínimo: 5 a 8 GB de espacio disponible para una instalación típica.
- Recomendado: 32 GB o más de espacio disponible para una instalación completa de los productos de *MathWorks*.

Tipo de disco físico:

- Mínimo: HDD – Disco duro.
- Recomendado: SSD – Disco de Estado Sólido.

Procesador:

- Mínimo: Cualquier procesador de Intel o AMD64 (x86-64).
- Recomendado: Cualquier procesador Intel o AMD64 (x86-64) con cuatro núcleos lógicos y soporte de conjunto de instrucciones AVX2.

Tarjeta de video:

- No se requiere de una tarjeta gráfica en específico. [9]

Aunque la página oficial de MATLAB brinda los requisitos mínimos y recomendados para instalar y ejecutar su software, el desarrollo de aplicaciones y su implementación puede hacer que se requieran más recursos del sistema, haciendo que aun cumpliendo con los requisitos mencionados en los puntos anteriores se requiera algún tipo de mejora, por ejemplo:

La memoria RAM recomendada es de 8 GB, pero debemos tomar en cuenta que al iniciar una computadora sin abrir ningún programa ya está utilizando los recursos requeridos por el Sistema Operativo (en este caso Windows 10), por tanto, podría ser que al ejecutar las aplicaciones el consumo sea mayor y de no contar con suficiente memoria RAM libre puede generar que el equipo se detenga, que la aplicación no se ejecute de forma correcta o un cierre inesperado del programa.

1.2.2. Características del equipo de cómputo utilizado

Considerando las especificaciones establecidas por MATLAB, se buscó tener el equipo de cómputo con las características recomendadas y no las mínimas.

El equipo de cómputo utilizado al momento de instalar, desarrollar e implementar las aplicaciones para este trabajo de tesis son:

- Sistema Operativo: Windows 10 *Home Single Lenguage*, Versión 20H2.
- Memoria RAM: 8 GB.
- Tipo de disco físico: SSD A2000 NVMe PCle.
- Capacidad del SSD: 500 GB.
- Procesador: Intel® Core™ i5-8300H.
- Tarjeta de video: NVIDIA GeForce GTX 1050.

1.2.3. App Designer

App Designer integra las dos tareas principales de la creación de aplicaciones: diseñar los componentes visuales de una interfaz gráfica de usuario (GUI) y programar el comportamiento de la aplicación. En la página oficial de MATLAB puede observarse la siguiente descripción:

App Designer te permite crear aplicaciones profesionales sin tener que ser un desarrollador de software profesional, ya que se puede arrastrar y soltar componentes

visuales para desarrollar el diseño de la interfaz gráfica de usuario (GUI) y con el editor integrado se podrá programar rápidamente su comportamiento. [10]

Aunque la descripción menciona que no se tiene que ser un desarrollador de software profesional para utilizar App Designer, se mostrará con un ejemplo sencillo que es indispensable contar con conocimientos previos en fundamentos de programación, por lo que es aconsejable saber por lo menos algún tipo de lenguaje de programación de medio nivel como C y/o C++ o de alto nivel como java, C#, Python, etc., y preferentemente estar familiarizado con el lenguaje de programación .m[5] debido a las pequeñas o grandes diferencias con las que se lleguen a presentar.

Para comprender mejor la recomendación que se hace, se desarrolló un programa con el editor de MATLAB generando el archivo .m y el mismo ejemplo de programa será implementado en el lenguaje de programación C++, así se podrá comparar las diferencias que existen al programar en MATLAB y cuando se realiza la programación en un lenguaje diferente (en este caso C++).

Descripción del programa ejemplo: Implementando el ciclo for desarrollar un programa que realice un incremento de 10 en 10, el valor inicial debe ser cero y deberá detenerse cuando el contador llegue al número 100.

- Código implementado en MATLAB (Fig. 1.1) y resultados mostrados en *Command Window* (Fig. 1.2).

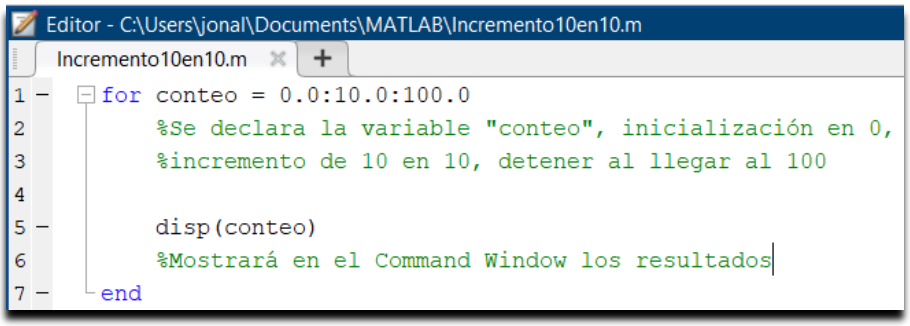


Fig. 1.1 Código del programa en el editor de MATLAB.

[5] .m es la extensión de los programas generados con el editor de MATLAB.

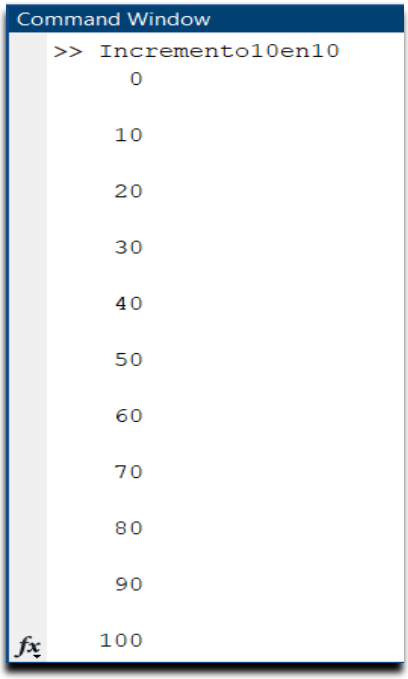


Fig. 1.2 Resultados de la programación en MATLAB.

- Código implementado en C++ (Fig. 1.3) y resultados mostrados en una ventana de sistema con la extensión .exe (Fig. 1.4).

Incremento10en10.cpp

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main(){
6
7       int numero=10;
8       //Variable de tipo entero con valor 10
9
10      for(int conteo=0; conteo<=10; conteo++){
11  //  for(inicialización en 0; números que mostrará la
12  //  consola (10 + inicialización = 11); incremento de 1 en 1)
13
14          cout<<numero*conteo<<endl;
15          //Función creada para indicar que el número
16          //que se ingresó debe multiplicarse por el
17          //incremento (1, 2, 3, etc.)
18      }
19      return 0;
20  }
```

Fig. 1.3 Código del programa en C++.

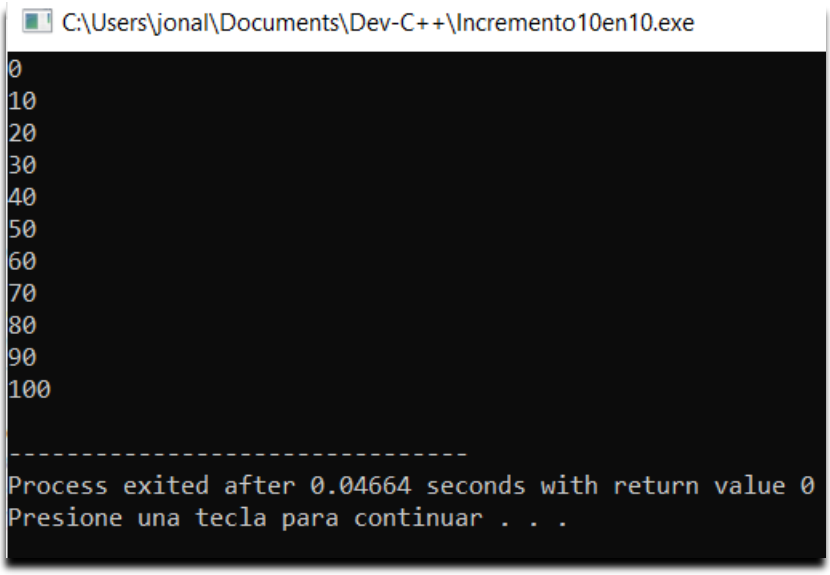


Fig. 1.4 Resultados de la programación en C++.

Como puede observarse en el ejemplo anterior, la sintaxis, líneas de código y estructura de un mismo programa puede cambiar de un lenguaje de programación a otro, sin embargo, la lógica de programación y los resultados obtenidos deben de ser los mismos. Por esta razón es recomendable contar con conocimientos básicos de programación en MATLAB y en algún otro lenguaje de medio o alto nivel.

1.3. Tarjetas de desarrollo para proyectos domóticos

Elegir entre una tarjeta de desarrollo, un microcontrolador o un miniordenador es el primer punto que debe plantearse antes de llevar a cabo el desarrollo de los proyectos, para ello cada persona debe considerar los puntos que se crean más convenientes como: costos, facilidad de compra, información impresa y/o digital que hay sobre el dispositivo a utilizar, los conocimientos previos y futuros que pueden llegar a adquirirse, por tal motivo en la Tabla 1 se muestran algunos puntos que fueron considerados en la elección de la tarjeta de desarrollo.

	Arduino UNO REV3	TARJETA CLON DE ARDUINO	TEXAS INSTRUMENTS TM4C123G	RASPBERRY PI 4 MODELO B 2 GB RAM	BASYS 3 Artix-7 FPGA
Costo[6]	**$ 509.13**	**$ 100.00**	**$ 968.40**	**$1,194.00**	**US$ 149.00**
Categoría	**Tarjeta de desarrollo**	**Tarjeta de desarrollo**	**Tarjeta de desarrollo**	**Miniordenador**	**Tarjeta de desarrollo**
Comunicación con MATLAB	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**
Paquete desarrollado en MATLAB para la tarjeta	**Sí**	**Sí**	**No**	**Sí**	**No**
Cuenta con Wifi o Bluetooth	**No**	**No**	**No**	**Sí**	**No**
Es compatible con módulos Wifi o Bluetooth	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**
Facilidad para adquirirla en tiendas de electrónica	**Fácil**	**Fácil**	**Limitada**	**Fácil**	**Muy limitada**
Calidad en los materiales utilizados	**Buena**	**Mala-Regular**	**Buena**	**Buena**	**Buena**

Tabla 1 Comparativo de las diferentes tarjetas de desarrollo.

1.3.1. Consideraciones y selección de tarjeta

Complementando la Tabla 1 se describen a continuación las características de las tarjetas de desarrollo evaluadas con el fin de poder definir con cuál de ellas se llevará a cabo la implementación de las aplicaciones descritas en este trabajo.

[6] Los precios de la placa Arduino UNO REV3 y de la TIVA C se obtuvieron de www.amazon.com.mx, mientras que el costo de la Tarjeta R3 CH340G compatible con Arduino UNO se obtuvo de https://store.robodacta.mx/, el costo de la Raspberry se obtuvo de https://www.agelectronica.com/index.php, mientras que la BASYS 3 solo pudo cotizarse en la página oficial https://store.digilentinc.com/ ya que no se encontró en venta en alguna de las páginas mencionadas por ello el precio mencionado está expresado en dólares.

La consulta fue realizada el día 22 de marzo de 2021 y en ningún caso se tomó en consideración costos de envío o materiales adicionales para su uso.

La tarjeta Raspberry Pi (Fig. 1.5) es catalogada como un miniordenador, la cual requiere de una fuente de alimentación externa, un monitor con entrada HDMI, cable HDMI, ratón y teclado con conexión USB, una tarjeta SD o microSD con adaptador para poder instalar el sistema operativo en ella, por último y de forma opcional es recomendable que se cuente con una carcasa de protección que incluya un ventilador para enfriar la tarjeta junto con sus disipadores de calor; se hace mención a estos puntos ya que todos estos dispositivos y complementos no vienen incluidos al momento de adquirirla, siendo esta la principal razón para descartar su uso.

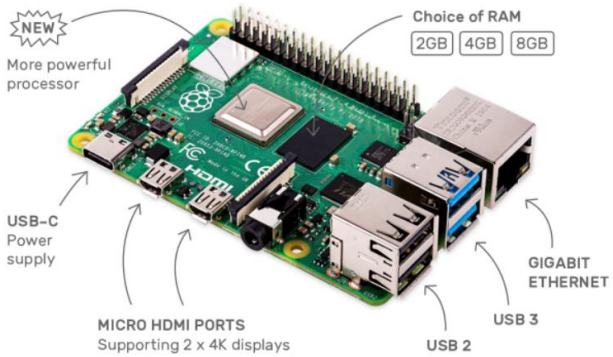


Fig. 1.5 Raspberry Pi 4. [11]

Otra opción que se tiene es la BASYS 3 (Fig. 1.6), aunque cuenta con elementos que se pueden utilizar para futuros proyectos y no solo para este trabajo, la dificultad y costos elevados de adquisición hacen que no sea la tarjeta ideal para llevar a cabo el desarrollo de las aplicaciones.

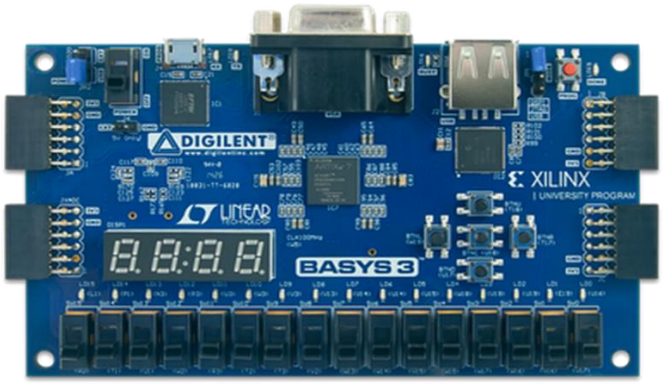


Fig. 1.6 Basys 3. [12]

Tomando en cuenta las especificaciones anteriores y de acuerdo con la Tabla 1, se puede decir que la tarjeta TIVA C TM4C123G (Fig. 1.7) es la mejor opción, ya que cuenta con características muy similares a las que tiene una tarjeta Arduino UNO y aún más, considerando que:

- El procesamiento de datos es más rápido.
- Cuenta con más pines de entrada y de salida.
- Es compatible con MATLAB.
- Cuenta con un switch para energizar o no a la tarjeta, lo cual en cuestiones de seguridad y alguna emergencia que lo requiera puede ayudar bastante.
- Tiene integrado tres botones para su uso.
- Integra en la misma tarjeta un RGB.
- La alimentación del circuito es a través de un cable de tipo microUSB.
- Se pueden hacer conexiones con sus pines a través de cables con entrada hembra o macho.

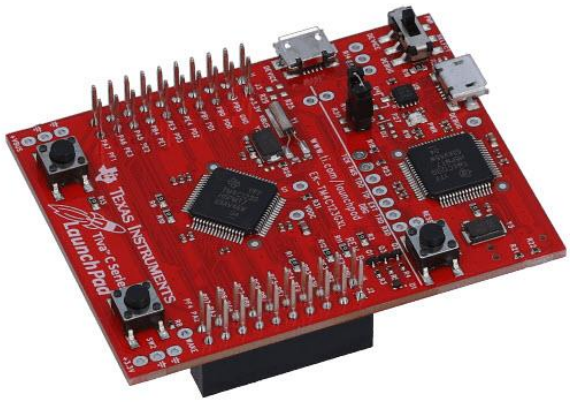


Fig. 1.7 TIVA C TM4C123G. [13]

Sin embargo, el principal motivo para no utilizar esta tarjeta de desarrollo fue que, para adquirirla solo puede ser en lugares muy específicos como la página oficial de *Texas Instruments*[7], dependiendo del tiempo de entrega, lo que genera que obtenerla pueda ser un proceso de espera largo a diferencia de los lugares en donde se compra alguna otra tarjeta de

[7] https://www.ti.com/ es la página oficial de Texas Instruments

desarrollo. En la página oficial el precio de la tarjeta está expresado en dólares, por tanto, la compra queda sujeta al tipo de cambio que hay al momento de su adquisición y es muy probable que se generen costos adicionales a la entrega además del precio base de la tarjeta lo cual menciona *Texas Instruments* en la página de compra.

También puede adquirirse en Amazon, sin embargo, el costo cotizado en esta tienda en línea es de casi el doble en comparación de la tarjeta Arduino UNO REV3 y nueve veces más que el de una tarjeta clon de Arduino (Tabla 1).

1.3.2. La tarjeta Arduino

Tomando en cuenta las características mencionadas en la Tabla 1 y lo visto en el apartado "1.4.1. Consideraciones de las tarjetas propuestas", se decidió utilizar la tarjeta original Arduino UNO REV3 ya que cuenta con los requisitos necesarios para desarrollar las aplicaciones de este trabajo de tesis, precio accesible y ofrece otras características deseables para futuros proyectos que se quieran desarrollar:

- **Compatibilidad:** Es compatible con MATLAB, App Designer y otros softwares de desarrollo de aplicaciones o adquisición de datos para futuros proyectos domóticos o no domóticos.
- **Comunicación:** La comunicación entre la tarjeta y el equipo de cómputo puede hacerse de forma alámbrica (Serial mediante los puertos USB) o inalámbrica a través de Bluetooth® o Wifi, tomando en consideración los aditamentos y configuraciones marcadas por MATLAB para hacer la comunicación.
- **Información:** Se puede encontrar información en libros, artículos en formato físico o digital, así como videos o bien desde su página oficial. La información es abundante en comparación con las de otras tarjetas de desarrollo.
- **Microcontrolador:** Cuenta con un microcontrolador de arquitectura AVR ATmega328P. Hay placas Arduino UNO con el chip de montaje superficial y otras con el encapsulado tipo DIP, por tanto, si se cuenta con la tarjeta con encapsulado DIP se puede desmontar el microcontrolador y este utilizarlo en algún proyecto que

solo requiera el chip y posteriormente volverlo a montar a la placa, configurarlo y seguir utilizando la tarjeta de desarrollo.

- **Programador:** Puede ser utilizado como un programador de microcontroladores con arquitectura AVR.
- **Precio:** De acuerdo con la Tabla 1, la tarjeta Arduino UNO es de las más económicas que pudieron cotizarse.
- **Fácil adquisición:** Puede adquirirse en la página oficial de Arduino, en alguna tienda dedicada a la venta de material electrónico o a través de las páginas para compras en línea, además que, al ser de software y hardware libre puede realizarse siguiendo los esquemáticos publicados en la página[8] o conseguir placas compatibles (mejor conocidas como placas clones).

1.4. Arduino

Arduino no solamente es una tarjeta de desarrollo, ya que de acuerdo con su página oficial: Arduino es una plataforma electrónica de código abierto basada en hardware y software fácil de usar.

Las tarjetas Arduino pueden leer entradas (luz en un sensor, un dedo en un botón o un mensaje de Twitter) y convertirlo en una salida, activando un motor, encendiendo un LED, publicando algo en línea. Puede indicar a su tablero qué hacer enviando un conjunto de instrucciones al microcontrolador en el tablero. Para hacerlo, utiliza el lenguaje de programación Arduino (basado en *Wiring*) y el Software Arduino (IDE), basado en *Processing*.

A lo largo de los años, Arduino ha sido el cerebro de miles de proyectos, desde objetos cotidianos hasta complejos instrumentos científicos. Una comunidad mundial de creadores (estudiantes, aficionados, artistas, programadores y profesionales) se ha reunido en torno a esta plataforma de código abierto; sus contribuciones se han sumado a una increíble cantidad de conocimiento accesible que puede ser de gran ayuda tanto para principiantes como para expertos.

[8] https://store.arduino.cc/usa/arduino-uno-rev3 buscando la sección nombrada: *Interactive Board Viewer*

Arduino nació en el *Ivrea Interaction Design Institute* como una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin experiencia en electrónica y programación. Tan pronto como llegó a una comunidad más amplia, la tarjeta Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta desde placas simples de 8 bits hasta productos para aplicaciones de IoT, *wearables*[9], impresión 3D y entornos integrados. Todas las tarjetas Arduino son completamente de código abierto, lo que permite a los usuarios construirlas de forma independiente y eventualmente adaptarlas a sus necesidades particulares. El software también es de código abierto y está creciendo gracias a las contribuciones de los usuarios de todo el mundo. [14]

Al utilizar la tarjeta original de Arduino consideramos que se debe contar un poco de su historia, ya que citando a Napoleón Bonaparte: "Aquel que no conoce su historia está condenado a repetirla".

Esto se menciona ya que hay tarjetas que catalogaremos como:

- Originales.
- Clones o compatibles con Arduino.
- Falsas.

La diferencia entre cada una de las tarjetas radicará primordialmente en el precio, diseño y materiales utilizados en la elaboración de estas, sin embargo, las placas falsas en muchas ocasiones llegan a ser vendidas al mismo precio que una original y los materiales utilizados para su fabricación carecen de calidad.

[9] Se relaciona con la oración en inglés *wearable technology* son dispositivos electrónicos inteligentes incorporados a la vestimenta o usados corporalmente como implantes o accesorios que pueden actuar como extensión del cuerpo o mente del usuario.

Para más información visita la página: https://es.wikipedia.org/wiki/Tecnolog%C3%ADa_vestible

1.4.1 ¿Cómo distinguir un Arduino UNO original de uno falso?

Por tal motivo se decidió dedicar un espacio en este trabajo para poder saber que es llamada falsa a todas las tarjetas que utilizan el nombre y marca de Arduino para ser distribuidas como tarjetas originales, sin embargo, no lo son. Hay muchas formas de comprobar si la placa adquirida es original (Fig. 1.8) o es falsa (Fig. 1.9), tales como:

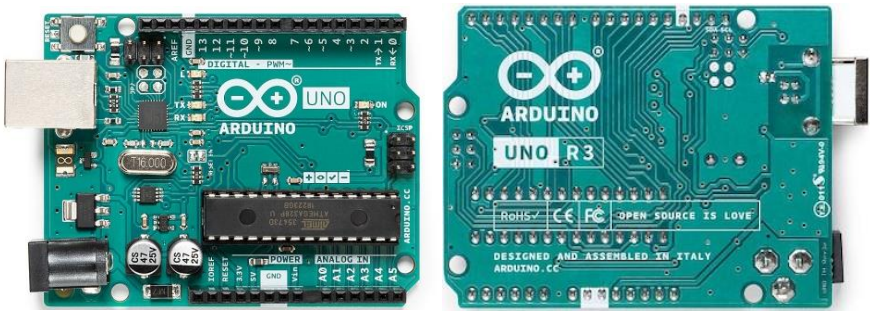
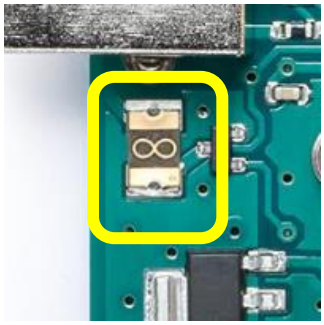


Fig. 1.8 Tarjeta Original Arduino UNO REV3 (parte frontal y trasera). [15]

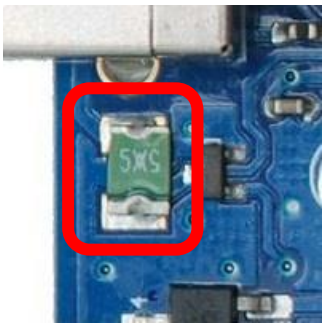
Fig. 1.9 Tarjeta falsa (parte frontal). [15]

El fusible PTC debe ser de un color negro y dorado personalizado, puede contar con la numeración “5040”, "501K", un dato similar impreso (Fig. 1.10 b) o el símbolo de infinito que representa a Arduino, sin embargo, esta característica se encuentra en las tarjetas más nuevas (Fig. 1.10 a). Mientras que, en una tarjeta falsa (Fig. 1.10 c), puede ser un componente genérico, de color verde o diferente y en muchas ocasiones el soldado que se realiza en esta parte puede carecer de calidad.



a) Fusible con símbolo de infinito.

b) Fusible personalizado “501K”.



c) Fusible genérico.

Fig. 1.10 a) y b) son las placas originales y c) es una placa falsa. [15]

Las diferencias contempladas entre una tarjeta Arduino UNO original y una clon en la elaboración de este trabajo fueron:

- **Comunicación entre software y hardware.**
  Al comunicar la tarjeta con MATLAB, ya que el puerto de comunicación serial (puerto COM) siempre será el mismo una vez que le fue asignado, a diferencia de la tarjeta clon, en la cual tendremos que consultar constantemente en que puerto COM se encuentra conectada para poder indicarlo en el programa.
- **Diseño impreso de las placas y sus componentes.**
  Esto puede ser un punto visual muy importante al momento de realizar las conexiones de los circuitos a la tarjeta Arduino UNO, ya que una tarjeta original Arduino cuenta con el grabado de los puertos en la parte plana de los headers hembra.
- **Calidad en los materiales utilizados.**
  Las tarjetas Arduino brindan garantía con sus componentes contra defectos de fábrica, esto quiere decir, que si un componente está dañado o en mal estado puede regresarse la tarjeta y ser reemplazada por una nueva o que cumpla con la calidad adecuada.

La calidad de los materiales utilizados en una tarjeta de desarrollo es primordial para la implementación de las aplicaciones y los sistemas electrónicos en cualquier lugar, ya que cuando se utilizan componentes electrónicos debería ser prioridad el tema de seguridad y asimismo proteger el entorno en el que uno se encuentra, por tal motivo, si la tarjeta utilizada cuenta con un diseño erróneo, hay una mala conexión entre sus terminales o no cumple con

las normas de seguridad se pueden llegar a dañar los dispositivos utilizados, pero sobre todo se pone en riesgo la integridad física de uno y de quienes nos rodean.

Arduino cuenta con otros modelos de tarjetas de desarrollo como pueden ser:

- MEGA.
- NANO.
- MICRO.
- LEONARDO.

Estos modelos cuentan con ciertas particularidades a diferencia de una tarjeta Arduino UNO en algunos casos el tamaño de la tarjeta es más grande o pequeña, el número de pines con los que cuentan, tipo de cable USB para realizar la comunicación, el modelo y tipo de encapsulado del microcontrolador, etc.

El motivo de elegir Arduino UNO REV3 sobre las otras tarjetas es que la adquisición de ella es más fácil, el costo es menor, los puertos con los que cuenta sirven adecuadamente para la implementación de las aplicaciones descritas en este trabajo, si se decidiera utilizar por ejemplo una tarjeta Arduino MEGA no se tendría ninguna diferencia respecto a las aplicaciones desarrolladas en este trabajo ya que las diferencias radican en el tipo de microcontrolador y número mayor de pines que tiene lo cuales no se aprovecharían al máximo en este momento.

La página oficial de Arduino proporciona la distribución de puertos establecidos en las placas Arduino UNO REV3 (Fig. 1.11), en ella se describen los puertos de uso digitales y analógicos, las fuentes de alimentación con las que se pueden contar (internas y externas), conexiones a tierra, LED soldado a la placa, etc. [16]

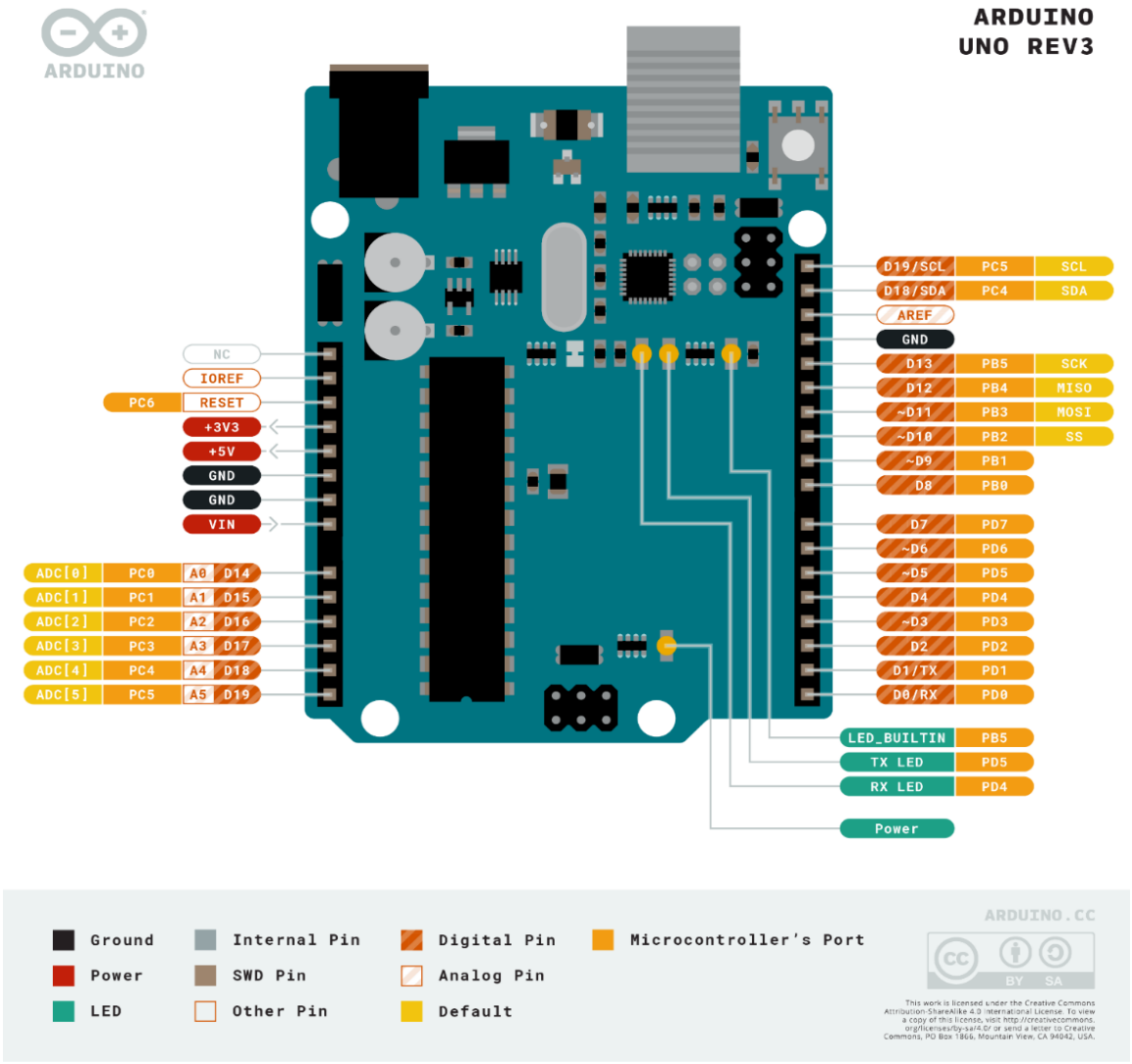


Fig. 1.11 Distribución de puertos en placa Arduino UNO REV3. [16]

CAPÍTULO 2

PREPARACIÓN DEL SISTEMA

2.1. Configurando MATLAB con Arduino UNO

Una vez que se eligió a Arduino UNO REV3 como la tarjeta de desarrollo a utilizar, se consideraron los elementos para trabajar y así poder dar inicio con la instalación del software. MATLAB cuenta con distintos paquetes para poder generar comunicación con diversos dispositivos, como: cámaras web, tarjetas de desarrollo, microcomputadoras, módulos electrónicos, etc.

A continuación, se muestra cómo realizar la configuración del software de MATLAB y la placa Arduino UNO se contó con:

- 1 computadora con MATLAB R2020b instalado.
- 1 tarjeta de desarrollo Arduino UNO REV3.
- 1 cable USB tipo A macho – tipo B macho.

Al iniciar MATLAB R2020b la imagen de inicio es la mostrada en la Fig. 2.1.

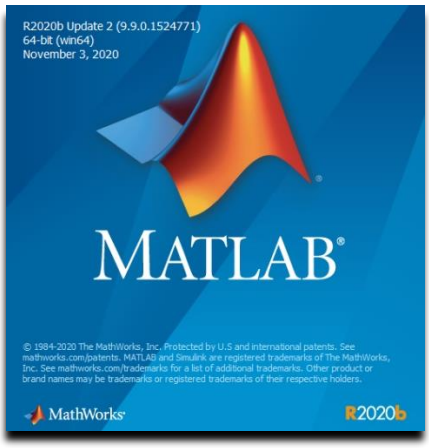


Fig. 2.1 Interfaz al iniciar MATLAB.

Una vez que el sistema cargó por completo se muestra en la parte inferior izquierda de la ventana de *Details* un mensaje con la palabra *Ready* tal como se muestra en la Fig. 2.2, y esto indicará que todos los complementos previamente instalados han sido cargados y que se puede utilizar el software sin ningún inconveniente.

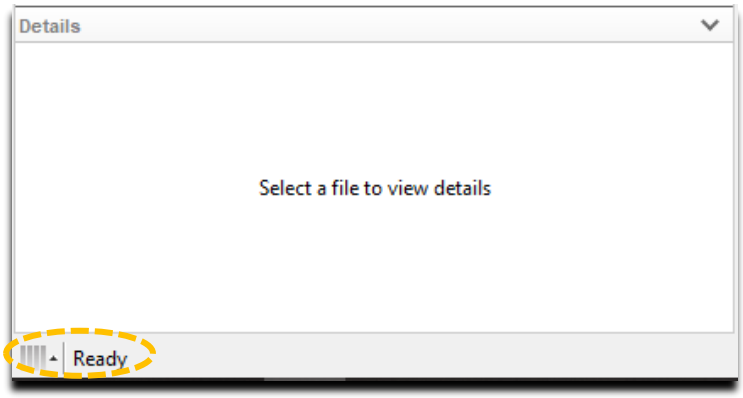


Fig. 2.2 Ventana de detalles en MATLAB.

Se verifica que el equipo de cómputo se encuentre conectado a internet y así poder descargar los paquetes que sean necesarios para realizar la configuración entre MATLAB y la placa Arduino UNO.

En la barra de herramientas se encuentra el ícono *Add – Ons*, al presionarlo se despliega un menú con 5 opciones, se selecciona la que diga: *Get Hardware Support Packages* (Fig. 2.3).

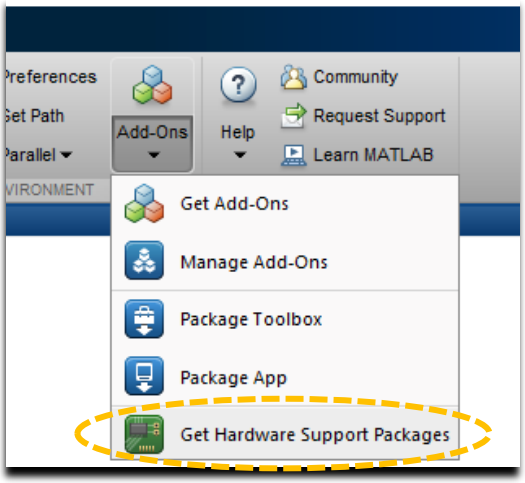


Fig. 2.3 Botón de acceso para agregar paquetes a MATLAB.

Y la página que se abre es: *Add-On Explorer* la cual contiene todos los paquetes que pueden ser descargados e instalados; para efectos de este trabajo con ayuda de la barra de búsqueda se escribe: *MATLAB Support Package for Arduino Hardware* (Fig. 2.4).

Fig. 2.4 Barra de búsqueda y botón de selección en forma de lupa.

Para seleccionar el paquete se da clic sobre la imagen o en el nombre marcado en negritas (Fig. 2.5).

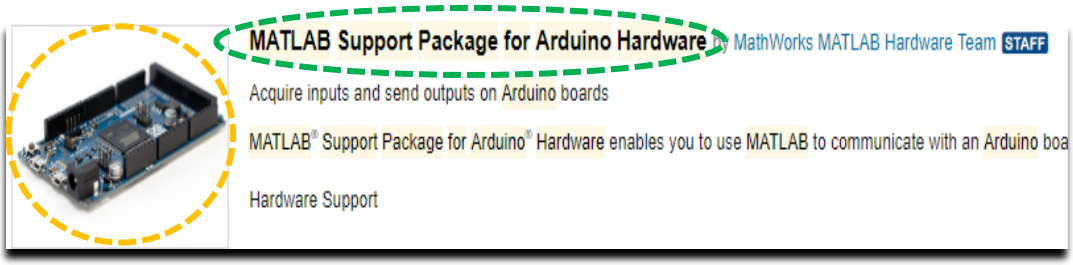


Fig. 2.5 Imagen para identificar el paquete a instalar. [17]

Se abrirá una ventana en donde puede observarse del lado derecho dos botónes en color azul, al darse clic en el botón *Install* se deben desplegarse dos opciones más, se debe presionar *Install* y así comenzará a realizarse la descarga e instalación del paquete (Fig. 2.6).

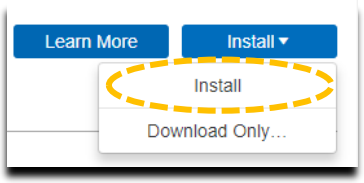


Fig. 2.6 Vista de botones para instalación y solo descarga del paquete.

Posteriormente aparece un recuadro llamado *MathWorks Auxiliary Software License Agreement* (Fig. 2.7), al darse clic en el botón *I Accept* se están aceptando los acuerdos de licencias de *MathWorks*, de no aceptarlos no se puede seguir con la instalación del paquete.

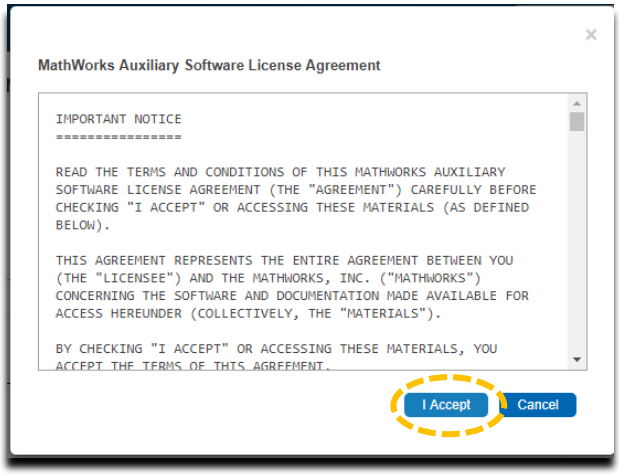


Fig. 2.7 Ventana *MathWorth Auxiliary Software License Agreement*.

Una vez que se aceptaron los acuerdos de licencia se procede a realizar la descarga e instalación. Para efectuar este paso primero se debe aceptar la instalación del software de terceros que se muestra en la Fig. 2.8.

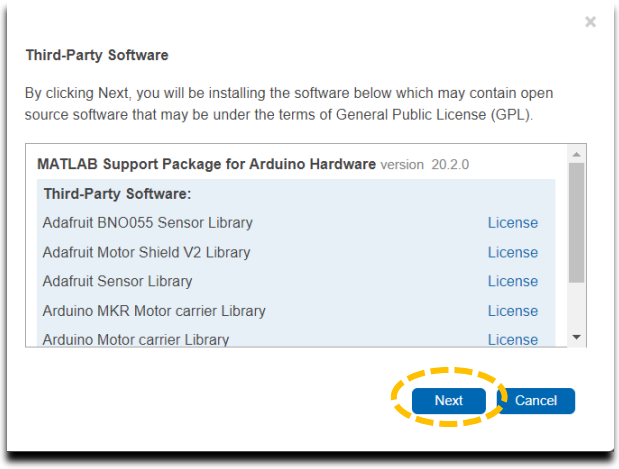


Fig. 2.8 Ventana *Third-Party Software*.

Posteriormente aparece una ventana pidiendo autorización para que la aplicación haga cambios en el dispositivo y para aceptarlo simplemente se da clic en el botón Sí (Fig. 2.9).

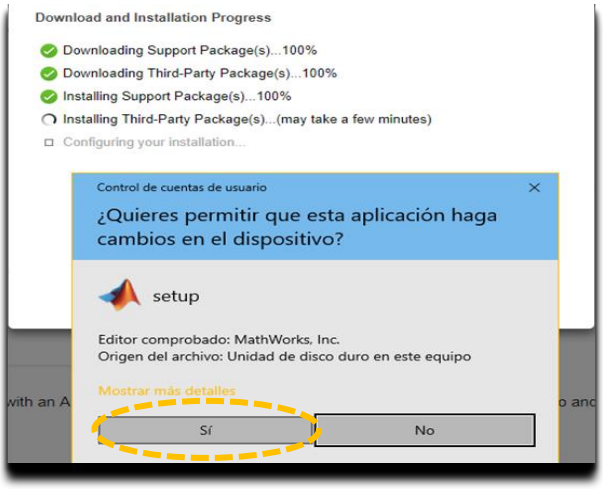


Fig. 2.9 Ventana de permisos para permitir cambios en el dispositivo[10].

Al finalizar la instalación aparecerá un mensaje indicando que la instalación está completa como se muestra en la Fig. 2.10, se da clic en el botón *Setup Now* y a partir de ese momento dará inicio con la instalación del controlador del puerto USB.

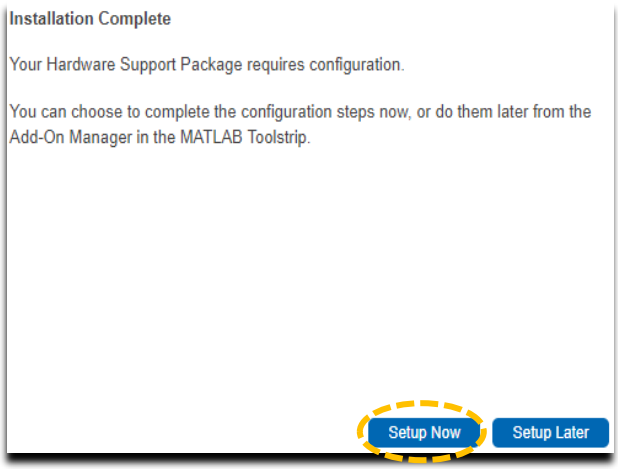


Fig. 2.10 Ventana de Instalación Completa del paquete.

En la ventana *Arduino USB Driver Installation* debe aparecer una casilla activada, esta corresponde a: *Enable installation of Arduino USB Driver*, en caso de no encontrarse

[10] Esta ventana puede o no aparecer de acuerdo con los niveles de seguridad establecidos en la configuración de control de cuentas de usuario en el sistema operativo Windows 10.

marcada la casilla se da clic sobre ella para activarla y posteriormente clic en *Next* para continuar con la configuración (Fig. 2.11).

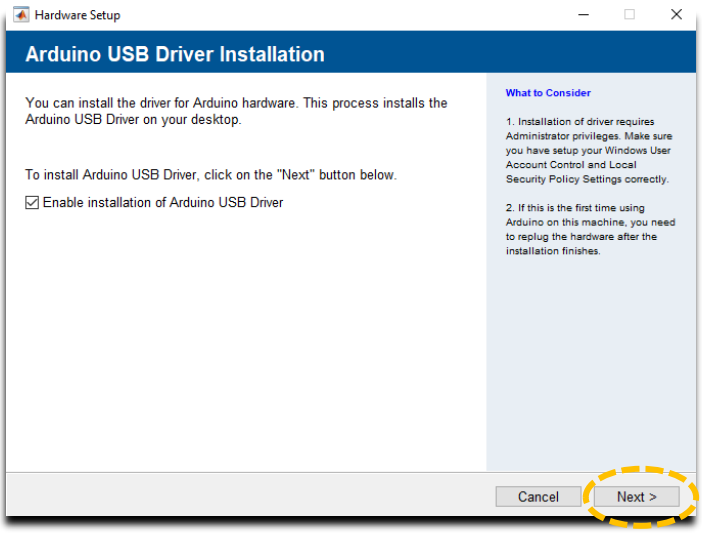


Fig. 2.11 Ventana *Arduino USB Driver Installation*.

Durante este proceso se mostrarán en total tres ventanas, las cuales solicitarán permisos para poder realizar cambios en el dispositivo, se aceptan o se dan los permisos correspondientes a todas las ventanas para poder continuar con el proceso.

Posteriormente se abrirá una ventana llamada *Arduino Hardware Setup* que preguntará: ¿quieres configurar la conexión con tu tarjeta Arduino?, se selecciona la casilla *Yes* y posteriormente se da clic en *Next* para continuar con la configuración tal como se muestra en la Fig. 2.12.

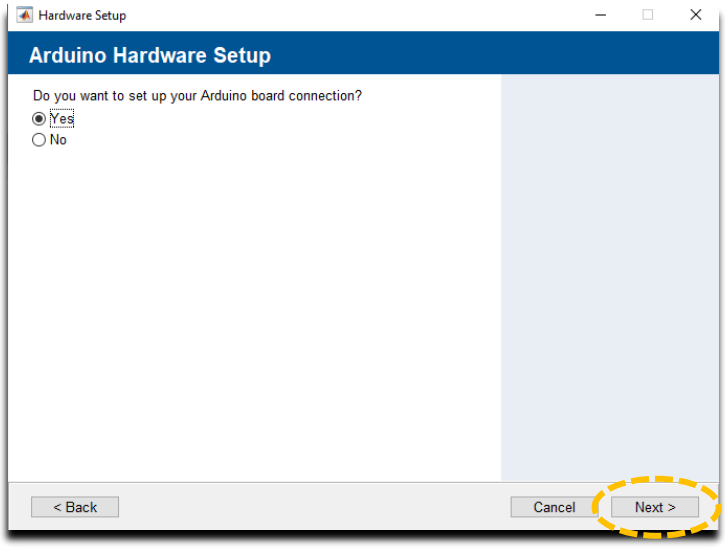


Fig. 2.12 *Arduino Hardware Setup*.

Las opciones que brinda MATLAB para poder hacer la comunicación con Arduino UNO son tres y pueden ser de forma alámbrica o inalámbrica, esto puede realizarse mediante:

- USB.
- Bluetooth®.
- WiFi.

De acuerdo con el tipo de comunicación que se decida utilizar tendrá sus propias especificaciones de configuración y puede requerirse la adaptación de otros dispositivos adicionales. Para las aplicaciones que se desarrollarán en este trabajo de tesis se optó por enviar los datos de forma serial (alámbrica), por tanto, se tiene que configurar y comunicar a MATLAB con la tarjeta de desarrollo a través del puerto USB, para ello, se conecta la tarjeta Arduino UNO a la computadora y se da clic en *Next* (en la Fig. 2.13 se muestran los tipos de conexiones a elegir y un esquema básico de la conexión entre la computadora y la tarjeta Arduino UNO).

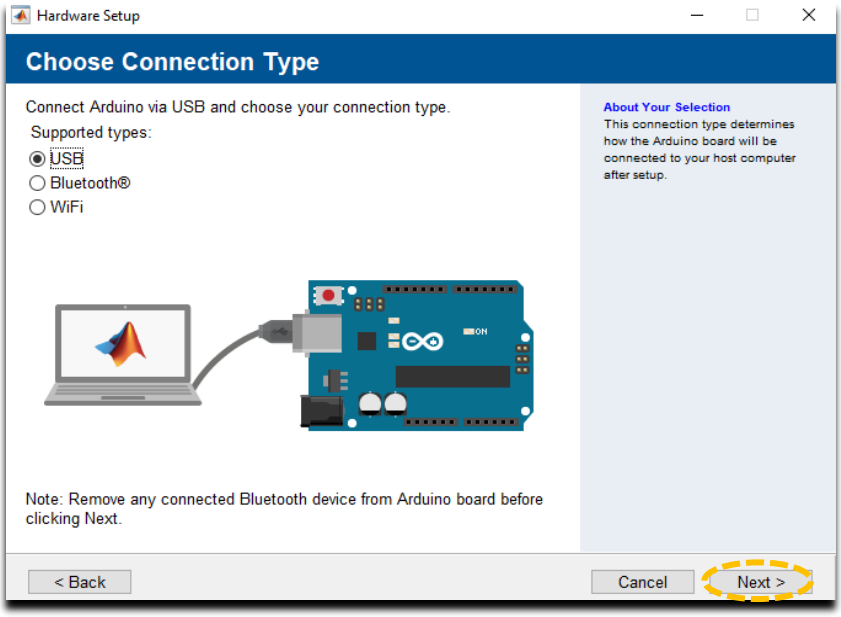


Fig. 2.13 Elección del tipo de conexión/comunicación.

En la ventana *Upload Arduino Server* (Fig. 2.16) se mostrarán las opciones:

- ***Choose board*:** Al dar clic en ella se desplegarán los modelos de tarjetas Arduino compatibles con este paquete. Se selecciona Uno (Fig. 2.14) ya que este es el que corresponde al modelo de la tarjeta utilizada.

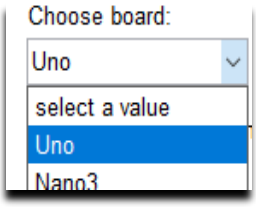


Fig. 2.14 Elección del modelo de tarjeta de desarrollo.

- ***Choose port*:** En este apartado se selecciona el puerto COM al que se encuentra conectada la tarjeta Arduino UNO; para este caso, el puerto al que se encuentra conectada es el puerto COM3 (Fig. 2.15), sin embargo, debe de considerarse que el puerto COM puede ser distinto al mostrado en este trabajo.

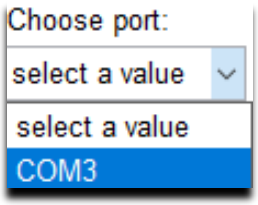


Fig. 2.15 Elección del puerto de comunicación.

- ***Select libraries to be included in the server*:** Se seleccionan las librerías que se desean instalar; es recomendable solo dejar aquellas que están marcadas en la Fig. 2.16.

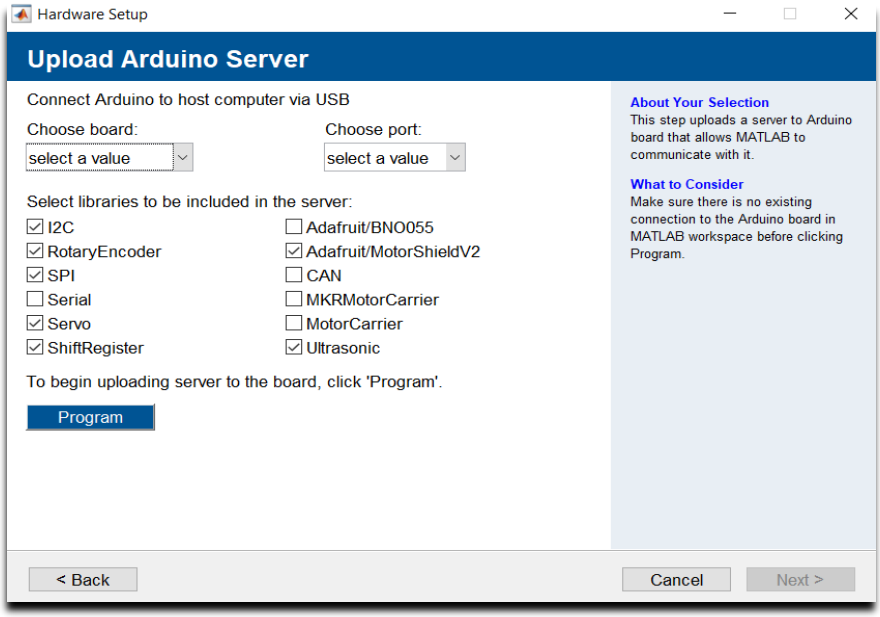


Fig. 2.16 Selección de librerías a instalar.

Después de seleccionar el modelo de Arduino, el puerto de comunicación y las librerías se procede a programar la tarjeta, para lograrlo se presiona el botón *Program*, aparecerá una barra de carga y se atenuará la ventana que se tenía mientras termina el proceso de programación de la tarjeta, al finalizar debajo del botón *Program* aparecerá un mensaje en color verde que dice: *Success! Click Next to proceed*. Se da clic en *Next* sin desconectar la tarjeta (Fig. 2.17).

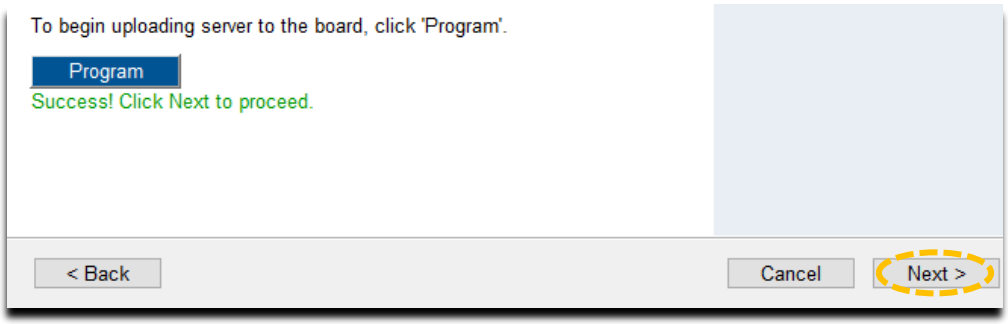


Fig. 2.17 Botón de programación de complementos a la tarjeta Arduino UNO.

Continuando con el proceso se mostrará el reporte de la configuración de la placa que se realizó, indicando el tipo de conexión, puerto, tarjeta conectada y librerías instaladas como se muestra en la Fig. 2.18.

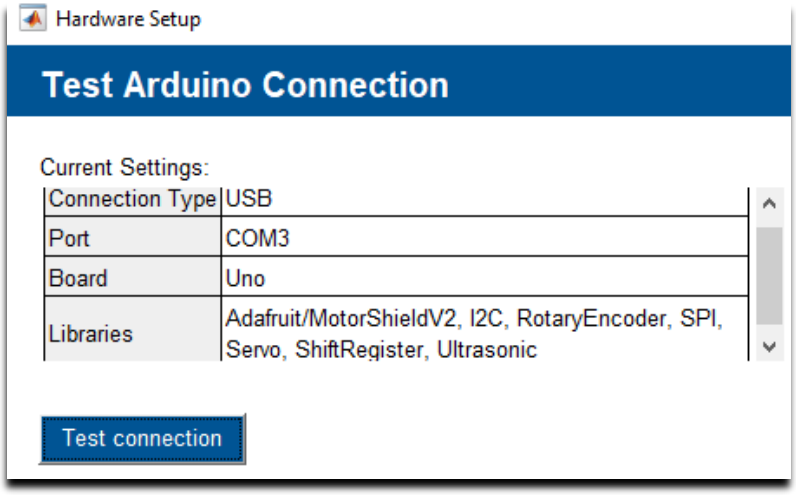


Fig. 2.18 Ventana de informes posterior a la programación de la tarjeta.

El botón *Test connection* (Fig. 2.18) permitirá hacer una prueba para validar que exista realmente la conexión entre la tarjeta Arduino UNO y MATLAB R2020b.

Al presionarlo y finalizar la prueba mostrará como en la Fig. 2.19 un mensaje que dice: Para usar tu tarjeta Arduino con las configuraciones actuales escribe: a = arduino. Para tarjetas Arduino no oficiales (clon), escribe a = arduino(*port*,*board*).

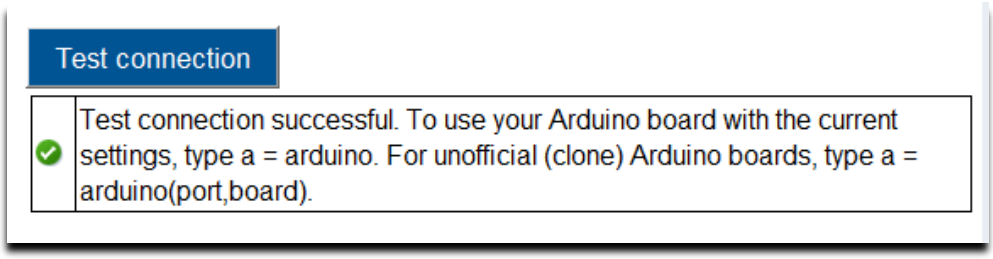


Fig. 2.19 Informe de conexión.

Se da clic en Next y aparecerá la ventana *Hardware Setup Complete* (Fig. 2.20), en ella se encuentra una casilla indicando que puede mostrar ejemplos relacionados con el paquete instalado, para este caso se desmarcó la casilla, sin embargo, si se deseará consultar los ejemplos que brinda MATLAB puede dejarse seleccionada, sin embargo, para continuar con el trabajo se quita la selección de la casilla y se siguió con el proceso dando clic en *Finish*.

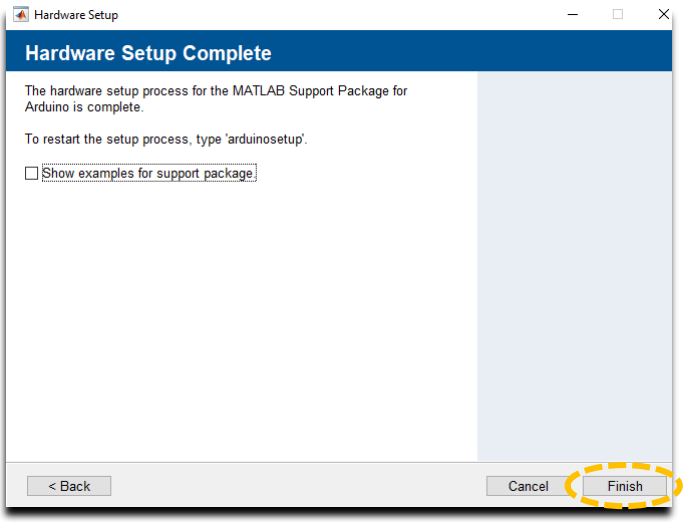


Fig. 2.20 Ventana de configuración completada.

Se verifica que en *Command Windows* (Fig. 2.21) aparezca la leyenda: "Arduino UNO detectado, este dispositivo está listo para usarse con *MATLAB Support Pakage for Arduino Hardware*. Inicia con ejemplo y otra documentación. Para utilizar este dispositivo con *Simulink*, instala *Simulink Support Package for Arduino Hardware*." Ya que esto indica que la comunicación entre software y hardware está establecida.

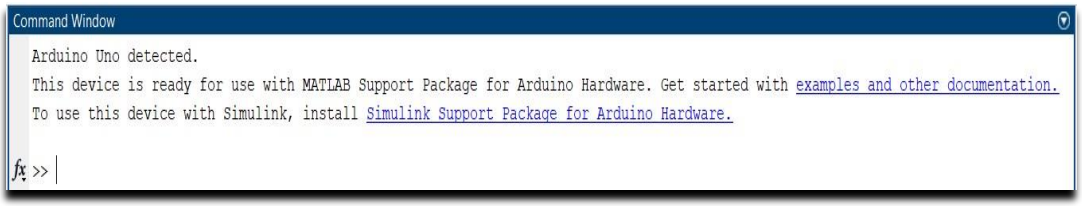


Fig. 2.21 Ventana de comandos en MATLAB indicando que se detectó un Arduino UNO.

Por último, en *Command Windows* se escribe: a = arduino() y al presionar la tecla *enter* deberá desglosarse las propiedades de la placa de Arduino UNO que se configuró (Fig. 2.22):

- Puerto de conexión.
- Modelo de la placa.
- Pines que podemos utilizar.
- Pines Digitales.
- Pines válidos para PWM.
- Pines Analógicos.
- Bus de I2C.
- Librerías.

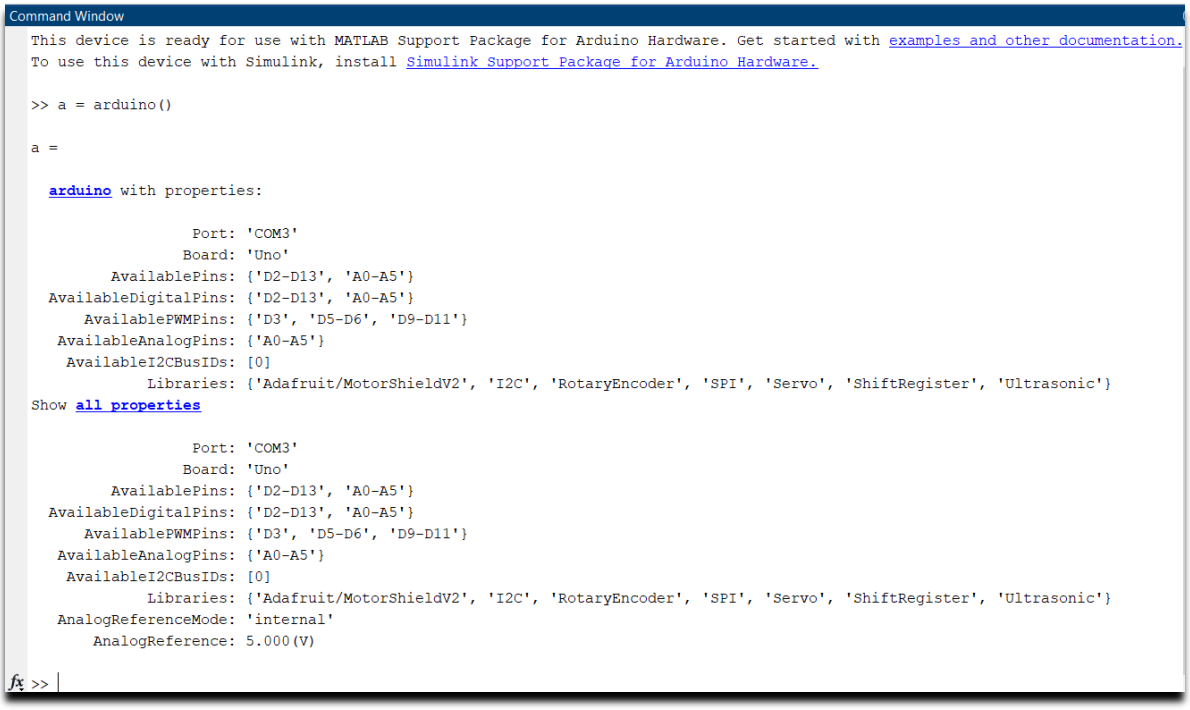


Fig. 2.22 Ejecución y respuesta del sistema al ejecutar el comando: a = arduino().

2.2. Iniciando con App Designer

Una vez que ha quedado configurada la comunicación entre MATLAB y la tarjeta Arduino UNO se ingresa a App Designer, este paso puede realizarse de dos formas distintas:

1. A través de *Command Window*, bastará con escribir la palabra *appdesigner* y pulsar la tecla *enter*. (Fig. 2.23)

Fig. 2.23 Ejecución de comando: appdesigner.

2. O bien, a través de la pestaña *HOME* de MATLAB, buscando y seleccionando el ícono *New* y dando clic en *App* (Fig. 2.24).

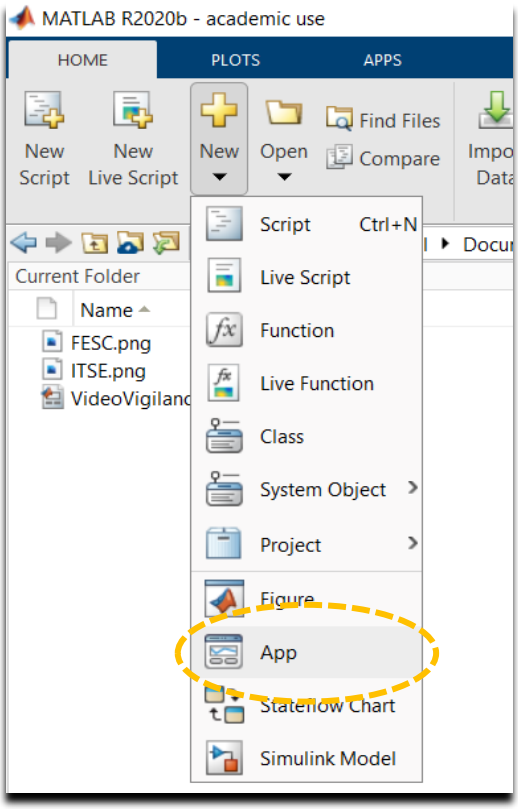


Fig. 2.24 Modo para ingresar a App Designer sin comando.

Se observará que se abre una ventana llamada App Designer (Fig. 2.25), esta ventana contiene una pestaña llamada *New*, al dar clic en ella se desplegarán varias opciones y se selecciona *Blank App* como se muestra en la Fig. 2.25.

Fig. 2.25 Ventana en App Designer para elegir un área de trabajo en blanco.

Una vez hecho esto, se abrirá el entorno de trabajo posicionado por defecto en la barra de herramientas en la pestaña *DESIGNER*, asimismo, estará activada la sección *Design View* (Fig. 2.26), siendo este el lugar en donde colocarán los elementos para la parte gráfica de la o las aplicaciones que se desean desarrollar.

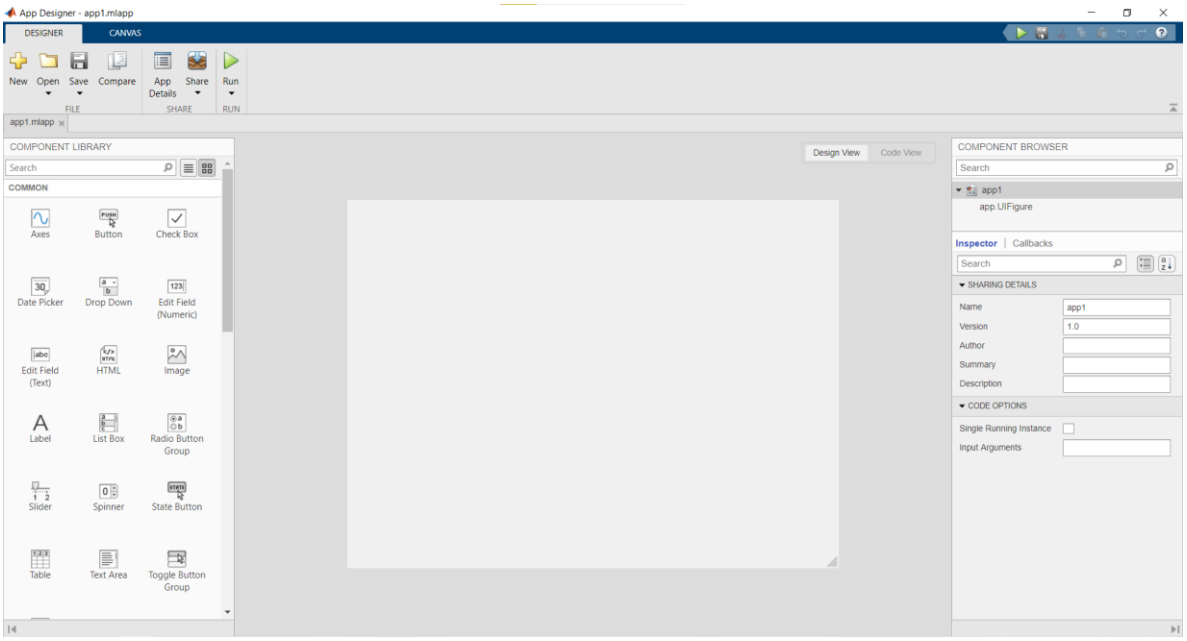


Fig. 2.26 Área de trabajo para la parte visual del programa.

Para lograr que los elementos de la interfaz de usuario que fueron colocados en el *Design View* ejecuten las acciones esperadas, se ingresa al *Code View* (Fig. 2.27) dando clic en el botón que lleva ese nombre, se abrirá el área de trabajo para visualizar e ingresar el código de funcionamiento con apoyo del *CODE BOWSER* → *Callbacks / Functions / Properties*.

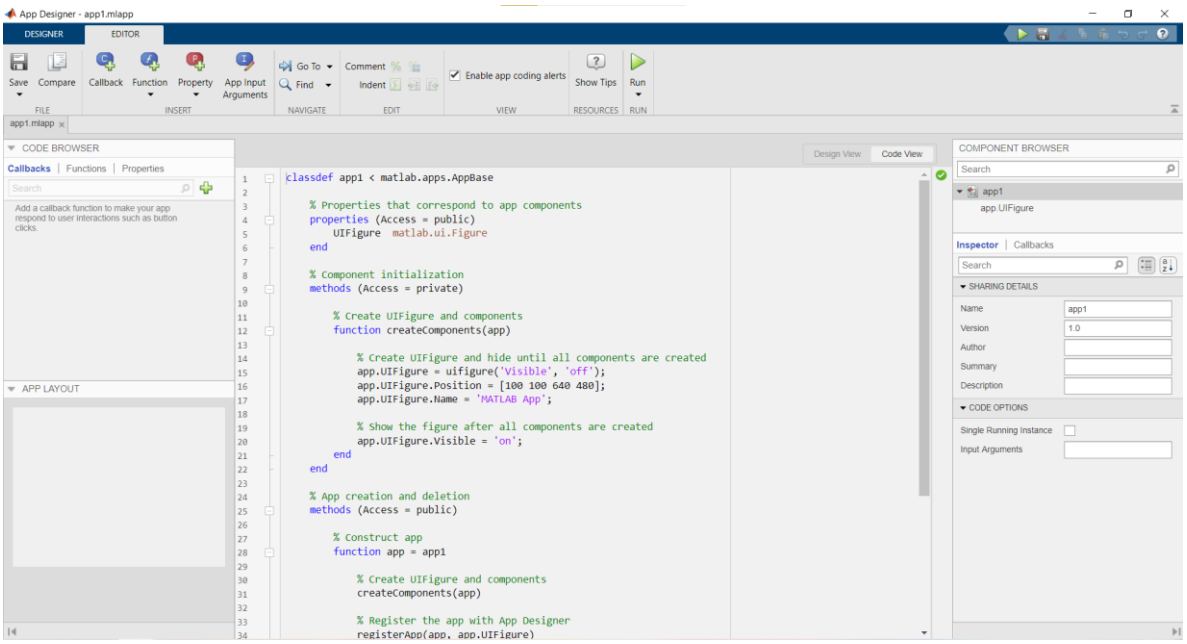


Fig. 2.27 Área de trabajo mediante código.

App Desiner cuenta con *COMPONENT LIBRARY* (Fig. 2.28), este contiene los componentes disponibles para el desarrollo visual de la aplicación como son: botones, editores de campo de texto o numérico, lámparas, perillas, interruptores, etc., y se encuentran clasificados en: *COMMON*, *CONTAINERS*, *FIGURE TOOLS*, *INSTRUMENTATION* y *AEROSPACE* [11].

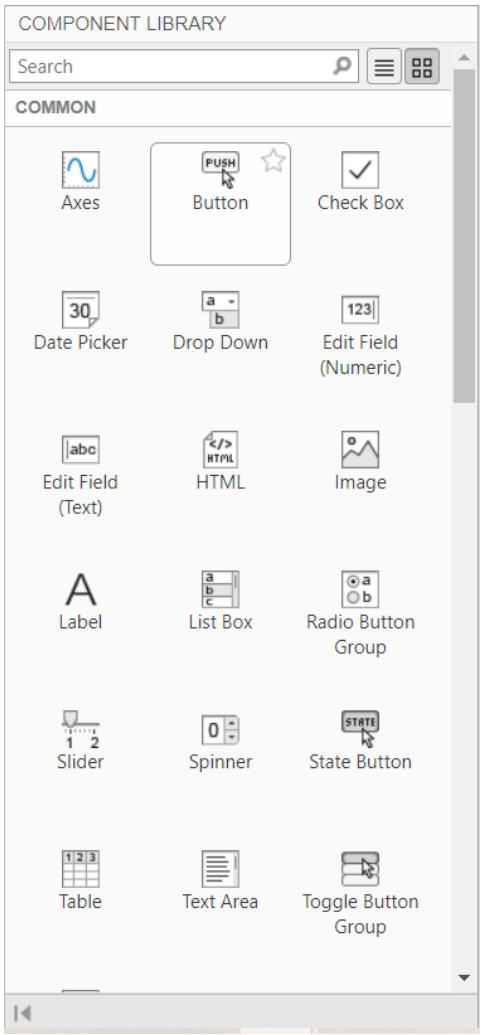


Fig. 2.28 Librería de componentes en App Designer.

Para poder añadir los elementos planeados bastará con ubicarlos en el *COMPONENT LIBRARY*, arrastrarlos y soltarlos al área de trabajo, por ejemplo: el componente *Lamp* se encuentra ubicado en *COMPONENT LIBRARY* → *INSTRUMENTATION* (Fig. 2.29 circulo

[11] Para la realización de este trabajo en ningún momento se utilizará *AEROSPACE*.

punteado color anaranjado), siguiendo los pasos anteriormente mencionados debería quedar el elemento en el área de trabajo de color gris (Fig. 2.29 recuadro punteado color verde).

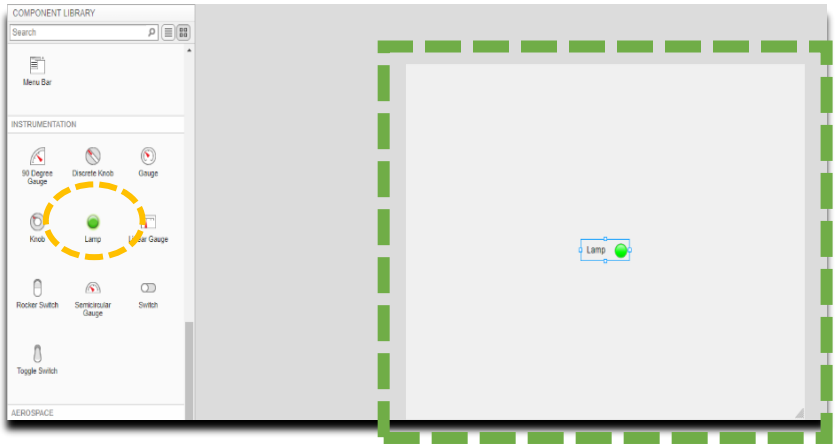


Fig. 2.29 Ejemplo al seleccionar y colocar un componente de tipo *lamp* en el área de trabajo.

Una de las ventajas que tiene App Designer es que permite modificar las propiedades del elemento (tamaño, color, nombre, etc.) desde el panel *COMPONENT BROWSER* y/o en algunos casos desde el mismo elemento.

Por ejemplo, considerando el elemento *Lamp* que se acaba de utilizar, si se quisiera agrandar su tamaño podría realizarse mediante los puntos marcados en las líneas azules que están rodeando el elemento o bien como se muestra en la Fig. 2.30 a través del *COMPONENT BROWSER* → *POSITION* → *Position*.

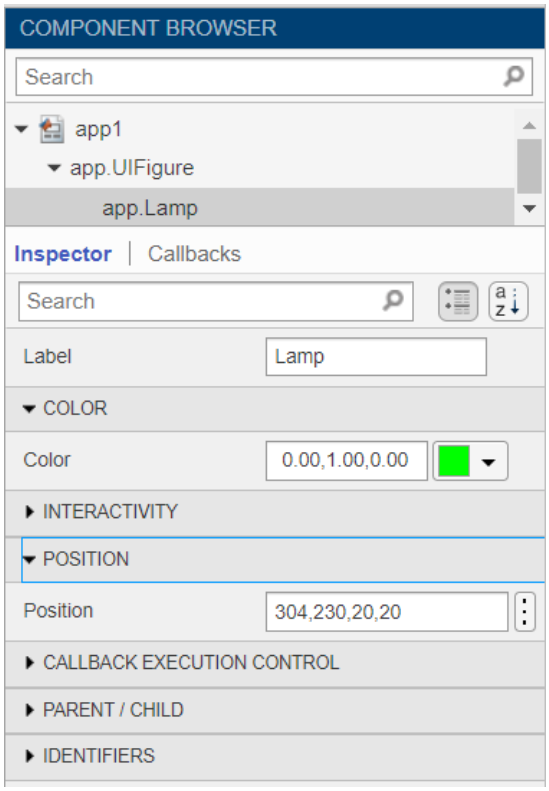


Fig. 2.30 Sección para editar propiedades de la aplicación.

Para ejemplificar este proceso se conservarán los dígitos “336,222” y se quitarán los dos últimos valores “20,20” (Fig. 2.31) y en su lugar se ingresan los valores “200,200” (Fig. 2.32).

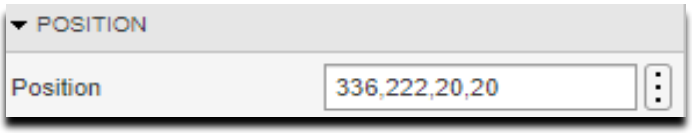


Fig.2.31 Valores actuales del objeto *Lamp*.

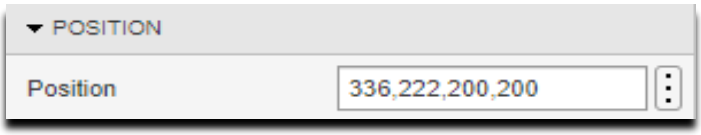


Fig. 2.32 Nuevos valores del objeto *Lamp*.

Como se observa en la Fig. 2.33 el nombre del elemento *Lamp* no fue modificado, sin embargo, el tamaño del ícono verde es considerablemente más grande a diferencia del mostrado en la Fig. 2.29.

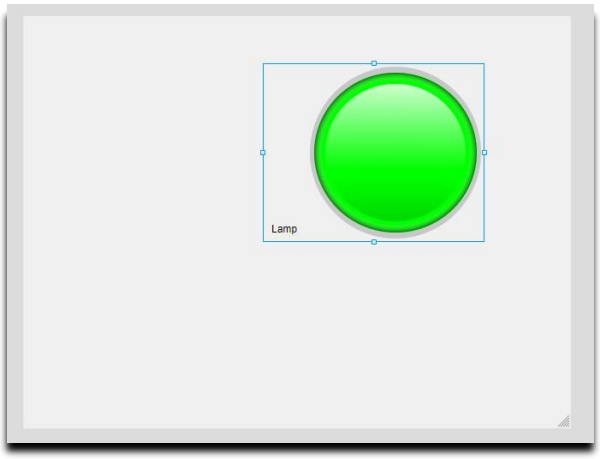


Fig. 2.33 Vista después de aplicar los cambios sugeridos.

De este modo es cómo se puede comenzar a desarrollar la interfaz de usuario y adecuando los componentes existentes a las necesidades del proyecto.

2.3. Flujo de proceso para desarrollar aplicaciones

Para efectuar el desarrollo de las aplicaciones conjuntas entre App Designer de MATLAB y la tarjeta Arduino UNO se debe seguir un proceso que permita la ejecución correcta de la aplicación, tal como se muestra en el siguiente esquema (Fig. 2.34):

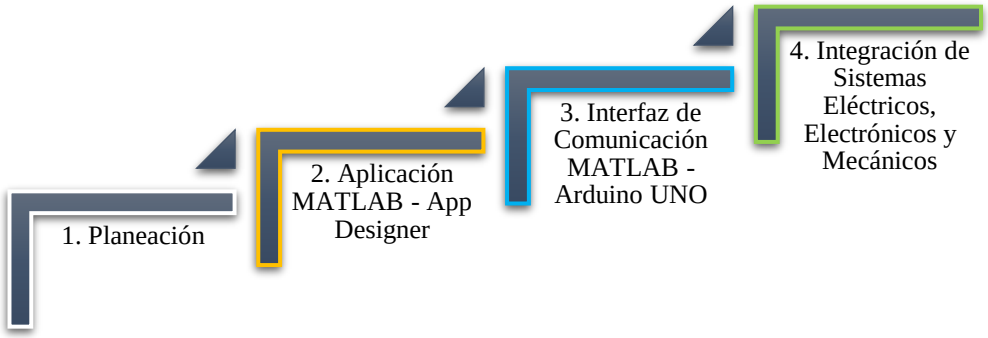


Fig. 2.34 Esquema de planeación.

Parte 1. Se eligió el sistema domótico a desarrollar considerando que los materiales a utilizar fueran de fácil adquisición, bajo costo y que el producto final permita al lector comprender el tema mostrado.

Parte 2. Al llegar a este punto se desarrolla la aplicación utilizando App Designer de MATLAB, generando los parámetros correspondientes para preparar la comunicación con la tarjeta de desarrollo.

Parte 3. En este apartado se debe de garantizar que la comunicación serial sea la adecuada, además de realizarse las interconexiones con los demás elementos del sistema que no tengan conexión directa con la computadora como: sensores, actuadores, relevadores, etc.

Parte 4. Aquí se realiza todo el acondicionamiento de los componentes eléctricos, electrónicos y mecánicos adicionales para garantizar el funcionamiento adecuado de la aplicación en conjunto del circuito electrónico.

En la Fig. 2.35 se muestra el flujo del funcionamiento en general del sistema, en donde el usuario deberá poder utilizar de forma natural la interfaz generada, es decir, no debe tener la necesidad o preocupación de contar con algún conocimiento técnico, dejando a un lado los aspectos que refieren a programación y comunicación entre software y hardware, cabe destacar que el "microcontrolador utilizado o tarjeta de desarrollo" será quien reciba las instrucciones generadas por la aplicación y éste envíe los datos a los dispositivos electrónicos.

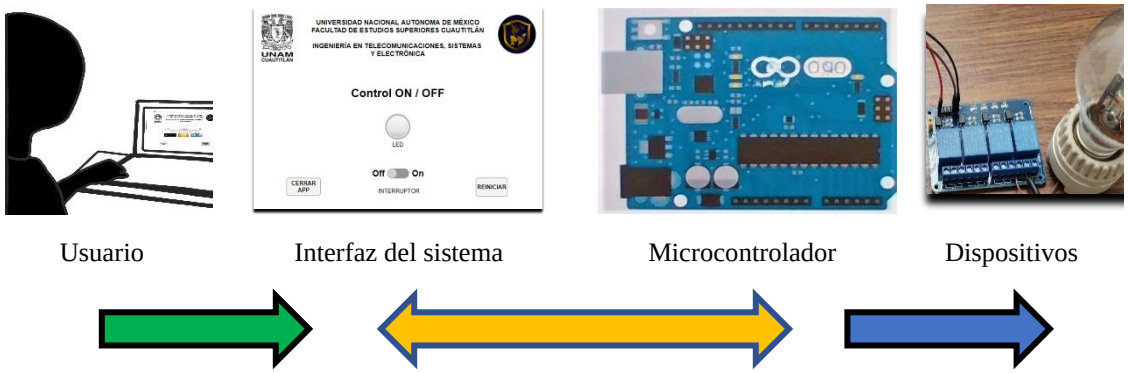


Fig. 2.35 Flujo del sistema.

CAPÍTULO 3

SISTEMAS DE AUTOMATIZACIÓN Y CONFORT LUMÍNICO

3.1. Control ON/OFF con un interruptor digital

Al momento de realizar la planeación de las aplicaciones, se puede considerar que uno de los puntos que no pueden dejarse a un lado en la domótica son los sistemas de automatización y confort lumínico, por ello, uno de los sistemas de control más conocidos cuando se habla del encendido y apagado de una luminaria en el ámbito domótico es el control ON/OFF de forma digital, sin embargo, en la mayoría de los hogares se puede observar que para controlar las luminarias se cuenta con interruptores electromecánicos instalados.

Ya sea que se utilice un interruptor digital o electromecánico ambos van a considerar únicamente los estados de operación “0 o 1”, es decir:

Apagado o encendido.
Cerrado o abierto.

Este tipo de sistemas que solo manejan estados de “1 o 0” se puede decir que está apegado a la lógica bivalente, esta lógica menciona que una proposición solo puede ser verdadera o falsa, por lo tanto, no existen valores intermedios, caso contrario a lo que pasa en la lógica difusa. Retomando los ejemplos anteriores, en este sistema de control no puede decirse que algo está:

Medio encendido o medio apagado.
80% encendido y 20% apagado.

Para poder comprender mejor el funcionamiento de un “Sistema de control ON/OFF con un interruptor digital” se desarrolló una aplicación con App Designer y se implementó un circuito sencillo tomando en consideración los siguientes materiales y equipos:

- 1 computadora con MATLAB R2020b instalado.
- 1 tarjeta de desarrollo Arduino UNO.
- 1 cable USB tipo A macho – tipo B macho.

- 1 LED RGB.
- 1 tableta de conexiones.
- Cables de conexiones.

Para comenzar con el desarrollo del proyecto se inicia App Designer y en la sección *Component Library*, se arrastran los elementos *Label*, *Lamp*, *Switch* y dos *Button* al área de *Design View* de modo que los elementos se muestren como en la Fig. 3.1.

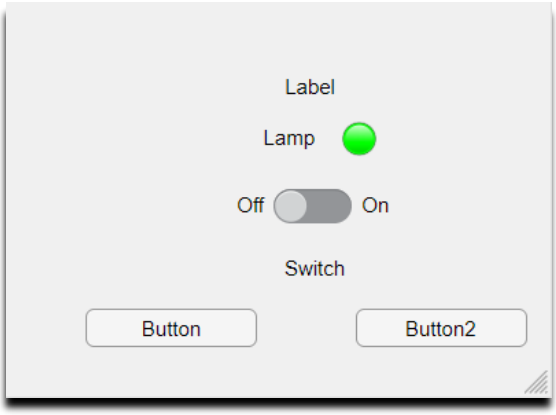


Fig. 3.1 Diseño para control del sistema de iluminación.

Una vez que se realizaron estos pasos, podemos dirigirnos a la pestaña nombrada como *Code View* y en ella se puede observar en forma de código (Fig. 3.2) los elementos que se colocaron en el *Design View* (Fig. 3.1).

```
classdef app1_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure     matlab.ui.Figure
        Button2      matlab.ui.control.Button
        Button       matlab.ui.control.Button
        Switch       matlab.ui.control.Switch
        SwitchLabel  matlab.ui.control.Label
        Lamp         matlab.ui.control.Lamp
        LampLabel    matlab.ui.control.Label
        Label        matlab.ui.control.Label
    end
```

Fig. 3.2 Propiedades generadas por MATLAB correspondiente a los objetos de la interfaz.

Aunque se pueden observar las propiedades y posiciones que tienen cada elemento que fue colocado, no pueden modificarse desde esta pantalla (Fig. 3.3), para realizar algún cambio se debe ir nuevamente a la pantalla de *Design View* y dar clic en el elemento a modificar siguiendo los pasos mostrados en el capítulo 2 cuando se abordó el tema "Iniciando con App Designer".

```
% Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'MATLAB App';

            % Create Label
            app.Label = uilabel(app.UIFigure);
            app.Label.Position = [324 328 35 22];

            % Create LampLabel
            app.LampLabel = uilabel(app.UIFigure);
            app.LampLabel.HorizontalAlignment = 'right';
            app.LampLabel.Position = [307 280 35 22];
            app.LampLabel.Text = 'Lamp';

            % Create Lamp
            app.Lamp = uilamp(app.UIFigure);
            app.Lamp.Position = [357 280 20 20];

            % Create SwitchLabel
            app.SwitchLabel = uilabel(app.UIFigure);
            app.SwitchLabel.HorizontalAlignment = 'center';
            app.SwitchLabel.Position = [322 181 41 22];
            app.SwitchLabel.Text = 'Switch';

            % Create Switch
            app.Switch = uiswitch(app.UIFigure, 'slider');
            app.Switch.Position = [319 218 45 20];

            % Create Button
            app.Button = uibutton(app.UIFigure, 'push');
            app.Button.Position = [198 146 100 22];

            % Create Button2
            app.Button2 = uibutton(app.UIFigure, 'push');
            app.Button2.Position = [385 146 100 22];
```

```
            app.Button2.Text = 'Button2';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = app1_exported

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

Fig. 3.3 Código generado por MATLAB que contiene las propiedades de los elementos usados.

Al dar doble clic sobre el nombre de los elementos podrá modificarse el texto o valores de los componentes, en esta aplicación se modificaron los tamaños de los elementos y se sustituyeron las palabras:

- *Label* por “Control ON / OFF”.
- *Lamp* por “LED”.
- *Switch* por “INTERRUPTOR”.
- *Button* por “CERRAR APP”.
- *Button2* por “REINICIAR”.

En la Fig. 3.4. se muestra el circuito que se armó, es importante mencionar que la tarjeta Arduino UNO no debe estar conectada a la computadora o algún otro tipo de fuente de voltaje externa mientras se realizan las conexiones de los puertos con los dispositivos electrónicos, al considerar esta indicación se podrá prevenir algún tipo de accidente.

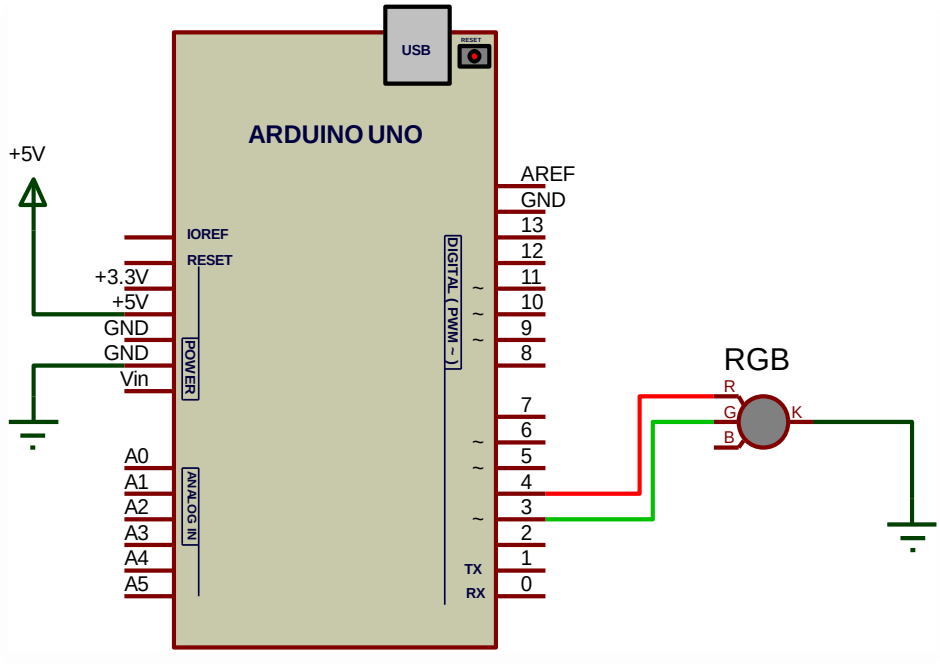


Fig. 3.4 Diagrama para la implementación física del sistema.

El modo para iniciar y guardar el contenido de una aplicación desarrollada en App Designer es presionando el botón *Run* (Fig. 3.5).

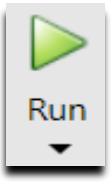


Fig. 3.5 Botón para ejecutar la aplicación.

La interfaz de usuario que se desarrolló es la que se muestra en la Fig. 3.6, mientras que, el circuito implementado con la aplicación iniciada es el que se muestra en la Fig. 3.7.

Fig. 3.6 Interfaz de control ON/OFF con interruptor digital.

Fig. 3.7 Implementación con un LED RGB y el programa iniciado.

Al dar clic sobre el INTERRUPTOR, se realiza el cambio de la posición Off a On y el LED de la aplicación cambiará su estado de gris a color verde (Fig. 3.8) mientras que el LED RGB conectado a la tarjeta Arduino realiza el mismo procedimiento físico (Fig. 3.9).

Fig. 3.8 Interfaz cuando el interruptor está en la posición On.

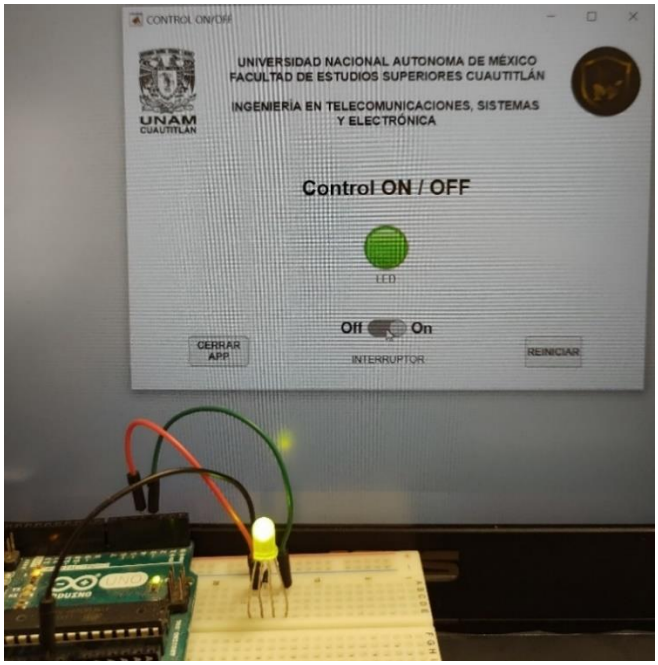


Fig. 3.9 Implementación del sistema: estado del interruptor = On.

Si se presiona nuevamente el INTERRUPTOR, el estado cambiará a Off, y como sucedió anteriormente el LED de la aplicación cambia, pero en este caso a color rojo (Fig. 3.10) y de igual manera pasa con el LED RGB conectado a la tarjeta Arduino (Fig. 3.11).

Fig. 3.10 Interfaz cuando el interruptor está en la posición Off.

Fig. 3.11 Implementación del sistema: estado del interruptor = Off.

El botón REINICIAR, tal como su nombre lo indica, reinicia la aplicación, apagando el LED mostrado en la aplicación y el LED RGB que se encuentra conectado en la tarjeta Arduino, generando que el sistema regrese a su estado inicial tal como se muestra en las Fig. 3.6 y la Fig. 3.7.

Para cerrar las aplicaciones bastaría con dar clic en el botón de "X" que viene por defecto en la aplicación, sin embargo, esto no garantiza que se borre la información guardada en alguna variable o puerto que se haya utilizado al momento de realizar la transferencia de datos, por ello, se decidió implementar el botón CERRAR APP. Una vez que se decide cerrar la aplicación el sistema finaliza toda la comunicación establecida con la tarjeta de desarrollo, sin importar si el INTERRUPTOR se encuentra en estado On o en Off.

3.2. Control ON/OFF implementando un LDR

Como fue mostrado en la sección anterior, el control ON/OFF con interruptor digital es muy similar a los interruptores electromecánicos, en donde van a prevalecer únicamente dos estados lógicos que representaron lo siguiente:

0 lógico = 0V = foco apagado.
1 lógico = 5V[12] = foco encendido.

Retomando este concepto se sustituirá el interruptor digital por un sensor LDR (*Ligth-Dependent Resistor*), también nombrado fotorresistor; este sensor se encarga de ir cambiando el valor de su resistencia ya que actúa como una resistencia variable de acuerdo con la cantidad de iluminación que va captando.

Al ser un sensor de tipo analógico hay una gran cantidad de valores a considerar, es decir, los datos que capte pueden encontrarse dentro del intervalo de 0V a 5V, algunas lecturas que pueden llegar a visualizarse son:

[12] El valor de 5V es considerado en este caso porque es el valor que otorga la fuente al LED, pero este valor puede cambiar de acuerdo con el voltaje implementado.

0.0150V, 1.3148V, 2.0001V, 3.0731V, 4.9609V, …, hasta llegar al límite.

Un ejemplo de la implementación de este sistema puede observarse en los hogares que requieren que se encienda alguna luminaria de forma automática en los momentos en los que la luz solar disminuye o desaparece por completo.

Este tipo de sistema de control asegurará que el área requerida se encuentre alumbrada eficientemente sin la necesidad de oprimir algún interruptor de forma manual.

La aplicación desarrollada en este tema pretende controlar el encendido y apagado de un foco de 100W, así como monitorear los datos capturados por el LDR ya que al cumplirse que el voltaje detectado por el LDR sea:

- **Menor o igual a 2V**
  El foco deberá encender, esto quiere decir que, a menor luminosidad detectada, menor será la resistencia del LDR.

- **Mayor a 2V**
  El foco deberá permanecer apagado, esto quiere decir que, a mayor luminosidad detectada, mayor será la resistencia del LDR.

¿Por qué elegir el valor de 2V?

Durante tres días se hicieron mediciones dentro y fuera de una casa con el fin de tratar de detectar el mejor momento para alumbrar la sala y el pasillo de la entrada principal a la casa, la lectura que aproximadamente siempre coincidió fue la de 1.48 V. Se decidió redondear este valor considerando que es mejor encender el alumbrado antes para no percibir el cambio de luz tan repentino, sin embargo, este valor puede modificarse de acuerdo con las necesidades de cada área de instalación o gusto de las personas.

Para el control del sistema con un LED basta con colocar el diodo en serie a una resistencia, un pin iría a GND y el otro al puerto digital asignado para escribir en él los datos programados.

Si se deseara sustituir el LED por un foco ahorrador o con tecnología LED es necesario implementar un sistema de potencia, que sea capaz de poder manejar la carga de CA y al mismo tiempo proteger nuestro microcontrolador y el equipo de cómputo, por ello, en este capítulo se darán a conocer los componentes electrónicos que componen al módulo de relevadores utilizado (Fig. 3.12).

El material y equipo utilizado para la implementación de este sistema fue:

- 1 computadora con MATLAB R2020b instalado.
- 1 tarjeta de desarrollo Arduino UNO.
- 1 cable USB tipo A macho – tipo B macho.
- 1 foco de 100W.
- 1 portalámpara cerámico.
- 1 clavija con cable.
- 1 LDR.
- 1 resistencia de 10kΩ.
- 1 módulo de Relevador.
- 1 tableta de conexiones.
- Cables para conexiones.

En la Fig. 3.13 se muestra la interfaz de usuario, en donde se podrá ir monitoreando los datos que lea el LDR, para poder iniciar la lectura se debe hacer clic en el botón INICIAR LECTURA, la creación de este método de inicio es para tener un control de inicio únicamente, sin embargo, se puede quitar y el funcionamiento no debería de afectar.

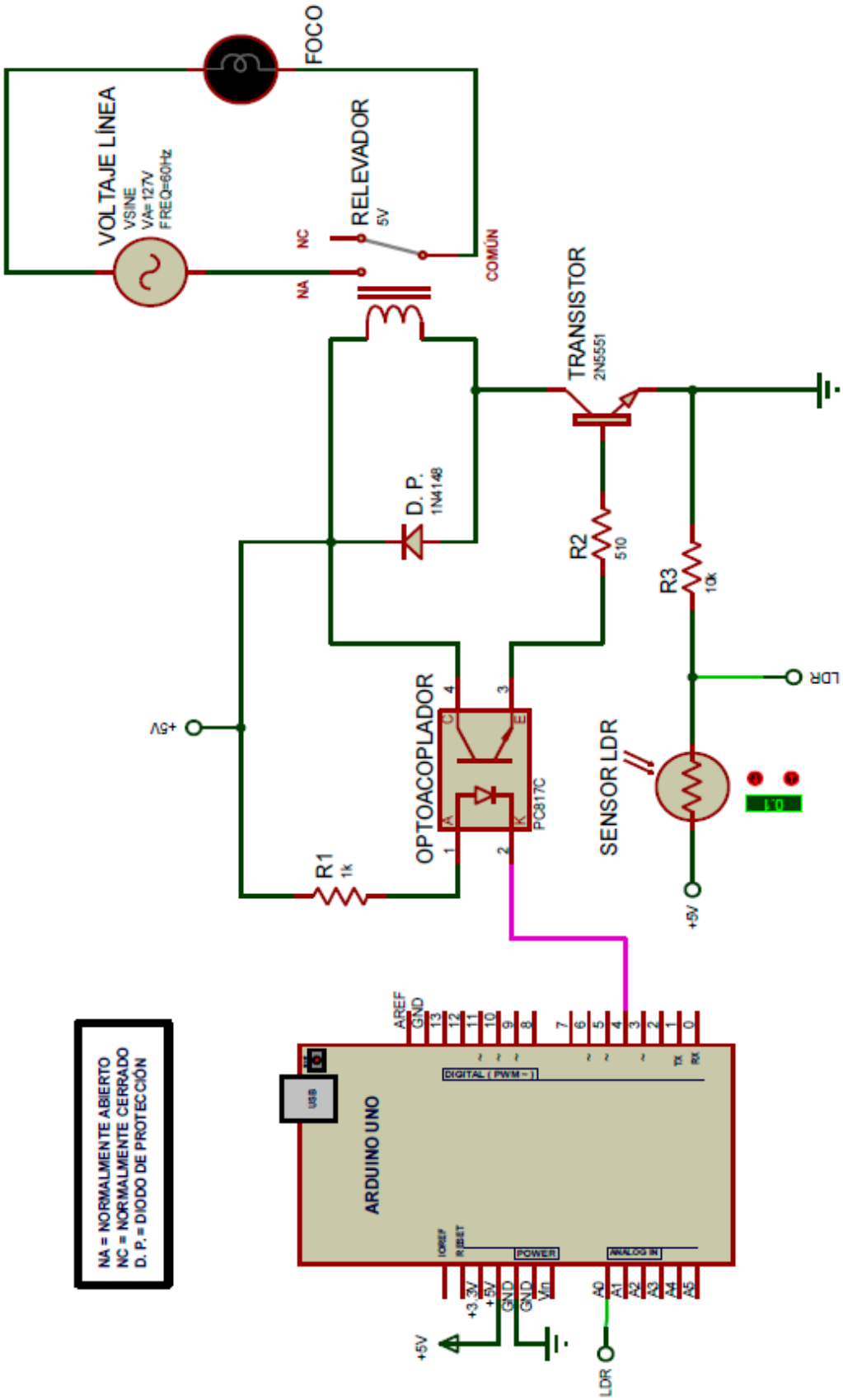


Fig. 3.12 Diagrama de conexiones con elementos del módulo de relevador desglosado.

Fig. 3.13 Interfaz de usuario antes de activar el sistema.

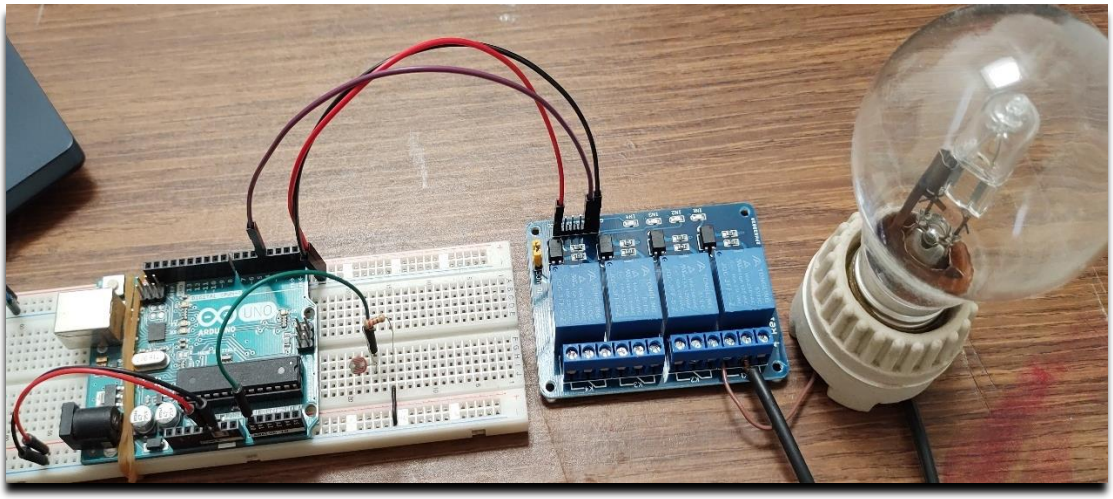


Fig. 3.14 Implementación física conforme al circuito (sistema sin iniciar).

La Fig. 3.14 muestra el circuito armado sin estar conectado aún a la corriente eléctrica y sin iniciar las lecturas con la aplicación. Una vez que el botón de INICIAR LECTURA se presionó, el sensor LDR comenzará a realizar las lecturas correspondientes; podrá observarse que a mayor luminosidad, mayor es el valor del voltaje captado, tal como se muestra en la Fig. 3.15, lo cual genera que el foco conectado al sistema permanezca apagado.

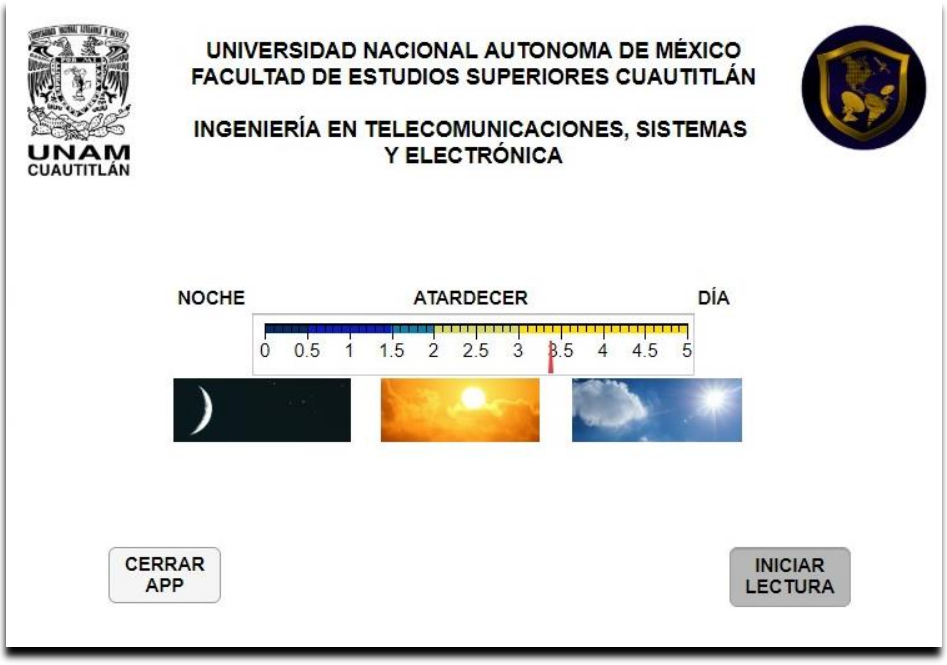


Fig. 3.15 Interfaz en funcionamiento. Lectura de 3.37V.

El programa fue diseñado para que encienda el foco una vez que el sensor LDR detecte los valores iguales o menores a 2V, en este caso, la luminosidad detectada en el cuarto de pruebas fue de 1V tal como se muestra en las Fig. 3.16 y la Fig. 3.17.

Fig. 3.16 Interfaz en funcionamiento. Lectura de 1V.

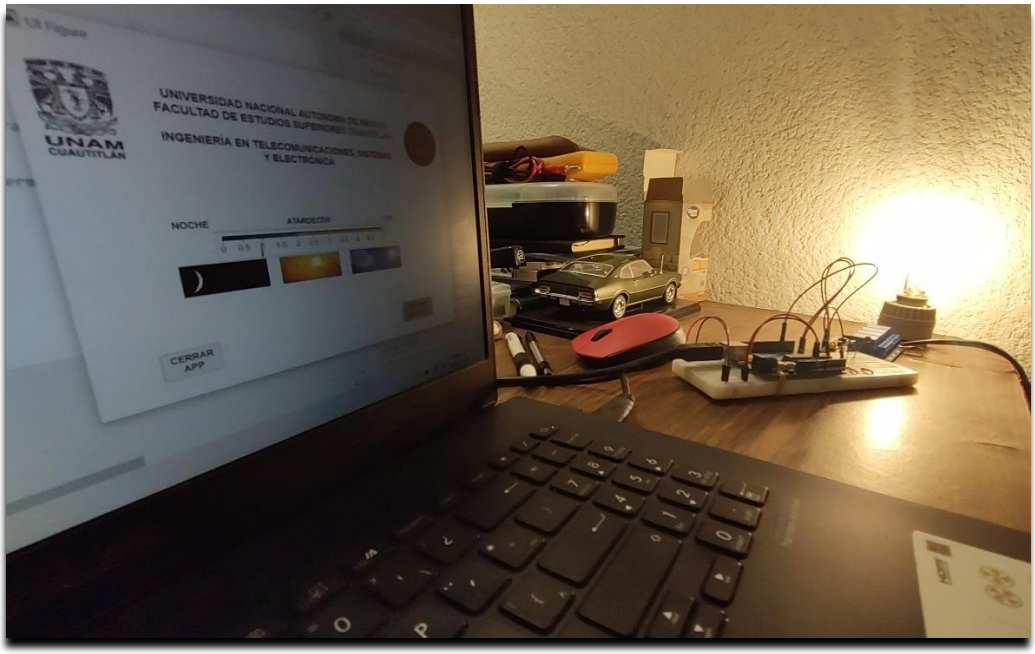


Fig. 3.17 Implementación física del sistema cuando el valor captado por el LDR es de 1V.

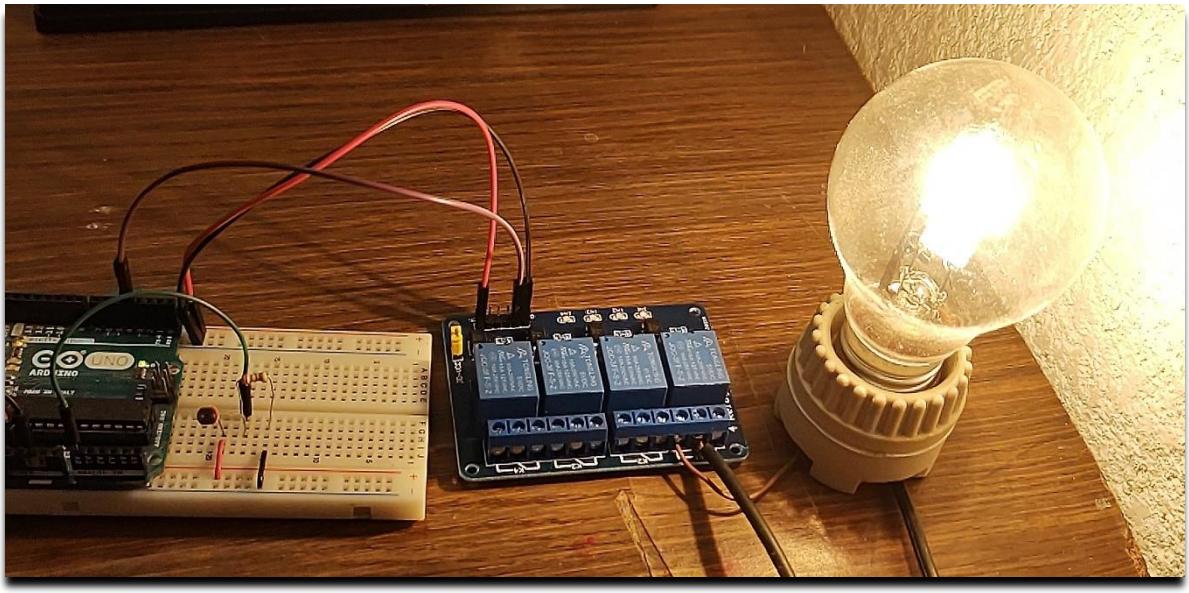


Fig. 3.18 Implementación física. Lectura de 1V y foco activado.

Algo muy importante a considerar al implementar este tipo de sistemas de control de iluminación, es que el sensor LDR siempre debe estar direccionado en un punto donde la luminosidad detectada sea únicamente la del ambiente (Fig. 3.18), ya que si se pusiera el LDR en dirección a otra luminaria este detectaría esa luminosidad generada y si esta es mayor

al valor programado provocará que el foco no encienda, por consiguiente el área que se requiera iluminar probablemente no logre ser alumbrada; considerando esta situación, se puede decir que ésta es una de las desventajas de implementar este tipo de sensores.

Otro punto por tomar en cuenta es que no siempre se tienen lecturas exactas, por tanto, aunque el sensor muestre una lectura de 1.99V puede ser que en un segundo detecte 2V ocasionando que haya intermitencia en la activación, ocasionando que el foco encienda y apague de acuerdo con los valores detectados, para solucionar o evitar esta situación se debe establecerse una condición al momento de realizar la programación la cual permita estabilizar el sistema brindándole un rango de valores que tengan prioridad sobre la activación del encendido del foco.

3.3. Regulación de intensidad luminosa con un dimmer digital

Además de los sistemas de control de tipo ON/OFF se cuenta con los sistemas de regulación de intensidad luminosa, es decir que entre menos voltaje medio se le proporcione al LED menor será su brillo y entre más voltaje medio tenga, el brillo del LED será mayor. Con el fin de cumplir con estas condiciones se tendrá que realizar la regulación del voltaje de alimentación del circuito por medio de la Modulación por Ancho de Pulso (también conocida como PWM por sus siglas en inglés *Pulse-Width Modulation*).

La modulación por ancho de pulsos o PWM es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

Para lograr este efecto en la aplicación que se desarrollará se pueden utilizar distintos dispositivos electrónicos. Para este trabajo al utilizar la tarjeta Arduino UNO se debe saber que ya cuenta con salidas destinadas a cumplir este propósito en específico.

Para identificar los pines digitales con PWM basta con verificar que esté el símbolo de una virgulilla (~) a un costado del número de los puertos digitales correspondientes.

Los puertos que se describen a continuación serán los únicos que se pueden utilizar como salida PWM en la tarjeta Arduino UNO REV3:

- D3
- D5
- D6
- D9
- D10
- D11

El dispositivo electrónico más conocido para realizar la regulación de luminosidad es el dimmer, este dispositivo en la actualidad puede conseguirse con gran facilidad, hay dimmers que cuentan con una perilla giratoria, aquellos que pueden regularse mediante pulsaciones, otros que cuentan con un módulo para comunicación inalámbrica (wifi o bluetooth).

En este apartado se desarrolló una aplicación capaz de variar la intensidad de iluminación, para implementar el sistema se requirió de:

- 1 computadora con MATLAB R2020b instalado
- 1 tarjeta de desarrollo Arduino UNO
- 1 cable USB tipo A macho – tipo B macho
- 1 resistencia de 330Ω
- 1 LED
- 1 tableta de conexiones
- Cables para conexiones

Las conexiones electrónicas fueron con base en el circuito de la Fig. 3.19.

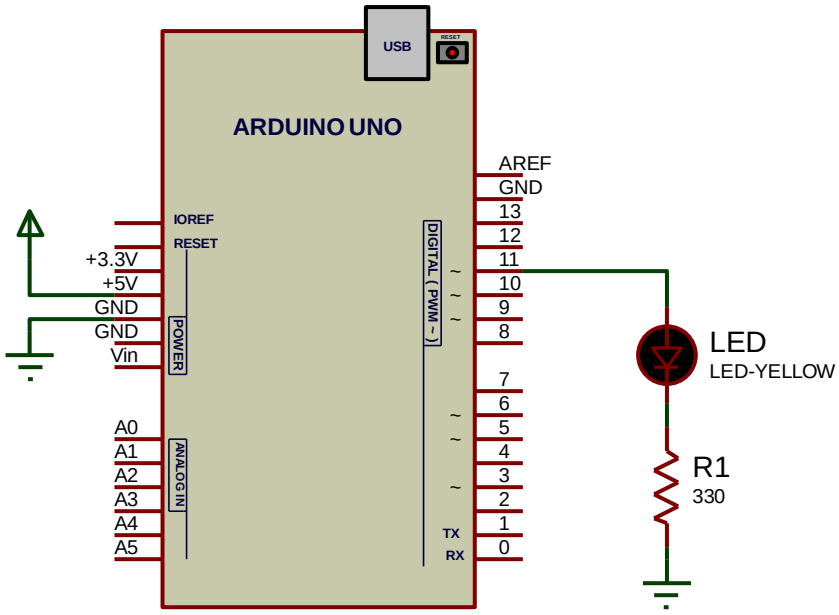
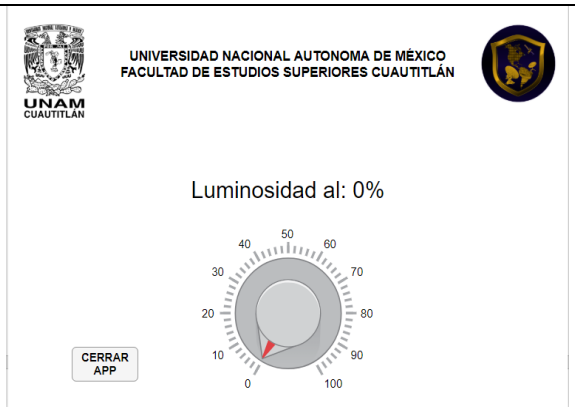
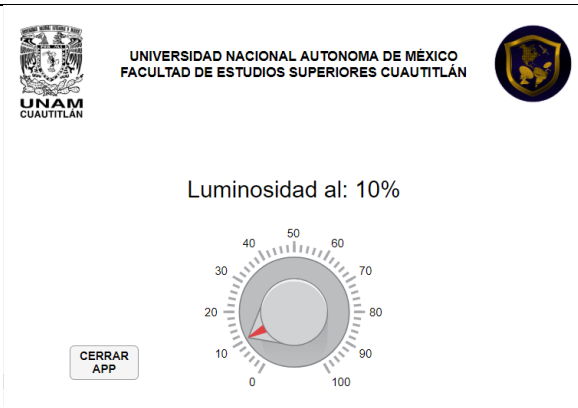
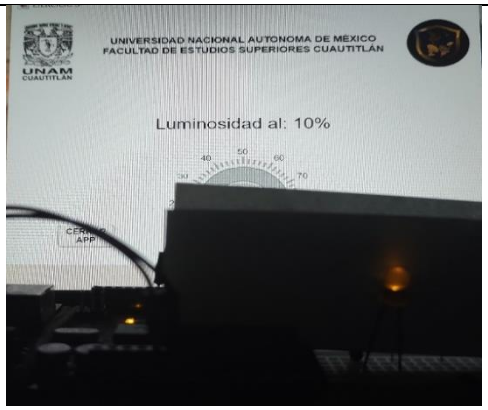
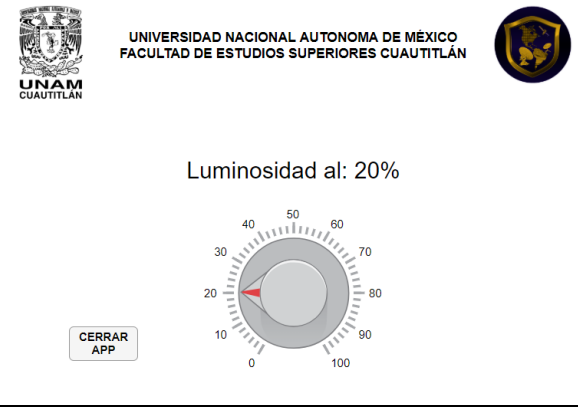
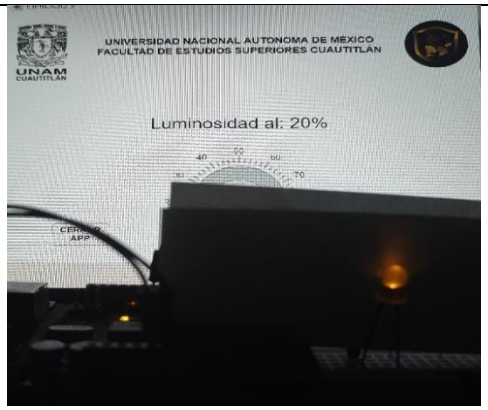
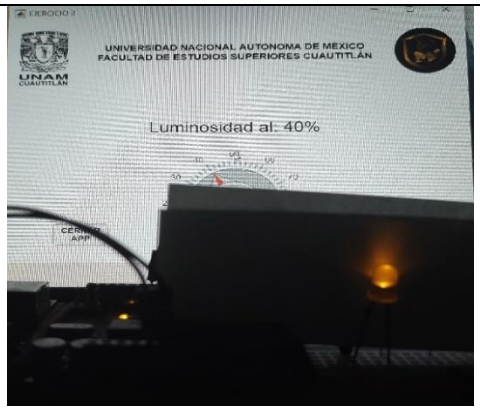
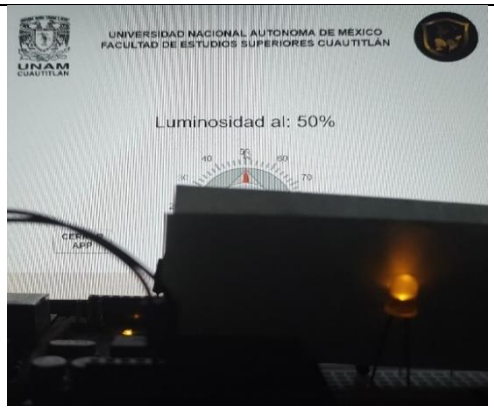


Fig. 3.19 Diagrama de conexión del LED con tarjeta Arduino UNO.

La interfaz de usuario permite manejar a través de un potenciómetro digital el ancho de pulso que llega al LED. Una vez que se comienza a girar la perilla, se observa que el texto "Regulación de intensidad luminosa" (Fig.3.20) cambia por "Luminosidad al: 0%", el número del porcentaje dependerá del valor que se esté otorgando a través de la perilla, estos valores van del 0% - 100% tomando en consideración valores decimales (Tabla 2).

Fig. 3.20 Interfaz de usuario de la aplicación antes de girar el potenciómetro.

APLICACIÓN	IMPLEMENTACIÓN
	
	
	

UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN Luminosidad al: 30% CERRAR APP	
UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN Luminosidad al: 40% CERRAR APP	
UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN Luminosidad al: 50% CERRAR APP	
UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN Luminosidad al: 60% CERRAR APP	

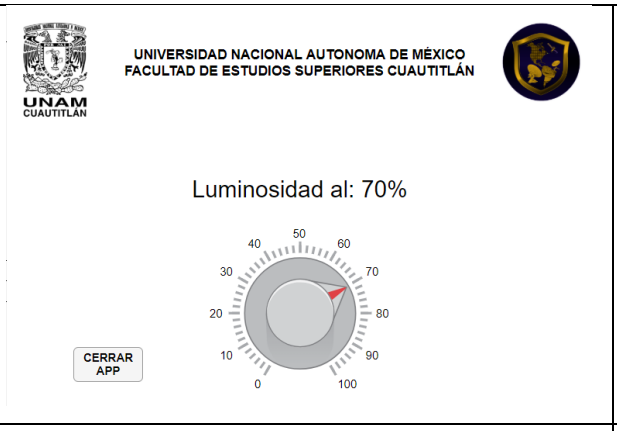
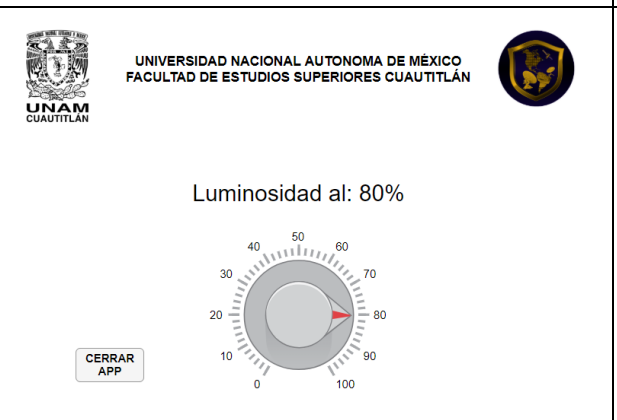
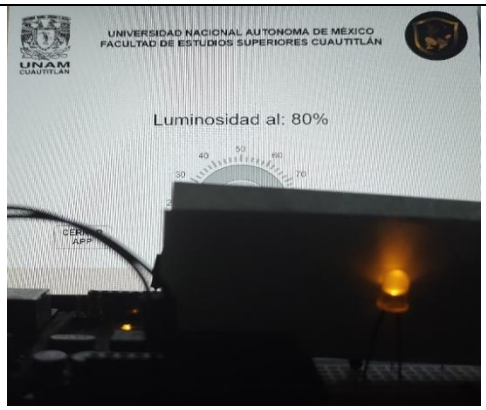
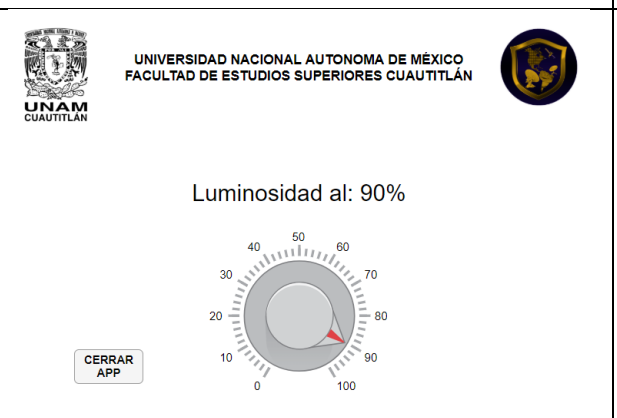
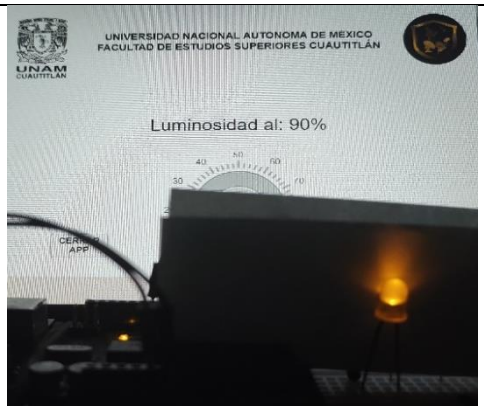
	
	
	
	

Tabla 2. Comparación del encendido del LED implementando la aplicación tomando valores del 0% al 100%.

Realizar el control de luminosidad de un LED mediante un dimmer digital implementando PWM no es tan complicado a diferencia de cuando se requiere implementar con un foco ahorrador o con tecnología LED, ya que en ese caso intervienen muchos factores electrónicos como son las etapas de potencia, detecciones de cruces por cero, programación, entre otros factores.

La aplicación aquí descrita permite tener un punto de partida para continuar con futuras implementaciones además del dimmer con un foco de cualquier tipo de potencia o tecnología. Podría implementarse un sistema capaz de manejar la climatización de algún lugar equivalente a lo que se llama en inglés *Heating, Ventilating and Air Conditioning* o por sus siglas HVAC o también se podría generar un sistema de control en la apertura de persianas o cortinas, en el cual se indique el porcentaje de apertura.

CAPÍTULO 4

SISTEMAS DE SEGURIDAD Y PROTECCIÓN

4.1. Seguridad por control de acceso

Sentirse seguro es una prioridad para muchos y que mejor lugar para gozar de ese sentimiento que estando en el hogar, por ello, se ha buscado implementar mejoras en los sistemas que controlan el acceso a un lugar, sea una casa, escuela, restaurante, oficina, equipos de cómputo, etc.

Partiendo de este punto, se pueden catalogar a los sistemas de control de acceso en dos grupos:

- **Sistemas no identificables**

  Este tipo de sistemas no se preocupan por el reconocimiento de la persona que está tratando de ingresar al sistema, es decir, se asume que la persona que está ingresando cuenta con la autorización correspondiente. Algunas formas de asumir esto es cuando se tiene una llave, las cuentas y claves de acceso, algún tipo de tarjeta con tecnología RFID, sensores de proximidad o infrarrojos.

- **Sistemas identificables**

  Estos sistemas requieren de un paso extra para permitir el acceso, ya que deben verificar que la persona que va a ingresar es el propietario del mismo sistema o que se encuentra autorizada para ingresar, para ello se requiere ayuda de algún dispositivo biométrico basado en el reconocimiento de huellas dactilares, reconocimiento facial, mediante iris, forma de la mano, voz, reconocimiento de escritura, mapa de las venas en manos o retina, etc.

La aplicación desarrollada aquí corresponde a un control de acceso no identificable (Fig. 4.1), en la cual se solicita que el usuario del sistema ingrese con los datos mencionados a continuación para que se conceda o deniegue el acceso a la aplicación que se desarrolló en el capítulo 3, tema “Sistema de control ON/OFF”.

Fig. 4.1 Interfaz del control de acceso.

Los datos que se registraron para permitir el acceso son:

- Usuario: John Fuentes
- Contraseña: FESC.2019

Al ingresar cualquier palabra o palabras diferentes a “John Fuentes” en la sección de usuario, pero escribiendo la contraseña correcta, al dar clic en el botón INGRESAR deberá aparecer una ventana con el mensaje: “USUARIO INCORRECTO”, de este modo el sistema detecta que la contraseña no es el problema, pero sí lo es el usuario que se ingresó (Fig. 4.2).

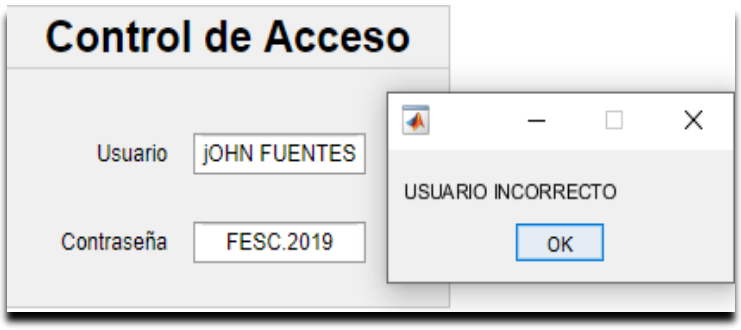


Fig. 4.2 Mensaje: USUARIO INCORRECTO.

Cuando se hace clic en el botón OK y posteriormente se pulsa el botón LIMPIAR DATOS, las casillas de usuario y contraseña quedarán disponibles para ingresar datos nuevamente.

En el caso de ingresar el nombre de usuario correcto y se pone cualquier contraseña diferente a “FESC.2019” el mensaje que debe mostrarse es “CONTRASEÑA INCORRECTA” tal como se muestra en la Fig. 4.3.

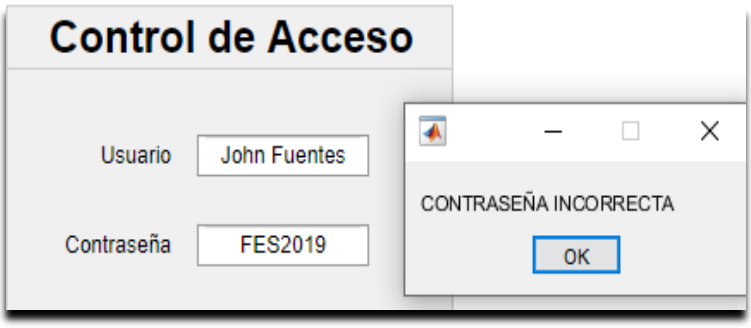


Fig. 4.3 Mensaje: CONTRASEÑA INCORRECTA.

Cuando no se ingresa ningún dato en usuario y contraseña, el sistema arrojará un mensaje de error y este mensaje debe decir: “ERROR DEBES INGRESAR DATOS EN LAS CASILLAS”. A diferencia de los cuadros anteriores, este cuadro cuenta con la leyenda: *Error Dialog* a un costado del ícono de MATLAB, esto se debe a que se cambió en la programación el cuadro de mensajes (*msgbox*) por el cuadro de errores que brinda el software (*errordlg*), lo que se planeo es que el signo de admiración en color rojo y blanco tenga un mayor impacto visual (Fig. 4.4).

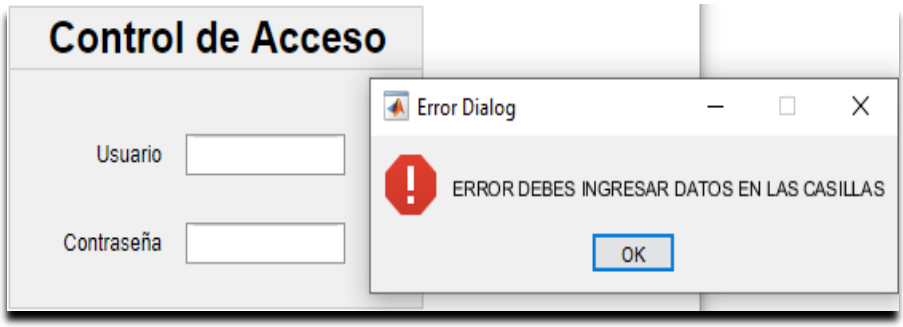


Fig.4.4 Mensaje al dejar los campos vacíos y presionar INGRESAR.

Una de las últimas pruebas realizadas con datos erróneos en esta aplicación es poner los datos de usuario y contraseña distintos a los que nos dan acceso, el mensaje que debe aparecer es: “USUARIO Y CONTRASEÑA INCORRECTOS” (Fig. 4.5).

Fig.4.5 Mensaje al no ingresar el usuario y contraseña de forma correcta.

Por último, como se muestra en la Fig. 4.6, se ingresan los datos correctos en las casillas correspondientes, al dar clic en INGRESAR la aplicación permitirá el acceso al sistema y mostrará el mensaje: “USUARIO Y CONTRASEÑA CORRECTOS” (Fig. 4.7).

Fig.4.6 Ingresando los datos correctos en la aplicación.

Después de mostrarse el mensaje, el sistema cerrará la aplicación de control de acceso y abrirá el área de trabajo de la aplicación del sistema de control ON/OFF como se muestra en la Fig. 4.7.

Fig. 4.7 Mensaje mostrado cuando el usuario y la contraseña son correctos.

La aplicación es capaz de diferenciar entre mayúsculas y minúsculas, espaciados y signos de puntuación, ayudando que al generar las contraseñas se pueda brindar mayor seguridad al intercalar estos caracteres.

La implementación de una aplicación como esta debe ser la primera interacción que se tiene con el sistema, ya que es la protección que se tiene para que no tengan acceso a datos o aplicaciones personas que no estén autorizadas.

En el siguiente capítulo se encontrará que hay otros sistemas de seguridad relacionados a detección de rostros, estos sistemas pueden integrarse o sustituir este tipo de ingresos, pero como todo tendrá sus ventajas y desventajas.

4.2. Sistemas de videovigilancia

El uso de tecnologías en materia de seguridad ciudadana es una práctica cada vez más aceptada y utilizada en México. Por parte de los gobiernos, es notable el despliegue tecnológico para realizar acciones de prevención del delito con el uso de drones, arcos de detección de alarmas, sistemas de videovigilancia, entre otras tantas opciones que están disponibles en el mercado. Se invierten elevadas sumas del presupuesto público en la adquisición de estos productos que se consideran "eficaces" y menos intrusivos, sin embargo, aún no se cuenta con evaluaciones que se permitan conocer con certeza la prevalencia de este tipo de tecnologías.

La videovigilancia es uno de los temas que más resaltan al hablar sobre seguridad en el hogar, pasando de ser un lujo a una necesidad ante la inseguridad que se vive día con día. Aquí es en donde la domótica entra, desde realizar una aplicación e instalación cableada de las cámaras de videovigilancia hasta generar una web e implementar cámaras IP para interiores y/o exteriores, aplicar técnicas de reconocimiento o bien solamente visualizar en el monitor en tiempo real lo que puede capturar la cámara.

La aplicación que se muestra a continuación cuenta con un sistema de control ON/OFF, tal como lo dice el nombre se podrá encender y apagar el monitoreo de las cámaras cuando así se necesite. Las cámaras muestran en tiempo real todo que está sucediendo frente a ellas, pero la aplicación no cuenta con alguna función que permita guardar en algún formato de video lo que se ha estado visualizando.

Para el desarrollo e implementación se requirió:

- 1 computadora con cámara web integrada.
- MATLAB R2020b instalado.
- 1 cámara web alámbrica con conexión USB.
- Instalar el Toolbox de MATLAB Support Package for USB Webcams (Fig. 4.8).

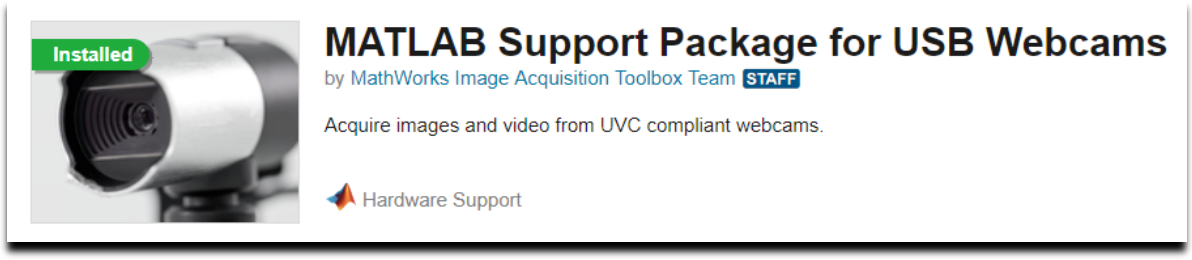


Fig. 4.8 Paquete necesario para la detección de cámaras web externas.

Para desarrollar la interfaz de usuario mostrada en la Fig. 4.9 se utilizó:

- 2 *Axes* → Se podrá observar las imágenes capturadas por las cámaras web.
- 1 *Switch* → Interruptor de tipo ON/OFF para poder iniciar o detener la transmisión.
- 3 *Labels* → Se ocupan para el título de la aplicación y las 2 etiquetas (LAPTOP y CUNA) debajo de los *Axes*.
- 2 *Button* → Botones utilizados para cerrar la aplicación y para realizar el reconocimiento de rostros a partir de una imagen guardada.

Fig. 4.9 Interfaz cámaras de videovigilancia.

Como se muestra en la Fig. 4.10, una vez colocado el interruptor en la posición On comienza a transmitirse los videos en tiempo real de acuerdo con lo capturado en cada cámara.

Fig. 4.10 Cámaras de videovigilancia en funcionamiento.

Para el desarrollo de esta aplicación se tuvo que instalar previamente el software correspondiente a la cámara web conectada por el puerto USB, ya que para ser detectada por MATLAB se requiere de los controladores para poder operar y realizar la comunicación de forma adecuada.

4.3. Detección de rostros para sistemas de seguridad

Como se mostró en la aplicación anterior se puede ejecutar en tiempo real la visualización de videos en diferentes ventanas, partiendo de esto se pueden buscar formas de hacer que se detecten rostros; para hacer la implementación se tuvo que considerar el estudio de la detección de rostros en cascada y el algoritmo propuesto en 2001 por Paul Viola y Michael Jones. [18]

4.3.1. Detector de Objetos en Cascada

Antes de comenzar con la implementación de detección de rostros se debe de conocer la parte teórica del proceso que se lleva a cabo. El detector de objetos en cascada usa el algoritmo *Viola – Jones* para detectar rostros, narices, ojos, boca o la parte superior del cuerpo de las personas, para lograr que se detecten estos rasgos en una imagen debe crearse el objeto *vision.CascadeObjectDetector* y establecer sus propiedades:

```
variable = vision.CascadeObjectDetector('indicar el modelo de clasificación');
```

Posteriormente se llama al objeto con argumentos, como si fuera una función.
Con esta función se crea un detector para utilizar el algoritmo de detección de objetos de Viola – Jones conforme al modelo de clasificación utilizado.

4.3.2. Modelos de clasificación

El algoritmo de *Viola – Jones* cuenta con 12 modelos de clasificación (Tabla 3) para poder hacer la detección de alguna parte específica de la cara de una persona.

MODELO DE CLASIFICACIÓN	¿QUÉ ES LO QUE DETECTA CADA MODELO?
'FrontalFaceCART'	Rostros en posición vertical y que miran de frente.
'FrontalFaceLBP'	Rostros en posición vertical y que miran de frente. Las características de LBP[13] pueden proporcionar robustez frente a variaciones en la iluminación.
'UpperBody'	La región correspondiente a la parte superior del cuerpo, que se define como el área de la cabeza y los hombros.
'EyePairBig'	Un par de ojos.

[13] Las características LBP: Patrón binario local, función de patrón binario local, es un operador utilizado para describir las características de textura locales de una imagen.

El operador de la función LBP es fácil de calcular, tiene un buen efecto y tiene una pequeña cantidad de datos.

'EyePairSmall'	Un par de ojos más pequeños en comparación de los que puede detectar el modelo '*EyePairBig*'.
'LeftEye'	El ojo izquierdo. Utiliza las funciones de *Haar* para codificar detalles.
'RightEye'	El ojo derecho. Utiliza las funciones de *Haar* para codificar detalles.
'LeftEyeCART'	El ojo izquierdo. Los clasificadores débiles que componen estos modelos son los árboles CART.
'RightEyeCART'	El ojo derecho. Los clasificadores débiles que componen estos modelos son los árboles CART
'ProfileFace'	Detecta los perfiles faciales en posición vertical.
'Mouth'	Detecta la boca.
'Nose'	Detecta la nariz.

Tabla 3 Modelos de clasificación para vision.CascadeObjectDetector.

De estos modelos se eligió utilizar *'FrontalFaceCART'*, que se compone por clasificadores débiles, basados en el análisis de árbol de clasificación y regresión (CART). Estos clasificadores usan características de *Haar* para codificar rasgos faciales. Los clasificadores basados en CART brindan la capacidad de modelar dependencias de orden superior entre rasgos faciales. [18]

El algoritmo CART es el acrónimo de *Classification And Regression Trees* (Árboles de Clasificación y de Regresión) fue diseñado por Breiman et al. (1984). Con este algoritmo, se generan árboles de decisión binarios, lo que quiere decir que cada nodo se divide en exactamente dos ramas. Este modelo admite variables de entrada y de salida nominales, ordinales y continuas, por lo que se pueden resolver tanto problemas de clasificación como de regresión. [19] Las características de *Haar* son funciones rectangulares simples de dos dimensiones en las que se varía el tamaño y la posición de recuadros blancos y negros tal como se muestra en la Fig. 4.11.

Fig. 4.11 Filtros *Haar* escalados en diferentes posiciones de la imagen. [20]

Cada característica es un valor único que se obtiene al restar la suma de píxeles en el rectángulo blanco de la suma de píxeles en el rectángulo negro.

Hay que tomar en consideración que para esta aplicación no se desarrolló ninguna técnica, tampoco se trata de desglosar los cálculos matemáticos que componen al algoritmo de *Viola – Jones* o con el desarrollo del entrenamiento para la detección de rostros[14].

4.3.3. Implementación del algoritmo Viola – Jones en aplicación desarrollada

Retomando la aplicación desarrollada para videovigilancia se lleva a cabo la implementación de la técnica de detección de rostros mediante una imagen. La aplicación cuenta con un botón que fue nombrado como: RECONOCE, este botón permitirá detectar rostros a partir de una imagen que se haya guardado previamente con el nombre: caraitse.

La detección de rostros no se puede generar en tiempo real en la aplicación desarrollada, pero sí se permite capturar una imagen del video que está ejecutándose en la aplicación con ayuda

[14] Para conocer más información sobre el entrenamiento o el desarrollo de la técnica se puede consultar: https://la.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html?searchHighlight=viola%20jones&s_tid=srchtitle

de los menús predeterminados que proporciona App Designer y que se encuentran descritos en la Tabla 4.

El acceso a estos botones (Fig. 4.12) aparece cuando se pasa el cursor sobre los cuadros que están capturando el video.

Fig. 4.12 Opciones que brinda el sistema al incluir un *Axes*.

BOTÓN		ACCIÓN
		Cuenta con 3 submenús: • *Save As* (Guardar como) • *Copy As Image* (Copiar como imagen) • *Copy As Vector Graphic* (Copiar como un vector gráfico)
	***PAN***	Mueve la pantalla en la que se proyecta el video.
	***ZOOM IN***	Permite hacer un acercamiento al cuadro de lo que se está capturando a través de la cámara web.
	***ZOOM OUT***	Permite alejar el cuadro de lo que se está capturando a través de la cámara web.
	***RESTORE***	Regresa a los valores predefinidos.

Tabla 4 Acciones de los botones que brinda un *Axes*.

Con el cursor sobre el recuadro nombrado LAPTOP se selecciona la opción: *Save As*, se procede a guardar la imagen nombrándola como: caraitse.png. El nombre puede cambiarse ingresando en el código de la aplicación, solo debe considerarse la estructura que es compuesta por el nombre que se le quiere dar a la imagen (caraitse) y la extensión del tipo de archivo (.png).

En una de las pruebas realizadas se capturó una imagen, se guardó y se presionó el botón RECONOCE se obtuvo como resultado la detección de una sola cara, aun cuando en el recuadro del video se mostraban dos personas y el resultado obtenido fue tal como se muestra en la Fig. 4.13.

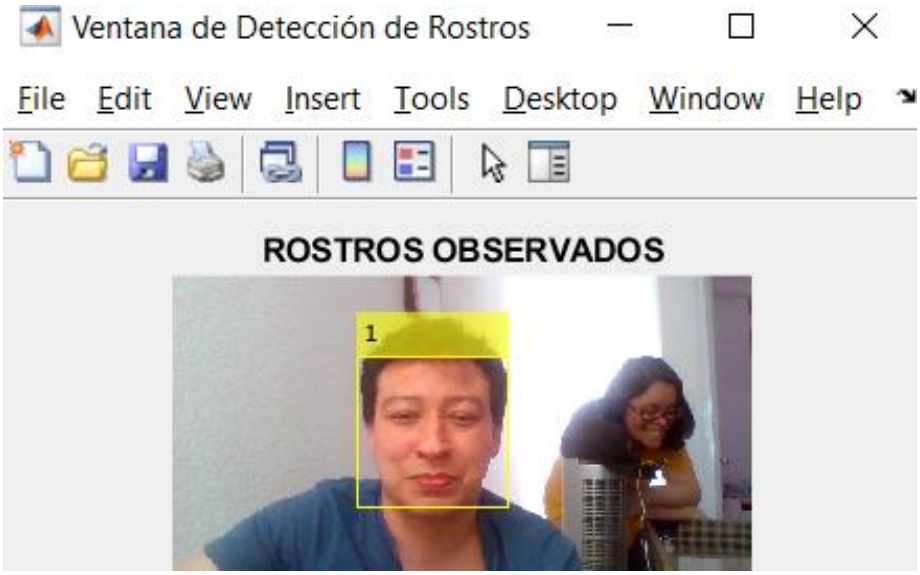


Fig. 4.13 Rostro detectado al ejecutar la aplicación.

Para realizar la detección de rostros en esta aplicación se utilizó el modelo *'FrontalFaceCART'*, si se consulta nuevamente la Tabla 3 se puede verificar que este modelo de clasificación requiere que el o los rostros capturados estén en posición vertical y mirando de frente, siendo esta una de las posibles causas por las cuales pudo no haberse detectado el segundo rostro, sin embargo, otros factores que pueden influir de manera negativa en la detección del rostro son:

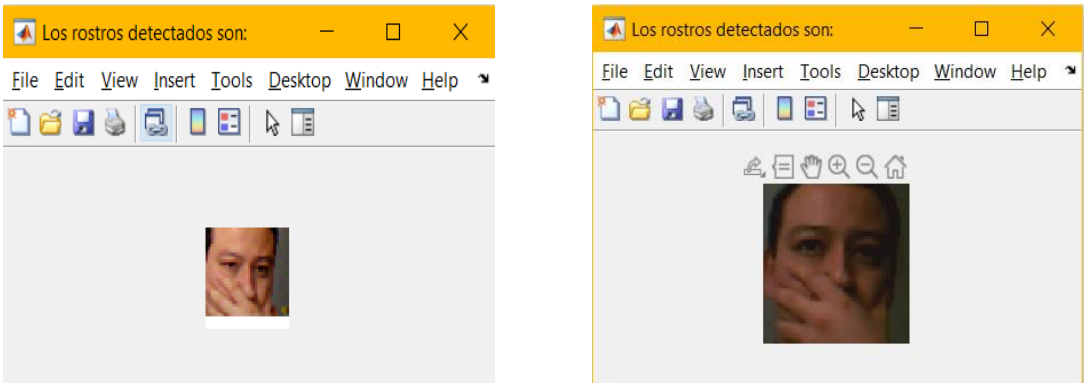
- Baja calidad de la imagen a examinar.
- Luz que hay de fondo.
- Elementos que obstaculizan una buena comparación.
- El equipo con el que se cuenta no logra realizar los cálculos adecuados en un tiempo determinado.
- Falta de similitudes con los datos que tiene previamente entrenados de acuerdo con el modelo de clasificación utilizado.

Con base en los resultados obtenidos en la prueba anterior se ejecutó una prueba más, en cual se intentaría no cumplir con algunas condiciones generales del algoritmo implementado. Para llevar a cabo esta prueba se decidió cubrir la boca, reducir la iluminación, alejar y posicionar la cámara correspondiente a LAPTOP tratando de hacer que la cara no parezca estar en posición vertical, mientras que en la cámara nombrada CUNA se realizó un acercamiento tratando de que la cara cumpliera con el requisito de ver de frente y estar en posición vertical; los resultados obtenidos fueron los que se muestran en la Fig. 4.14.

Fig. 4.14 Rostros detectados con poca iluminación haciendo captura de pantalla desde la aplicación.

Como puede observarse, la aplicación fue capaz de detectar los dos rostros (aunque el rostro sea el mismo para ambas cámaras) incluso quitando algunos de los elementos que nos dicen que son considerados en el entrenamiento del algoritmo para realizar una detección correcta, lo que hace que a pesar de no haber detectado un rostro en la Fig. 4.13 el algoritmo se ejecuta de una forma adecuada tomando en cuenta que todo programa puede contar con ligeras excepciones y/o rangos de error.

Una particularidad que se puede destacar de este tipo de reconocimientos es que además de poner en un recuadro amarillo y enumerar los rostros detectados, se puede programar una sección que permita extraer las caras detectadas en ventanas individuales tal como se muestra en la Fig. 4.15.



Rostro 1 Rostro 2

Fig. 4.15 Recorte y separación de los rostros 1 y 2 detectados en la Fig. 4.14.

La importancia de contar con este tipo de implementaciones en un sistema de videovigilancia es que en términos de seguridad esto puede ayudar en gran medida cuando se quiere quitar el fondo que puede impedir una buena visualización del rostro detectado, además que dentro de las herramientas que integra cada ventana se puede hacer un acercamiento mayor a la imagen capturada, guardarla y procesarla para recuperar mayor calidad de ella.

CONCLUSIONES

Conclusiones del trabajo de tesis

En virtud de lo descrito en este trabajo de tesis, el desarrollo e implementación de las aplicaciones y la comunicación entre el software y hardware mencionados en los capítulos 2, 3 y 4 se lograron efectuar sin ningún inconveniente, sin embargo, en el proceso de planeación se había considerado profundizar un poco más, realizando el desarrollo de los sistemas con otros componentes electrónicos, pero la situación actual que se está viviendo con la pandemia ha complicado la adquisición de algunos dispositivos generando que se trabajara con los componentes existentes o con aquellos que sí se podían conseguir aunque estos no fueran los que se habían estimado en el proceso de planeación.

Considerando el tema "tipo de soportes utilizados como vía de comunicación", se planeaba implementar la transmisión y recepción de datos mediante el módulo de Bluetooth HC-05[15] y con esto poder efectuar la comunicación de forma inalámbrica. El obstáculo que se presentó con esta idea fue que, aunque se contaba con el módulo HC-05 para poder hacer la comunicación con MATLAB, se requiere de un dispositivo convertidor serial a TTL basado en circuito FTDI, este es fundamental en la configuración en MATLAB ya que permite generar la programación y comunicación adecuada con el hardware, sin embargo, la adquisición del dispositivo se complicó ya que algunas tiendas en línea no manejaban el modelo FTDI232 y tenían otros modelos, otras tiendas que sí contaban con él daban tiempos de entrega mayores a tres meses y había muchos comentarios negativos sobre los vendedores y los productos que ofrecían, lo que generó desconfianza solicitar este dispositivo, por tanto, se decidió dejarlo como una mejora futura para el entendimiento de este tipo de comunicaciones inalámbricas.

Algo muy importante para tener en cuenta cuando se habla de domótica es saber qué tipo de tecnología de control se decidirá implementar. En el caso de las aplicaciones que se desarrollaron no se tiene una tecnología descrita como tal, ya que pueden ser catalogadas como centralizadas, descentralizadas o mixtas de gestión, de acuerdo a la forma en que se

[15] El módulo HC-05 cumple con las especificaciones del estándar Bluetooth 2.0 que es perfectamente compatible con celulares o smartphones Android y MATLAB con apoyo del convertidor serial.

decida ver la implementación; por ejemplo, para decir que son sistemas centralizados se puede resaltar que un sistema fue empleado con un solo microcontrolador (tarjeta de desarrollo Arduino UNO), lo cual generaba que si había algún tipo de problema de comunicación o falla electrónica el sistema quedaría sin funcionamiento; por otro lado, al emplear los sistemas por separado y ninguno de ellos interactuar entre sí se diría que es un sistema descentralizado, y por último, si se realizara algún tipo de mejora a las aplicaciones, estas se conjuntaran en una sola y combinando la idea del sistema descentralizado se puede decir que este sistema es mixto de gestión ya que podría visualizarse y cambiarse los estados de entrada y salida en los microcontroladores desde un solo mando.

El principal aspecto para destacar independientemente de la tecnología de control a enfocar es el uso e implementación de ellas en la domótica ya que las aplicaciones en conjunto son capaces de brindar seguridad, confort, comunicación, ahorro energético y accesibilidad, siendo esto los aspectos característicos en los hogares domóticos.

Es necesario tener presente que, además de MATLAB y la tarjeta de desarrollo Arduino UNO hay muchas otras opciones que pueden llegar a considerarse para el desarrollo e implementación de las aplicaciones domóticas, todo dependerá únicamente del diseño y planeación que se realice al inicio, en el proceso y al final de los proyectos a desarrollar, ya que la domótica abarca muchos temas de estudio, que pueden ir de cosas muy sencillas a muy elaboradas, desde cambiar un foco convencional por uno que encienda una vez que no se cuenta con la iluminación adecuada, hasta el estudio completo del hogar tomando en consideración la modificación o generación de nuevas instalaciones eléctricas, que brinden un buen funcionamiento y protección a los sistemas de iluminación, riego, aire acondicionado, monitoreo de válvulas de gas, etc.

En la investigación realizada se encontró que en España se cuenta con una asociación llamada CEDOM, siendo una de las pocas asociaciones preocupadas en regularizar y estandarizar las definiciones e instalaciones de la domótica. En México, existen empresas dedicadas a la automatización de casas inteligentes, sin embargo, estas solo se encargan de implementar dispositivos que aparentemente hacen de la casa, un hogar inteligente, pero no toman en

consideración aspectos importantes como son: el tipo de cableado a utilizar, el ancho de banda que se tiene contratado para los dispositivos que vayan a estar comunicados por este medio o las tecnologías de control mencionadas en el marco teórico de este trabajo.

Ninguna de las aplicaciones que fueron desarrolladas en este trabajo tiene mayor importancia una sobre la otra, todas cumplen un papel significativo en la integración de los sistemas domóticos; como ya se ha mencionado, el mundo de las aplicaciones domóticas es basto y digno de su estudio a profundidad, por tal motivo, lo redactado en esta tesis da pie a generar más desarrollos, seguir investigando, cuestionar y refutar lo que se ha planteado hasta el momento en este trabajo, esperando una mejora continua para aquellos que lean este trabajo y para quienes se encuentran interesados en esta área de conocimiento de la ingeniería.

Finalmente, cabe recordar que uno de los objetivos particulares de este trabajo es que el lector despierte el interés por seguir desarrollando sistemas domóticos de manera correcta. Si se logrará llevar a cabo este objetivo se tiene la idea que pueda haber personas más capacitadas e interesadas en llevar a cabo instalaciones domóticas partiendo de una buena fundamentación teórica y llevando a cabo una buena implementación práctica.

REFERENCIAS

Referencias

[1] D. Spicer, «Computer History Museum,» 31 Mayo 2016. [En línea]. Available: https://computerhistory.org/blog/the-echo-iv-home-computer-50-years-later/. [Último acceso: 30 Noviembre 2020].

[2] J. C. CUEVAS y J. y. M. P. MARTÍNEZ, «El Protocolo x10: Una solución Antigua a Problemas actuales,» *Simposio de Informática y Telecomunicaciones SIT*, vol. 2, p. 89, 2002.

[3] C. d. Wikipedia, «Wikipedia,» 8 Mayo 2021. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Motorola_DynaTAC&oldid=135390449. [Último acceso: 12 Mayo 2021].

[4] CEDOM, «Asociación Española de Domótica e Inmótica - CEDOM,» [En línea]. Available: http://www.cedom.es/sobre-domotica/que-es-domotica. [Último acceso: 21 agosto 2020].

[5] CEDOM, «Asociación Española de Domótica e Inmótica - CEDOM,» [En línea]. Available: http://www.cedom.es/sobre-domotica/que-es-inmotica. [Último acceso: 21 Agosto 2020].

[6] F. d. E. S. Cuautitlán, «Universidad Nacional Autónoma de México. Facultad de Estudios Superiores Cuautitlán.,» 2019. [En línea]. Available: https://www.cuautitlan.unam.mx/licenciaturas/itse/plan_estudios.html. [Último acceso: 9 Agosto 2020].

[7] F. Guzmán Navarro y S. Merino Córdoba, de *Domótica. Gestión de la energía y gestión técnica de edificios*, Madrid, RA-MA, 2015, pp. 21-46.

[8] The MathWorks, Inc., «MathWorks,» [En línea]. Available: https://la.mathworks.com/products/matlab.html. [Último acceso: 21 Octubre 2020].

[9] The MathWorks, Inc., «MathWorks,» [En línea]. Available: https://la.mathworks.com/support/requirements/matlab-system-requirements.html. [Último acceso: 30 Noviembre 2020].

[10] The MathWorks, Inc., «MathWorks,» [En línea]. Available: https://la.mathworks.com/products/matlab/app-designer.html. [Último acceso: 2 Diciembre 2020].

[11] RaspberryPi, «RaspberryPi,» [En línea]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/. [Último acceso: 1 Febrero 2021].

[12] DIGILENT, «DIGILENT. A National Instruments Company,» [En línea]. Available: https://store.digilentinc.com/basys-3-artix-7-fpga-beginner-board-recommended-for-introductory-users/. [Último acceso: 1 Febrero 2021].

[13] Texas Instruments, «TEXAS INSTRUMENTS,» [En línea]. Available: https://www.ti.com/tool/EK-TM4C123GXL. [Último acceso: 1 Febrero 2021].

[14] Arduino, «Arduino.cc,» [En línea]. Available: https://www.arduino.cc/en/Guide/Introduction. [Último acceso: 6 Enero 2021].

[15] Arduino, «Arduino.cc,» [En línea]. Available: https://support.arduino.cc/hc/en-us/articles/360020652100. [Último acceso: 6 Diciembre 2020].

[16] Arduino, «Arduino.cc,» [En línea]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Último acceso: 1 Febrero 2021].

[17] M. M. H. Team, «MathWorks,» 2019. [En línea]. Available: https://la.mathworks.com/matlabcentral/fileexchange/47522-matlab-support-package-for-arduino-hardware. [Último acceso: 4 enero 2021].

[18] P. Viola y M. Jones, «Rapid object detection using a boosted cascade of simple features,» de *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition*, 2001.

[19] The MathWorks, Inc., «MathWorks,» [En línea]. Available: https://la.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html?searchHighlight=viola%20jones&s_tid=srchtitle#bs_kj0p. [Último acceso: 15 abril 2021].

[20] F. P. Rodríguez, «BOOKDOWN,» 25 Enero 2019. [En línea]. Available: https://bookdown.org/content/2274/portada.html. [Último acceso: 15 Abril 2021].

[21] C. J. P. HERRERA, «VISIÓN POR COMPUTADOR > Ing. Carlos Julio Pardo,» 2017. [En línea]. Available: https://carlosjuliopardoblog.wordpress.com/2017/05/12/filtros-haar-deteccion-de-rostros/. [Último acceso: 12 Abril 2021].

[22] F. Guzmán Navarro y S. Merino Córdoba, de *Domótica. Gestión de la energía y gestión técnica de edificios*, 2015.

ANEXOS

ANEXO 1. Control ON/OFF con interruptor digital

```
classdef OnOffInterruptor_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        OnOff                   matlab.ui.Figure
        CerrarApp               matlab.ui.control.Button
        Interruptor             matlab.ui.control.Switch
        INTERRUPTORSwitchLabel  matlab.ui.control.Label
        ReiniciaPrograma        matlab.ui.control.Button
        FESC                    matlab.ui.control.Image
        ITSE                    matlab.ui.control.Image
        ControlOnOffLabel       matlab.ui.control.Label
        UNAM                    matlab.ui.control.Label
        LED                     matlab.ui.control.Lamp
        LEDLabel                matlab.ui.control.Label
    end

    %Creamos una propiedad de acceso privado, en donde declaramos la
    %variable "a"
    properties (Access = private)
        a
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            app.a = arduino();
        end

        % Value changed function: Interruptor
        function InterruptorValueChanged(app, event)

            estadoActual = app.Interruptor.Value;

            % strcmp compara la cadena de caracteres guardadas en
            % estadoActual en este caso compara "On con Off"
            if strcmp(estadoActual,'On')

                % Si el Switch se encuentra en la posición "On" el Foco
                % de la app será Verde (g es la variable que MATLAB brinda)
                app.LED.Color = 'g';

                % El pin del RGB conectado a D4 del Arduino
                % permanecerá Apagado
                writeDigitalPin(app.a,'D11',0)

                % El pin del RGB conectado a D3 del Arduino
                % permanecerá en Verde
                writeDigitalPin(app.a,'D3',1)
```

```
            else
                % Si el Switch se encuentra en la posición Off el Foco
                % de la app será Rojo (r)
                app.LED.Color = 'r';

                % El RGB conectado a D4 del Arduino permanecerá Rojo
                writeDigitalPin(app.a,'D11',1)

                % El RGB conectado a D3 del Arduino permanecerá Apagado
                writeDigitalPin(app.a,'D3',0)
            end

        end

        % Button pushed function: ReiniciaPrograma
        function ReiniciaProgramaPushed(app, event)

            % Al presionar el botón de "Reinicio" se apagará el LED de la
            % App y el RGB conectado a través de los puertos de la placa
            % Arduino
            app.LED.Color = '0.90,0.90,0.90';

            % Los puertos D4 y D3 permanecerán en estado bajo
            writeDigitalPin(app.a,'D11',0)
            writeDigitalPin(app.a,'D3',0)

            % El interruptor regresa a la posición Off
            app.Interruptor.Value = 'Off';

        end

        % Button pushed function: CerrarApp
        function CerrarAppButtonPushed(app, event)

        % Cuando presionamos el botón de Cerrar App cerrará la ventana de
        % nuestra aplicación
        close(app.OnOff)
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create OnOff and hide until all components are created
            app.OnOff = uifigure('Visible', 'off');
            app.OnOff.Color = [1 1 1];
            app.OnOff.Position = [100 100 606 416];
            app.OnOff.Name = 'CONTROL ON/OFF';
            % Create LEDLabel
            app.LEDLabel = uilabel(app.OnOff);
            app.LEDLabel.HorizontalAlignment = 'center';
```

```
            app.LEDLabel.Position = [289 122 29 22];
            app.LEDLabel.Text = 'LED';

            % Create LED
            app.LED = uilamp(app.OnOff);
            app.LED.Position = [277 143 53 53];
            app.LED.Color = [0.902 0.902 0.902];

            % Create UNAM
            app.UNAM = uilabel(app.OnOff);
            app.UNAM.HorizontalAlignment = 'center';
            app.UNAM.FontSize = 14;
            app.UNAM.FontWeight = 'bold';
            app.UNAM.Position = [117 311 373 80];
            app.UNAM.Text = {'UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO'; 'FACULTAD DE
ESTUDIOS SUPERIORES CUAUTITLÁN'; ''; 'INGENIERÍA EN TELECOMUNICACIONES, SISTEMAS '; 'Y
ELECTRÓNICA'};

            % Create ControlOnOffLabel
            app.ControlOnOffLabel = uilabel(app.OnOff);
            app.ControlOnOffLabel.HorizontalAlignment = 'center';
            app.ControlOnOffLabel.FontSize = 24;
            app.ControlOnOffLabel.FontWeight = 'bold';
            app.ControlOnOffLabel.Position = [202.5 224 202 30];
            app.ControlOnOffLabel.Text = 'Control ON / OFF';

            % Create ITSE
            app.ITSE = uiimage(app.OnOff);
            app.ITSE.Position = [500 301 107 116];
            app.ITSE.ImageSource = 'ITSE.png';

            % Create FESC
            app.FESC = uiimage(app.OnOff);
            app.FESC.Position = [1 301 100 100];
            app.FESC.ImageSource = 'FESC.png';

            % Create ReiniciaPrograma
            app.ReiniciaPrograma = uibutton(app.OnOff, 'push');
            app.ReiniciaPrograma.ButtonPushedFcn = createCallbackFcn(app,
@ReiniciaProgramaPushed, true);
            app.ReiniciaPrograma.IconAlignment = 'center';
            app.ReiniciaPrograma.FontWeight = 'bold';
            app.ReiniciaPrograma.Position = [467 27 67 38];
            app.ReiniciaPrograma.Text = 'REINICIAR';

            % Create INTERRUPTORSwitchLabel
            app.INTERRUPTORSwitchLabel = uilabel(app.OnOff);
            app.INTERRUPTORSwitchLabel.HorizontalAlignment = 'center';
            app.INTERRUPTORSwitchLabel.Position = [260 27 92 22];
            app.INTERRUPTORSwitchLabel.Text = 'INTERRUPTOR';

            % Create Interruptor
            app.Interruptor = uiswitch(app.OnOff, 'slider');
            app.Interruptor.ValueChangedFcn = createCallbackFcn(app,
```

```
@InterruptorValueChanged, true);
                app.Interruptor.FontSize = 18;
                app.Interruptor.FontWeight = 'bold';
                app.Interruptor.Position = [282 64 45 20];

                % Create CerrarApp
                app.CerrarApp = uibutton(app.OnOff, 'push');
                app.CerrarApp.ButtonPushedFcn = createCallbackFcn(app,
@CerrarAppButtonPushed, true);
                app.CerrarApp.FontWeight = 'bold';
                app.CerrarApp.Position = [70 29 72 36];
                app.CerrarApp.Text = {'CERRAR'; 'APP'};

                % Show the figure after all components are created
                app.OnOff.Visible = 'on';
            end
        end

        % App creation and deletion
        methods (Access = public)

            % Construct app
            function app = OnOffInterruptor_exported

                % Create UIFigure and components
                createComponents(app)

                % Register the app with App Designer
                registerApp(app, app.OnOff)

                % Execute the startup function
                runStartupFcn(app, @startupFcn)

                if nargout == 0
                    clear app
                end
            end

            % Code that executes before app deletion
            function delete(app)

                % Delete UIFigure when app is deleted
                delete(app.OnOff)
            end
        end
    end
```

ANEXO 2. Control ON/OFF implementando un LDR

```
classdef LDR_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        EjercicioOcho   matlab.ui.Figure
        Image3          matlab.ui.control.Image
        Image2          matlab.ui.control.Image
        Image           matlab.ui.control.Image
        ATARDECERLabel  matlab.ui.control.Label
        DALabel         matlab.ui.control.Label
        NOCHELabel      matlab.ui.control.Label
        FESCimagen      matlab.ui.control.Image
        ITSEimagen      matlab.ui.control.Image
        UNAM            matlab.ui.control.Label
        Inicio          matlab.ui.control.StateButton
        Label           matlab.ui.control.Label
        CerrarApp       matlab.ui.control.Button
        Termometro      matlab.ui.control.LinearGauge
        TermmetroLabel  matlab.ui.control.Label
    end

    properties (Access = private)
        a
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            app.a = arduino();
        end

        % Button pushed function: CerrarApp
        function CerrarAppPushed(app, event)
            % Cuando presionamos el botón de Cerrar App cerrará la ventana
            % de nuestra aplicación
            close(app.EjercicioOcho)

            % Se utiliza este valor para limpiar la variable "v" y evitar
            % que la app muestre el mensaje de "error"
            clear InicioValueChanged;
            clear a;
        end

        % Value changed function: Inicio
        function InicioValueChanged(app, event)

        botonEstado = app.Inicio.Value;

        while(botonEstado==true)
```

```
            %El LDR permanecerá conectado a través del puerto Analógico A0
            %del Arduino UNO
            lecturaLDR = readVoltage(app.a,'A0');

            % Muestra el valor de la lectura obtenida a través del LDR
            app.Label.Text = char((string(lecturaLDR)));

            % Muestra de manera gráfica el valor en el Termómetro
            app.Termometro.Value = lecturaLDR;

            % Si
            if (lecturaLDR<=2)
                writeDigitalPin(app.a,'D4',0)
            else
                writeDigitalPin(app.a,'D4',1)
            end
        end

        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create EjercicioOcho and hide until all components are created
            app.EjercicioOcho = uifigure('Visible', 'off');
            app.EjercicioOcho.Color = [1 1 1];
            app.EjercicioOcho.Position = [100 100 606 416];
            app.EjercicioOcho.Name = 'UI Figure';

            % Create TermmetroLabel
            app.TermmetroLabel = uilabel(app.EjercicioOcho);
            app.TermmetroLabel.HorizontalAlignment = 'center';
            app.TermmetroLabel.FontWeight = 'bold';
            app.TermmetroLabel.Position = [368 184 75 22];
            app.TermmetroLabel.Text = 'Termómetro';

            % Create Termometro
            app.Termometro = uigauge(app.EjercicioOcho, 'linear');
            app.Termometro.Limits = [0 5];
            app.Termometro.ScaleColors = [0.0118 0.1765 0.4118;0.0471 0.1098 0.8;0.0353
0.5098 0.702;0.851 0.851 0.3569;1 0.8667 0;1 0.8667 0];
            app.Termometro.ScaleColorLimits = [0 0.5;0.5 1.5;1.5 2;2 3;3 4;4 5];
            app.Termometro.Position = [162 175 283 40];

            % Create CerrarApp
            app.CerrarApp = uibutton(app.EjercicioOcho, 'push');
            app.CerrarApp.ButtonPushedFcn = createCallbackFcn(app, @CerrarAppPushed, true);
            app.CerrarApp.FontWeight = 'bold';
            app.CerrarApp.Position = [70 29 72 36];
            app.CerrarApp.Text = {'CERRAR'; 'APP'};
```

```
            % Create Label
            app.Label = uilabel(app.EjercicioOcho);
            app.Label.BackgroundColor = [1 1 1];
            app.Label.HorizontalAlignment = 'center';
            app.Label.FontWeight = 'bold';
            app.Label.Position = [257 137 94 31];
            app.Label.Text = '';

            % Create Inicio
            app.Inicio = uibutton(app.EjercicioOcho, 'state');
            app.Inicio.ValueChangedFcn = createCallbackFcn(app, @InicioValueChanged, true);
            app.Inicio.Text = {'INICIAR'; 'LECTURA'};
            app.Inicio.FontWeight = 'bold';
            app.Inicio.Position = [467 27 78 38];

            % Create UNAM
            app.UNAM = uilabel(app.EjercicioOcho);
            app.UNAM.HorizontalAlignment = 'center';
            app.UNAM.FontSize = 14;
            app.UNAM.FontWeight = 'bold';
            app.UNAM.Position = [117 311 373 80];
            app.UNAM.Text = {'UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO'; 'FACULTAD DE
ESTUDIOS SUPERIORES CUAUTITLÁN'; ''; 'INGENIERÍA EN TELECOMUNICACIONES, SISTEMAS '; 'Y
ELECTRÓNICA'};

            % Create ITSEimagen
            app.ITSEimagen = uiimage(app.EjercicioOcho);
            app.ITSEimagen.Position = [499 301 108 116];
            app.ITSEimagen.ImageSource = 'ITSE.png';

            % Create FESCimagen
            app.FESCimagen = uiimage(app.EjercicioOcho);
            app.FESCimagen.Position = [1 301 100 100];
            app.FESCimagen.ImageSource = 'FESC.png';

            % Create NOCHELabel
            app.NOCHELabel = uilabel(app.EjercicioOcho);
            app.NOCHELabel.FontWeight = 'bold';
            app.NOCHELabel.Position = [114 214 49 22];
            app.NOCHELabel.Text = 'NOCHE';

            % Create DALabel
            app.DALabel = uilabel(app.EjercicioOcho);
            app.DALabel.HorizontalAlignment = 'right';
            app.DALabel.FontWeight = 'bold';
            app.DALabel.Position = [442 214 26 22];
            app.DALabel.Text = 'DÍA';

            % Create ATARDECERLabel
            app.ATARDECERLabel = uilabel(app.EjercicioOcho);
            app.ATARDECERLabel.FontWeight = 'bold';
            app.ATARDECERLabel.Position = [265 214 79 22];
            app.ATARDECERLabel.Text = 'ATARDECER';
```

```
            % Create Image
            app.Image = uiimage(app.EjercicioOcho);
            app.Image.Position = [70 132 196 41];
            app.Image.ImageSource = 'Noche.jpg';

            % Create Image2
            app.Image2 = uiimage(app.EjercicioOcho);
            app.Image2.Position = [178 132 234 41];
            app.Image2.ImageSource = 'Tarde.jpg';

            % Create Image3
            app.Image3 = uiimage(app.EjercicioOcho);
            app.Image3.Position = [323 132 196 41];
            app.Image3.ImageSource = 'Día.jpg';

            % Show the figure after all components are created
            app.EjercicioOcho.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = LDR_exported

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.EjercicioOcho)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.EjercicioOcho)
        end
    end
end
```

ANEXO 3. Regulación de intensidad luminosa con un dimmer digital

```
classdef ReguladorDeIntensidad_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        EjercicioCuatro             matlab.ui.Figure
        CerrarApp                   matlab.ui.control.Button
        Porcentaje                  matlab.ui.control.Label
        FESCimagen                  matlab.ui.control.Image
        ITSEimagen                  matlab.ui.control.Image
        UNAM                        matlab.ui.control.Label
        Dimmer                      matlab.ui.control.Knob
        ReguladorDeIntensidadLabel  matlab.ui.control.Label
    end

    properties (Access = private)
        %Creamos una variable para guardar el puerto COM de Arduino
        a
    end


    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)

            %Ingresamos la variable 'a' para brindarle las propiedades que
            %necesitamos
            app.a = arduino();
        end

        % Value changing function: Dimmer
        function DimmerValueChanging(app, event)

            valorDimmer = event.Value;

            %Sustituye el título de Control Luminosidad por el valor del
            %Dimmer más el signo de porcentaje
            app.ReguladorDeIntensidadLabel.Text = "Luminosidad al: ";
            app.Porcentaje.Text = char((string(valorDimmer)) + '%');

            % 1. Damos la instrucción "writePWMDutyCycle" para poder ocupar
            % el PWM del Arduino
            % 2. Indicamos que debe buscar el puerto asiganado a "app.a"
            % 3. El pin en donde se encontrará nuestra salida, para ello
            % solo podemos utilizar los pines digitales (PWM) marcados con
            % el símbolo " ~ " en la placa Arduino, siendo: D3, D5, D6, D9,
            % D10 o D11
            % 4. El valor que seleccione en el Dimmer de la App se dividirá
            % en 100 Ya que es el limite que le asignamos al Dimmer y el
            % para que el ciclo de trabajo sea válido debe abarcar valores
```

```
            % en el rango de 0 - 1
            % más en poder observar el brillo máximo del LED) para ir
            % graduando la luminosidad de 0V a 3.3V

            writePWMDutyCycle(app.a, 'D11', valorDimmer/(100));

        end

        % Button pushed function: CerrarApp
        function CerrarAppButtonPushed(app, event)
            close(app.EjercicioCuatro)

            %Limpiamos el puerto
            clear a;
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create EjercicioCuatro and hide until all components are created
            app.EjercicioCuatro = uifigure('Visible', 'off');
            app.EjercicioCuatro.Color = [1 1 1];
            app.EjercicioCuatro.Colormap = [0.2431 0.149 0.6588;0.251 0.1647 0.7059;0.2588
0.1804 0.7529;0.2627 0.1961 0.7961;0.2706 0.2157 0.8353;0.2745 0.2353 0.8706;0.2784 0.2549
0.898;0.2784 0.2784 0.9216;0.2824 0.302 0.9412;0.2824 0.3216 0.9569;0.2784 0.3451
0.9725;0.2745 0.3686 0.9843;0.2706 0.3882 0.9922;0.2588 0.4118 0.9961;0.2431 0.4353
1;0.2196 0.4588 0.9961;0.1961 0.4863 0.9882;0.1843 0.5059 0.9804;0.1804 0.5294
0.9686;0.1765 0.549 0.9529;0.1686 0.5686 0.9373;0.1529 0.5922 0.9216;0.1451 0.6078
0.9098;0.1373 0.6275 0.898;0.1255 0.6471 0.8902;0.1098 0.6627 0.8745;0.0941 0.6784
0.8588;0.0706 0.6941 0.8392;0.0314 0.7098 0.8157;0.0039 0.7216 0.7922;0.0078 0.7294
0.7647;0.0431 0.7412 0.7412;0.098 0.749 0.7137;0.1412 0.7569 0.6824;0.1725 0.7686
0.6549;0.1922 0.7765 0.6235;0.2157 0.7843 0.5922;0.2471 0.7922 0.5569;0.2902 0.7961
0.5176;0.3412 0.8 0.4784;0.3922 0.8039 0.4353;0.4471 0.8039 0.3922;0.5059 0.8 0.349;0.5608
0.7961 0.3059;0.6157 0.7882 0.2627;0.6706 0.7804 0.2235;0.7255 0.7686 0.1922;0.7725 0.7608
0.1647;0.8196 0.749 0.1529;0.8627 0.7412 0.1608;0.902 0.7333 0.1765;0.9412 0.7294
0.2118;0.9725 0.7294 0.2392;0.9961 0.7451 0.2353;0.9961 0.7647 0.2196;0.9961 0.7882
0.2039;0.9882 0.8118 0.1882;0.9804 0.8392 0.1765;0.9686 0.8627 0.1647;0.9608 0.8902
0.1529;0.9608 0.9137 0.1412;0.9647 0.9373 0.1255;0.9686 0.9608 0.1059;0.9765 0.9843
0.0824];
            app.EjercicioCuatro.Position = [100 100 606 416];
            app.EjercicioCuatro.Name = 'Regulador de Intensidad';

            % Create ReguladorDeIntensidadLabel
            app.ReguladorDeIntensidadLabel = uilabel(app.EjercicioCuatro);
            app.ReguladorDeIntensidadLabel.HorizontalAlignment = 'center';
            app.ReguladorDeIntensidadLabel.FontSize = 24;
            app.ReguladorDeIntensidadLabel.FontWeight = 'bold';
            app.ReguladorDeIntensidadLabel.Position = [160 218 289 56];
            app.ReguladorDeIntensidadLabel.Text = 'Regulador de intensidad';
```

```
            % Create Dimmer
            app.Dimmer = uiknob(app.EjercicioCuatro, 'continuous');
            app.Dimmer.Limits = [0 3];
            app.Dimmer.ValueChangingFcn = createCallbackFcn(app, @DimmerValueChanging,
true);
            app.Dimmer.Position = [246 44 116 116];

            % Create UNAM
            app.UNAM = uilabel(app.EjercicioCuatro);
            app.UNAM.HorizontalAlignment = 'center';
            app.UNAM.FontSize = 14;
            app.UNAM.FontWeight = 'bold';
            app.UNAM.Position = [117 311 373 80];
            app.UNAM.Text = {'UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO'; 'FACULTAD DE
ESTUDIOS SUPERIORES CUAUTITLÁN'; ''; 'INGENIERÍA EN TELECOMUNICACIONES, SISTEMAS '; 'Y
ELECTRÓNICA'};

            % Create ITSEimagen
            app.ITSEimagen = uiimage(app.EjercicioCuatro);
            app.ITSEimagen.Position = [499 301 108 116];
            app.ITSEimagen.ImageSource = 'ITSE.png';

            % Create FESCimagen
            app.FESCimagen = uiimage(app.EjercicioCuatro);
            app.FESCimagen.Position = [1 301 100 100];
            app.FESCimagen.ImageSource = 'FESC.png';

            % Create Porcentaje
            app.Porcentaje = uilabel(app.EjercicioCuatro);
            app.Porcentaje.HorizontalAlignment = 'center';
            app.Porcentaje.FontSize = 24;
            app.Porcentaje.FontWeight = 'bold';
            app.Porcentaje.Position = [238 186 132 33];
            app.Porcentaje.Text = '';

            % Create CerrarApp
            app.CerrarApp = uibutton(app.EjercicioCuatro, 'push');
            app.CerrarApp.ButtonPushedFcn = createCallbackFcn(app, @CerrarAppButtonPushed,
true);
            app.CerrarApp.FontWeight = 'bold';
            app.CerrarApp.Position = [70 29 72 36];
            app.CerrarApp.Text = {'CERRAR'; 'APP'};

            % Show the figure after all components are created
            app.EjercicioCuatro.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)


        % Construct app
        function app = ReguladorDeIntensidad_exported
```

```
            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.EjercicioCuatro)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.EjercicioCuatro)
        end
    end
end
```

ANEXO 4. Seguridad por control de acceso

```
classdef ControlDeAccesoNoIdentificable_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        EjercicioSeis              matlab.ui.Figure
        ControldeAccesoPanel       matlab.ui.container.Panel
        Contrasena                 matlab.ui.control.EditField
        ContraseaEditFieldLabel_2  matlab.ui.control.Label
        Usuario                    matlab.ui.control.EditField
        UsuarioEditFieldLabel      matlab.ui.control.Label
        CerrarApp                  matlab.ui.control.Button
        Ingresar                   matlab.ui.control.Button
        LimpiarBoton               matlab.ui.control.Button
        FESC                       matlab.ui.control.Image
        ITSE                       matlab.ui.control.Image
        UNAM                       matlab.ui.control.Label
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Value changed function: Contrasena
        function ContrasenaValueChanged(app, event)
            contrasena = app.Contrasena.Value;
        end

        % Value changed function: Usuario
        function UsuarioValueChanged(app, event)
            usuario = app.Usuario.Value;
        end

        % Button pushed function: LimpiarBoton
        function LimpiarBotonButtonPushed(app, event)

            %Al presionar el botón Limpiar Botón se borrarán todos los
            %datos ingresados en las casillas Usuario y Contraseña
            app.Contrasena.Value = '';
            app.Usuario.Value = '';
        end

        % Button pushed function: Ingresar
        function IngresarButtonPushed(app, event)
            contrasena = app.Contrasena.Value;
            usuario = app.Usuario.Value;

            if (usuario == "John Fuentes" && contrasena == "FESC.2019")

                msgbox('USUARIO Y CONTRASEÑA CORRECTOS');
                close(app.EjercicioSeis);
                open('C:\Users\jonal\Documents\App Designer\Capítulo 3
ControlOnOff\ControlOnOff.mlapp');
```

```
            %Sentencia cuando Ingresamos erroneamente nuestro USUARIO
        elseif (usuario ~= "John Fuentes" && contrasena == "FESC.2019")
            msgbox('USUARIO INCORRECTO');

            %Sentencia cuando Ingresamos erroneamente nuestra
            %CONTRASEÑA
        elseif (usuario == "" && contrasena == "")
            errordlg('ERROR DEBES INGRESAR DATOS EN LAS CASILLAS');

        elseif (usuario == "John Fuentes" && contrasena ~= "FESC.2019")
            msgbox('CONTRASEÑA INCORRECTA');

        else
            msgbox("USUARIO Y CONTRASEÑA INCORRECTOS");
        end

    end

    % Button pushed function: CerrarApp
    function CerrarAppButtonPushed(app, event)
        close(app.EjercicioSeis);
    end

    % Size changed function: ControldeAccesoPanel
    function ControldeAccesoPanelSizeChanged(app, event)
        position = app.ControldeAccesoPanel.Position;

        usuario = app.Usuario.Value;
        contrasena = app.Contrasena.Value;

    end
end

% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Create EjercicioSeis and hide until all components are created
        app.EjercicioSeis = uifigure('Visible', 'off');
        app.EjercicioSeis.Color = [1 1 1];
        app.EjercicioSeis.Position = [100 100 606 416];
        app.EjercicioSeis.Name = 'CONTROL DE ACCESO';

        % Create UNAM
        app.UNAM = uilabel(app.EjercicioSeis);
        app.UNAM.HorizontalAlignment = 'center';
        app.UNAM.FontSize = 14;
        app.UNAM.FontWeight = 'bold';
        app.UNAM.Position = [117 295 373 80];
        app.UNAM.Text = {'UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO'; 'FACULTAD DE
ESTUDIOS SUPERIORES CUAUTITLÁN'; ''; 'INGENIERÍA EN TELECOMUNICACIONES, SISTEMAS '; 'Y
ELECTRÓNICA'};
```

```
            % Create ITSE
            app.ITSE = uiimage(app.EjercicioSeis);
            app.ITSE.Position = [500 301 107 116];
            app.ITSE.ImageSource = 'ITSE.png';

            % Create FESC
            app.FESC = uiimage(app.EjercicioSeis);
            app.FESC.Position = [1 301 100 100];
            app.FESC.ImageSource = 'FESC.png';

            % Create LimpiarBoton
            app.LimpiarBoton = uibutton(app.EjercicioSeis, 'push');
            app.LimpiarBoton.ButtonPushedFcn = createCallbackFcn(app,
@LimpiarBotonButtonPushed, true);
            app.LimpiarBoton.FontWeight = 'bold';
            app.LimpiarBoton.Position = [433 34 100 36];
            app.LimpiarBoton.Text = {'LIMPIAR'; 'DATOS'};

            % Create Ingresar
            app.Ingresar = uibutton(app.EjercicioSeis, 'push');
            app.Ingresar.ButtonPushedFcn = createCallbackFcn(app, @IngresarButtonPushed,
true);
            app.Ingresar.FontWeight = 'bold';
            app.Ingresar.Position = [250 34 100 36];
            app.Ingresar.Text = 'INGRESAR';

            % Create CerrarApp
            app.CerrarApp = uibutton(app.EjercicioSeis, 'push');
            app.CerrarApp.ButtonPushedFcn = createCallbackFcn(app, @CerrarAppButtonPushed,
true);
            app.CerrarApp.FontWeight = 'bold';
            app.CerrarApp.Position = [69 34 100 36];
            app.CerrarApp.Text = {'CERRAR'; 'APP'};

            % Create ControldeAccesoPanel
            app.ControldeAccesoPanel = uipanel(app.EjercicioSeis);
            app.ControldeAccesoPanel.TitlePosition = 'centertop';
            app.ControldeAccesoPanel.Title = 'Control de Acceso';
            app.ControldeAccesoPanel.SizeChangedFcn = createCallbackFcn(app,
@ControldeAccesoPanelSizeChanged, true);
            app.ControldeAccesoPanel.FontWeight = 'bold';
            app.ControldeAccesoPanel.FontSize = 24;
            app.ControldeAccesoPanel.Position = [174 101 260 161];

            % Create UsuarioEditFieldLabel
            app.UsuarioEditFieldLabel = uilabel(app.ControldeAccesoPanel);
            app.UsuarioEditFieldLabel.HorizontalAlignment = 'right';
            app.UsuarioEditFieldLabel.Position = [49 71 47 22];
            app.UsuarioEditFieldLabel.Text = 'Usuario';

            % Create Usuario
            app.Usuario = uieditfield(app.ControldeAccesoPanel, 'text');
            app.Usuario.ValueChangedFcn = createCallbackFcn(app, @UsuarioValueChanged,
true);
```

```
            app.Usuario.HorizontalAlignment = 'center';
            app.Usuario.Position = [111 71 100 22];

            % Create ContraseaEditFieldLabel_2
            app.ContraseaEditFieldLabel_2 = uilabel(app.ControldeAccesoPanel);
            app.ContraseaEditFieldLabel_2.HorizontalAlignment = 'right';
            app.ContraseaEditFieldLabel_2.Position = [28 24 68 22];
            app.ContraseaEditFieldLabel_2.Text = 'Contraseña';

            % Create Contrasena
            app.Contrasena = uieditfield(app.ControldeAccesoPanel, 'text');
            app.Contrasena.ValueChangedFcn = createCallbackFcn(app,
@ContrasenaValueChanged, true);
            app.Contrasena.HorizontalAlignment = 'center';
            app.Contrasena.Position = [111 24 100 22];

            % Show the figure after all components are created
            app.EjercicioSeis.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = ControlDeAccesoNoIdentificable_exported

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.EjercicioSeis)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.EjercicioSeis)
        end
    end
end
```

ANEXO 5. Sistemas de videovigilancia

```
classdef VideoVigilancia_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        Video                          matlab.ui.Figure
        RECONOCEButton                 matlab.ui.control.Button
        CerrarApp                      matlab.ui.control.Button
        CMARASDEVIDEOVIGILANCIALabel   matlab.ui.control.Label
        ITSE                           matlab.ui.control.Image
        FESC                           matlab.ui.control.Image
        UNAM                           matlab.ui.control.Label
        CUNALabel                      matlab.ui.control.Label
        LAPTOPLabel                    matlab.ui.control.Label
        Interruptor                    matlab.ui.control.ToggleSwitch
        INTERRUPTORSwitchLabel         matlab.ui.control.Label
        WEBCAM                         matlab.ui.control.UIAxes
        LAPTOP                         matlab.ui.control.UIAxes
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Value changed function: Interruptor
        function InterruptorValueChanged(app, event)

            grabar = app.Interruptor.Value;

            % Creamos los objetos para utilizar las cámaras web
            laptop = webcam();
            camaraUSB = webcam(2);

            cuadro2 = snapshot(laptop);
            lap = fliplr(image(app.LAPTOP, zeros(size(cuadro2), 'uint8')));
            axis(app.LAPTOP, 'image');

            cuadro1 = snapshot(camaraUSB);
            usb = image(app.WEBCAM, zeros(size(cuadro1), 'uint8'));
            axis(app.WEBCAM, 'image');

            if strcmp(grabar,'On')

                preview(camaraUSB,usb);
                preview(laptop,lap);
                pause;

            end
        end

        % Button pushed function: CerrarApp
        function CerrarAppPushed(app, event)
            % Cerramos la aplicación sin importar el estado en el que se
```

```
        % encuentre
        close(app.video)

    end
```

ANEXO 6. Implementación del algoritmo Viola – Jones en aplicación desarrollada

```
classdef VideoVigilancia_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        Video                          matlab.ui.Figure
        RECONOCEButton                 matlab.ui.control.Button
        CerrarApp                      matlab.ui.control.Button
        CMARASDEVIDEOVIGILANCIALabel   matlab.ui.control.Label
        ITSE                           matlab.ui.control.Image
        FESC                           matlab.ui.control.Image
        UNAM                           matlab.ui.control.Label
        CUNALabel                      matlab.ui.control.Label
        LAPTOPLabel                    matlab.ui.control.Label
        Interruptor                    matlab.ui.control.ToggleSwitch
        INTERRUPTORSwitchLabel         matlab.ui.control.Label
        WEBCAM                         matlab.ui.control.UIAxes
        LAPTOP                         matlab.ui.control.UIAxes
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Value changed function: Interruptor
        function InterruptorValueChanged(app, event)

            grabar = app.Interruptor.Value;

            % Creamos los objetos para utilizar las cámaras web
            laptop = webcam();
            camaraUSB = webcam(2);

            cuadro2 = snapshot(laptop);
            lap = fliplr(image(app.LAPTOP, zeros(size(cuadro2), 'uint8')));
            axis(app.LAPTOP, 'image');

            cuadro1 = snapshot(camaraUSB);
            usb = image(app.WEBCAM, zeros(size(cuadro1), 'uint8'));
            axis(app.WEBCAM, 'image');

            if strcmp(grabar,'On')

                preview(camaraUSB,usb);
                preview(laptop,lap);
                pause;

            end
        end

        % Button pushed function: CerrarApp
        function CerrarAppPushed(app, event)
```

```
            % Cerramos la aplicación sin importar el estado en el que se
            % encuentre
            close(app.Video)

        end

        % Button pushed function: RECONOCEButton
        function RECONOCEButtonPushed(app, event)

            % Creamos el detector de objetos utilizando el objeto en cascada
            % con el algoritmo de Viola-Jones y su modelo de clasificación
            detectorCara = vision.CascadeObjectDetector('FrontalFaceCART');

            % Umbral para definir los criterios necesarios para declarar
            % la detección en un área donde hay múltiples objetos
            detectorCara.MergeThreshold = 3;

            % Varible en la que guardaremos la información de la imagen que
            % vamos a utilizar para detectar los rostros
            fotoLectura = imread('caraitse.png');

            % El rostro detectado en la imagen de entrada "fotoLectura"
            % será encerrado en una caja, por eso llamaremos de esa forma a
            % esta variable
            caja = detectorCara(fotoLectura);

            % Indicamos las propiedades de bboxes como figura en la que
            % quedarán encerrados los rostros y el título que se colocará
            conteoRostros =
insertObjectAnnotation(fotoLectura,'rectangle',caja,[1:size(caja,1)]);

            % Creamos una figura en donde se pueda mostrar la foto y la
            % detección de los rostros, será nombrada como 'Ventana de
            % Detección de Rostros'
            figure('Name','Ventana de Detección de Rostros','NumberTitle','off');

            % Mostramos los resultados guardados en la variable:
            % propiedadesCaja, de no ponerlo se mostraría una gráfica en
            % blanco
            imshow(conteoRostros)

            % Ponemos un titulo a la venta que muestra los rostros detectados
            title('ROSTROS OBSERVADOS');

            %Este dato nos indica cuántas caras detecto el sistema
            detecteCara=size(caja);

            %Matriz fija que guarda todas las caras detectadas
            todas=zeros(50,50,detecteCara(1,1));

            %Función para etiquetar y recortar los rostros detectados
            for i=1:detecteCara
                x=caja(i,1); y=caja(i,2); ix=caja(i,3); iy=caja(i,4);
```

```
            %imcrop corta la cara que hay en la imagen
            cara=imcrop(fotoLectura,[x y ix iy]);

            %guardamos la cara en la variable cortada
            size(cara);

            %Abrimos una ventana nueva y la mostramos
            figure('Name','Los rostros detectados son:','NumberTitle','off');
            imshow(cara)

            %Pasámos la cara detectada a un tamaño de 50 x 50
            cara50R=imresize(cara(:,:,1),[50 50]);
            todas(:,:,i)=cara50R;
        end
    end
end

% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Create Video and hide until all components are created
        app.Video = uifigure('Visible', 'off');
        app.Video.Color = [1 1 1];
        app.Video.Position = [100 100 640 480];
        app.Video.Name = 'Videovigilancia';

        % Create LAPTOP
        app.LAPTOP = uiaxes(app.Video);
        app.LAPTOP.PlotBoxAspectRatio = [1.32067510548523 1 1];
        app.LAPTOP.XColor = 'none';
        app.LAPTOP.XTick = [];
        app.LAPTOP.XTickLabel = '';
        app.LAPTOP.YColor = 'none';
        app.LAPTOP.YTick = [];
        app.LAPTOP.YTickLabel = '';
        app.LAPTOP.ZColor = 'none';
        app.LAPTOP.Position = [12 41 260 214];



        % Create WEBCAM
        app.WEBCAM = uiaxes(app.Video);
        app.WEBCAM.PlotBoxAspectRatio = [1.31645569620253 1 1];
        app.WEBCAM.TickLabelInterpreter = 'none';
        app.WEBCAM.XColor = 'none';
        app.WEBCAM.XTick = [];
        app.WEBCAM.YColor = 'none';
        app.WEBCAM.YTick = [];
        app.WEBCAM.ZColor = 'none';
        app.WEBCAM.Position = [370 41 260 214];
```

```
            % Create INTERRUPTORSwitchLabel
            app.INTERRUPTORSwitchLabel = uilabel(app.Video);
            app.INTERRUPTORSwitchLabel.HorizontalAlignment = 'center';
            app.INTERRUPTORSwitchLabel.FontSize = 10;
            app.INTERRUPTORSwitchLabel.FontWeight = 'bold';
            app.INTERRUPTORSwitchLabel.Position = [282 87 78 22];
            app.INTERRUPTORSwitchLabel.Text = 'INTERRUPTOR';

            % Create Interruptor
            app.Interruptor = uiswitch(app.Video, 'toggle');
            app.Interruptor.ValueChangedFcn = createCallbackFcn(app,
@InterruptorValueChanged, true);
            app.Interruptor.FontWeight = 'bold';
            app.Interruptor.Position = [306 141 29 67];

            % Create LAPTOPLabel
            app.LAPTOPLabel = uilabel(app.Video);
            app.LAPTOPLabel.HorizontalAlignment = 'center';
            app.LAPTOPLabel.FontWeight = 'bold';
            app.LAPTOPLabel.Position = [56 287 64 22];
            app.LAPTOPLabel.Text = 'LAPTOP';

            % Create CUNALabel
            app.CUNALabel = uilabel(app.Video);
            app.CUNALabel.HorizontalAlignment = 'center';
            app.CUNALabel.FontWeight = 'bold';
            app.CUNALabel.Position = [516 287 40 22];
            app.CUNALabel.Text = 'CUNA';

            % Create UNAM
            app.UNAM = uilabel(app.Video);
            app.UNAM.HorizontalAlignment = 'center';
            app.UNAM.FontSize = 14;
            app.UNAM.FontWeight = 'bold';
            app.UNAM.Position = [134 361 373 80];
            app.UNAM.Text = {'UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO'; 'FACULTAD DE
ESTUDIOS SUPERIORES CUAUTITLÁN'; ''; 'INGENIERÍA EN TELECOMUNICACIONES, SISTEMAS '; 'Y
ELECTRÓNICA'};

            % Create FESC
            app.FESC = uiimage(app.Video);
            app.FESC.Position = [20 353 100 100];
            app.FESC.ImageSource = 'FESC.png';

            % Create ITSE
            app.ITSE = uiimage(app.Video);
            app.ITSE.Position = [515 353 107 116];
            app.ITSE.ImageSource = 'ITSE.png';

            % Create CMARASDEVIDEOVIGILANCIALabel
            app.CMARASDEVIDEOVIGILANCIALabel = uilabel(app.Video);
            app.CMARASDEVIDEOVIGILANCIALabel.HorizontalAlignment = 'center';
            app.CMARASDEVIDEOVIGILANCIALabel.FontSize = 24;
            app.CMARASDEVIDEOVIGILANCIALabel.FontWeight = 'bold';
```

```
            app.CMARASDEVIDEOVIGILANCIALabel.Position = [211 271 220 54];
            app.CMARASDEVIDEOVIGILANCIALabel.Text = {'CÁMARAS DE'; 'VIDEOVIGILANCIA'};

            % Create CerrarApp
            app.CerrarApp = uibutton(app.Video, 'push');
            app.CerrarApp.ButtonPushedFcn = createCallbackFcn(app, @CerrarAppPushed, true);
            app.CerrarApp.IconAlignment = 'center';
            app.CerrarApp.FontWeight = 'bold';
            app.CerrarApp.Position = [555 6 75 36];
            app.CerrarApp.Text = {'CERRAR'; 'APP'};

            % Create RECONOCEButton
            app.RECONOCEButton = uibutton(app.Video, 'push');
            app.RECONOCEButton.ButtonPushedFcn = createCallbackFcn(app,
@RECONOCEButtonPushed, true);
            app.RECONOCEButton.IconAlignment = 'center';
            app.RECONOCEButton.FontWeight = 'bold';
            app.RECONOCEButton.Position = [282 13 89 29];
            app.RECONOCEButton.Text = 'RECONOCE';

            % Show the figure after all components are created
            app.Video.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = VideoVigilancia_exported

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.Video)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.Video)
        end
    end
end
```