ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE IoT

ASIGNATURA:	Desarrollo de IoT
PROFESOR:	Ing. Vanessa Guevara
PERÍODO ACADÉMICO:	2024-B

PROYECTO FINAL

TÍTULO:

Aplicación con IoT

Integrantes:

David Muela

Mateo Moran

Eduardo Lincango

Alejandro Gutiérrez

1. PROPÓSITO DE LA PRÁCTICA

Este proyecto tiene como objetivo desarrollar un control de luces en una casa que emplee tecnología IoT para supervisar y regular de forma remota.

2. MATERIALES:

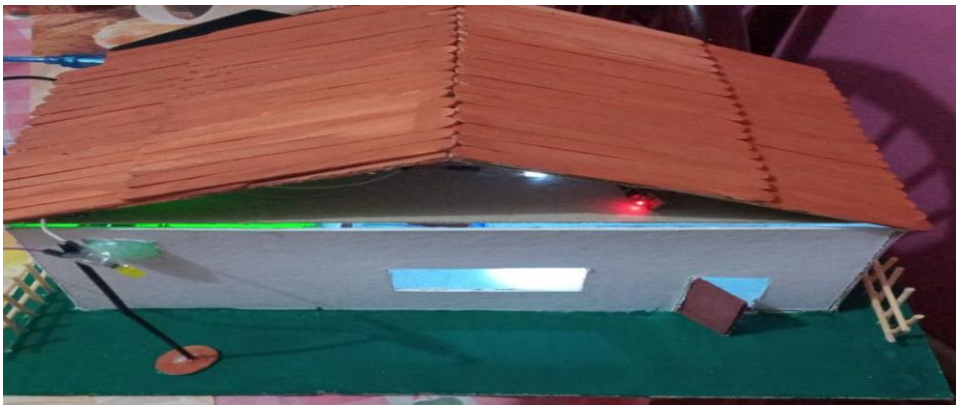
- Placa Arduino
- Módulo Wi-Fi
- Sensor de movimiento PIR
- Fotorresistencia
- Protoboard y cables de conexión.
- Leds
- Resistencias

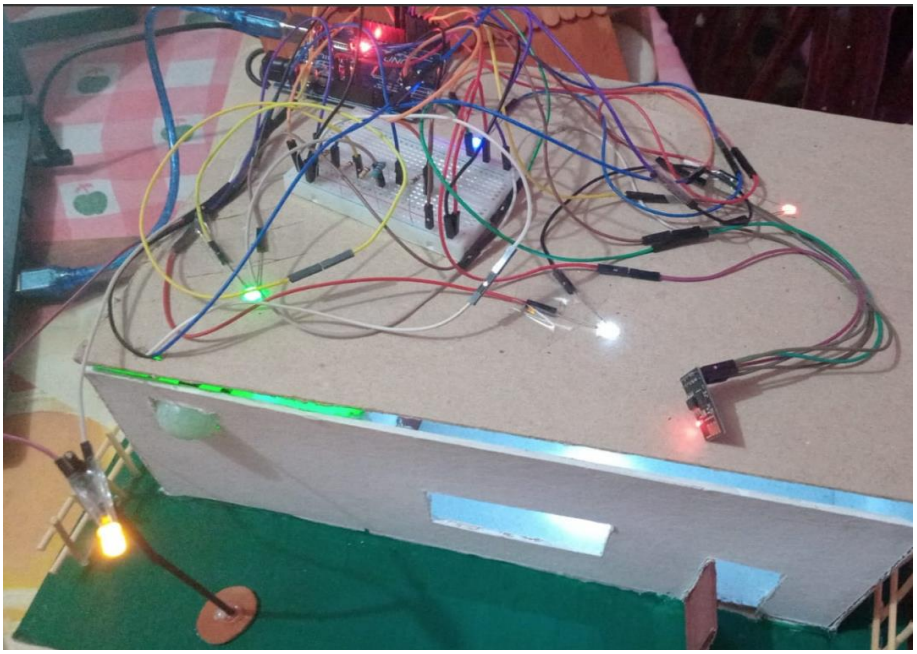
3. INSTRUCCIONES

- **Construcción de la maqueta**
    a. Diseñar y construir una maqueta que represente una casa con al menos los siguientes espacios: una habitación, cocina, sala, baño y patio.

Para la realización de la maqueta re requirió de algunos materiales como cartón prensado, palitos de helado, pintura, silicona caliente y los materiales antes listados para representar una casa con habitación, cocina, sala y baño.

El proceso del armado consistió en pegar las paredes de cartón prensado en conjunto el techo se lo armó con palitos de helado para dar una forma realística tal y como se muestra en las siguientes figuras.

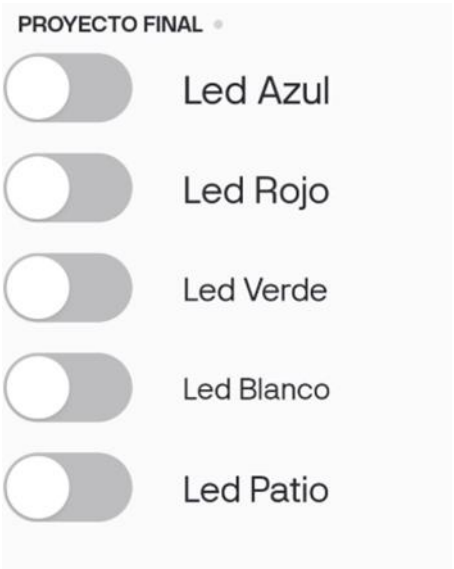




- **Aplicación Móvil (Blynk)**
    a. Configurar una aplicación móvil utilizando Blynk para permitir al usuario:
        i. Controlar la iluminación de la casa. El usuario debe encender y apagar LEDs que representen las luces de la habitación, cocina, sala, baño y patio.

Interfaz Web

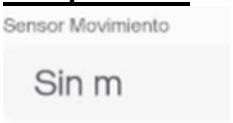
Interfaz Móvil



        ii. Enviar notificaciones al usuario cuando se detecte movimiento en la casa.

Interfaz Web

Interfaz móvil



Explicación

Primeramente, para realizar la conexión con blink requeríamos las credenciales que Blynk nos proporcionó que es el id, el user y el token con eso podemos conectarnos

```
// Colocar sus credenciales
#define BLYNK_TEMPLATE_ID "TMPL28rndTHcH"
#define BLYNK_TEMPLATE_NAME "PROYECTO FINAL"
#define BLYNK_AUTH_TOKEN "Q1-jgzuW2fAEJIo1W1KiG8NbUVJfIa60"
#define BLYNK_PRINT Serial
```

Segundo, agregamos nuestras credenciales en setup () con los como el siguiente fragmento de código

```
  // Conexión a Blynk
  Blynk.begin(token, wifi, ssid, pass);
```

Tercero, realizamos la lógica para cada LED de cada habitación para que podamos encenderlo y apagarlo con Blynk. Mediante una variable estado, obtenemos los valores 0 (apagado) y 1 (encendido) con el fin de enviar a digitalWrite para cambiar el estado del LED, permitiendo alternar entre encendido y apagado

```
  // Funciones para encender y apagar los LEDs con los switches de Blynk

BLYNK_WRITE(V0) { // Pin virtual V0 - LED Rojo
  int estado = param.asInt();
  digitalWrite(pinRojo, estado);
}

BLYNK_WRITE(V1) { // Pin virtual V1 - LED Azul
  int estado = param.asInt();
  digitalWrite(pinAzul, estado);
}

BLYNK_WRITE(V2) { // Pin virtual V2 - LED Verde
  int estado = param.asInt();
  digitalWrite(pinVerde, estado);
}

BLYNK_WRITE(V3) { // Pin virtual V3 - LED Blanco
  int estado = param.asInt();
  digitalWrite(pinBlanco, estado);
}

BLYNK_WRITE(V4) { // Pin virtual V4 - LED Amarillo del Patio
  int estado = param.asInt();
  digitalWrite(pinAmarillo, estado);
}
```

Adicionalmente, para enviar datos a Blynk cuando se detecta movimiento, utilizamos el siguiente código. Este verifica con una condición con if y, mediante un pin virtual, envía un mensaje indicando si se ha detectado movimiento o no

```
if (movimiento == HIGH) {
  Serial.println("Movimiento detectado");
  Blynk.virtualWrite(V5, "Movimiento detectado");
} else {
  Blynk.virtualWrite(V5, "Sin movimiento");
}
```

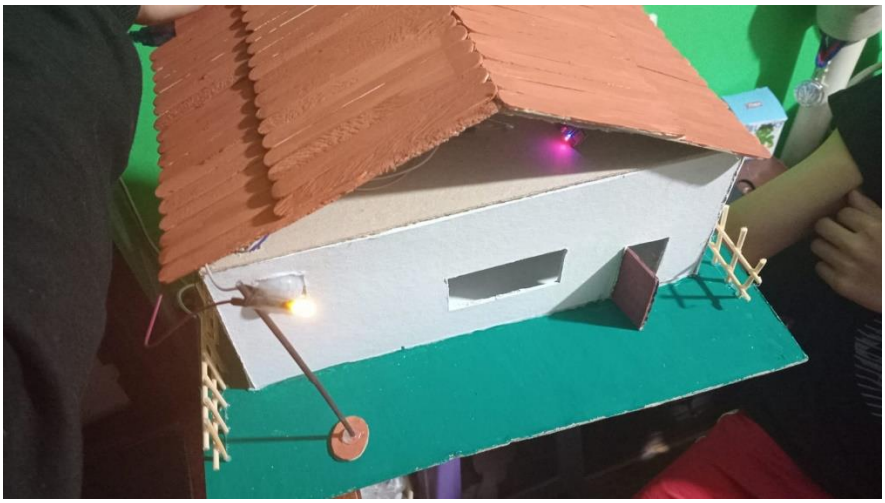
Finalmente, para establecer y mantener la conexión debemos agregar en el apartado de loop () el siguiente código

```
// Ejecuta Blynk.run()
Blynk.run();
```

- **Iluminación automática del patio:**
    a. Activar las luces del patio automáticamente al detectar condiciones de baja luminosidad (simulando el anochecer).

    Foto de la luz apagada porque hay luz de ambiente:

    Foto de la luz encendida cuanto hay poca luz



- **Integración con Realidad Aumentada**
    a. Usar OpenCV para identificar en tiempo real un marcador de ARUCO captados por una cámara.

    Gracias a OpenCV se pueden rastrear marcadores ARUCO mediante la cámara y en tiempo real. Para este proyecto se instalaron las librerías OpenCV y numpy.

```
PS C:\Users\PC\Desktop\bas eia\PROYECTO_FINAL> pip install opencv-python
  Downloading opencv_python-4.11.0.86-cp37-abi3-win_amd64.whl.metadata (20 kB)
Requirement already satisfied: numpy>=1.21.2 in c:\users\pc\appdata\local\programs\python\python312\lib\site-pack
grams\python\python312\lib\site-packages (from opencv-python) (1.26.4)
Downloading opencv_python-4.11.0.86-cp37-abi3-win_amd64.whl (39.5 MB)
   ---------------------------------------- 39.5/39.5 MB 14.4 MB/s eta 0:00:00
Installing collected packages: opencv-python
Successfully installed opencv-python-4.11.0.86

[notice] A new release of pip is available: 24.3.1 -> 25.0
[notice] To update, run: C:\Users\PC\AppData\Local\Programs\Python\Python312\python.exe -m pip install --upgrade
pip
PS C:\Users\PC\Desktop\bas eia\PROYECTO_FINAL>
```

```
PS C:\Users\PC\Desktop\bas eia\PROYECTO_FINAL> pip install numpy
Requirement already satisfied: numpy in c:\users\pc\appdata\local\programs\python\python312\lib\site-packages (1.
26.4)

[notice] A new release of pip is available: 24.3.1 -> 25.0
[notice] To update, run: C:\Users\PC\AppData\Local\Programs\Python\Python312\python.exe -m pip install --upgrade
pip
```

Para la lectura optima de los marcadores ARUCO usamos el siguiente script de Python:

```python
import cv2
import numpy as np
import cv2.aruco as aruco
import requests

# credenciales blynk
token = "Q1-jgzuW2fAEJIo1WlKiG8NbUVJfIa60"
pinLuz = 'V6'
blynk_api_url1 = f'https://blynk.cloud/external/api/get?token={token}&{pinLuz}'

# Blynk: Obtener los valores de los sensores de luz y distancia
def obtener_valores_blynk():
    valorSensorLuz = requests.get(blynk_api_url1)
    valorLuz = valorSensorLuz.text
    return valorLuz

parametros = cv2.aruco.DetectorParameters()
aruco_diccionario = aruco.getPredefinedDictionary(aruco.DICT_7X7_100)
captura = cv2.VideoCapture(0)
imagen_luz = cv2.imread("foco2.png")  # Imagen para el marcador de luz
valorLuz = ''
while True:
    lectura, frame = captura.read()
    cuadro_gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    detector = aruco.ArucoDetector(aruco_diccionario, parametros)
    esquinas, identificador, puntosRechazados = detector.detectMarkers(cuadro_gris)

    valorLuz= obtener_valores_blynk()

    if identificador is not None:
        aruco.drawDetectedMarkers(frame, esquinas, identificador)

        for i in range(len(identificador)):
            marker_corners = esquinas[i][0]

            x, y, w, h = cv2.boundingRect(marker_corners)

            if identificador[i] == 0:
                imagen_luz_resized = cv2.resize(imagen_luz, (w, h))
                frame[y:y+h, x:x+w] = imagen_luz_resized
                cv2.putText(frame, f"Luz: {valorLuz}", (x, y - 10), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 255, 0), 2)

    cv2.imshow('Aruco', frame)

    # Salir con la tecla 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Liberar la cámara y cerrar la ventana
captura.release()
cv2.destroyAllWindows()
```

Donde se inicializa la cámara con

```
captura = cv2.VideoCapture(0)
```

Luego, se condiguran los parámetros del detector y el diccionario de marcadores ARUCO.

```
parametros = cv2.aruco.DetectorParameters()
aruco_diccionario = aruco.getPredefinedDictionary(aruco.DICT_7X7_100)
```

Se captura un fotograma y se convierte a escala de grises, además se usa la función detectMarkers() para detectar si hay un marcador ARUCO en la imagen

```
lectura, frame = captura.read()
cuadro_gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
detector = aruco.ArucoDetector(aruco_diccionario, parametros)
esquinas, identificador, puntosRechazados = detector.detectMarkers(cuadro_gris)
```

Finalmente, si se detectan marcadores ARUCO se muestra una imagen sobre este marcador.

b. Vincular el sensor de luz con la aplicación de Realidad Aumentada.

El objetivo es que los valores de la fotorresistencia o sensor de luz se muestren en tiempo real cuando se detecte un marcador. Para esto hacemos uso de Blynk y su API, que permite la comunicación con sensores remotos.

Para esto, se definen las credenciales; token, pin virtual y la url específica para conectar con la fotorresistencia y sus valores.

```
token = "Q1-jgzuW2fAEJIo1WlKiG8NbUVJfIa60"
pinLuz = 'V6'
blynk_api_url1 = f'https://blynk.cloud/external/api/get?token={token}&{pinLuz}'
```

Definimos una función, la cual hace una petición a Blynk para obtener el valor de la fotorresistencia.

```
def obtener_valores_blynk():
      valorSensorLuz = requests.get(blynk_api_url1)
      valorLuz = valorSensorLuz.text
      return valorLuz
```

Finalmente, llamamos la función para actualizar el valor del sensor en tiempo real

```
valorLuz = obtener_valores_blynk()
```

c. Superponer el valor del sensor y una imagen cuando se detecte el marcador.

El objetivo al detectar un marcador ARUCO es superponer una imagen y mostrar el valor de la fotorresistencia. Para esto:

Se carga la imagen que se superpone cuando el marcador es detectado.

```
Imagen_luz = cv2.imread("foco2.png")
```

Si se detecta un marcado con ID = 0 se redimensiona la imagen para que tenga el mismo tamaño que el marcador, se sobrepone la imagen sobre el marcador y se muestra el valor arriba del marcador.

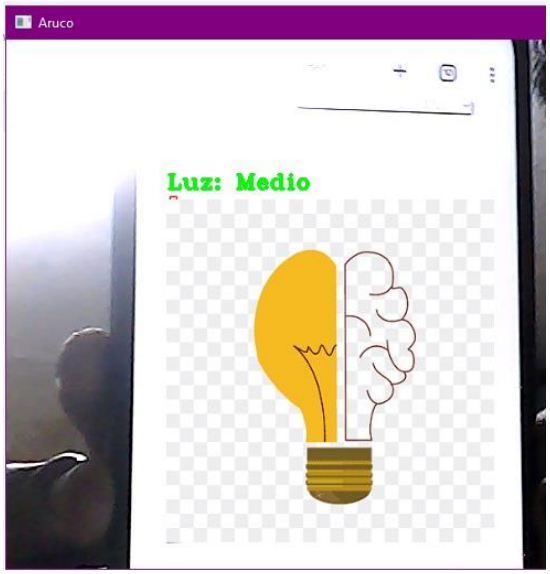
```
for i in range(len(identificador)):
            marker_corners = esquinas[i][0]

            x, y, w, h = cv2.boundingRect(marker_corners)

            if identificador[i] == 0:
                imagen_luz_resized = cv2.resize(imagen_luz, (w, h))
                frame[y:y+h, x:x+w] = imagen_luz_resized
                cv2.putText(frame, f"Luz: {valorLuz}", (x, y - 10),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 255, 0), 2)

    cv2.imshow('Aruco', frame)
```

Foto del marcador Aruco en funcionamiento.



CONCLUSIONES

- La implementación de tecnología IOT permite un control remoto de la iluminación del hogar con el fin de mejorar la comodidad y facilidad de gestinar el manejo de las luces.

- Con la ayuda de OpenCV y marcadores ARUCO proporciona una visualización interactiva de los datos de los sensores.

- El proyecto demuestra la utilidad de integrar la Realidad Aumentada (RA) con sensores IoT utilizando Python y Blynk. La combinación de estas tecnologías permite una interacción en tiempo real con elementos físicos y digitales.

RECOMENDACIONES

- Al momento de realizar las conexiones con la board, es fundamental verificar que la conexión Wi-Fi funcione correctamente para evitar tener que desarmar y volver a conectar los sensores, leds y los demás componentes.

- Para mejorar el rendimiento en dispositivos con menos capacidad se recomienda reducir la resolución del video capturado.

- Incorporar sensores adicionales como temperatura, humedad o proximidad puede hacer que él proyecto sea más interactivo.