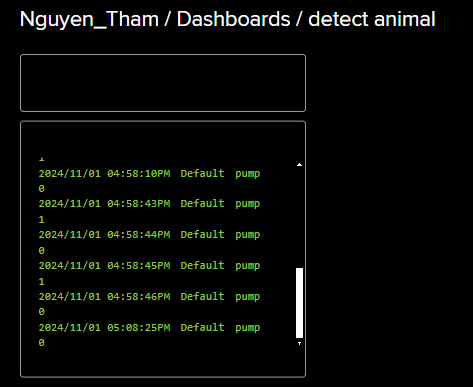
Lab report

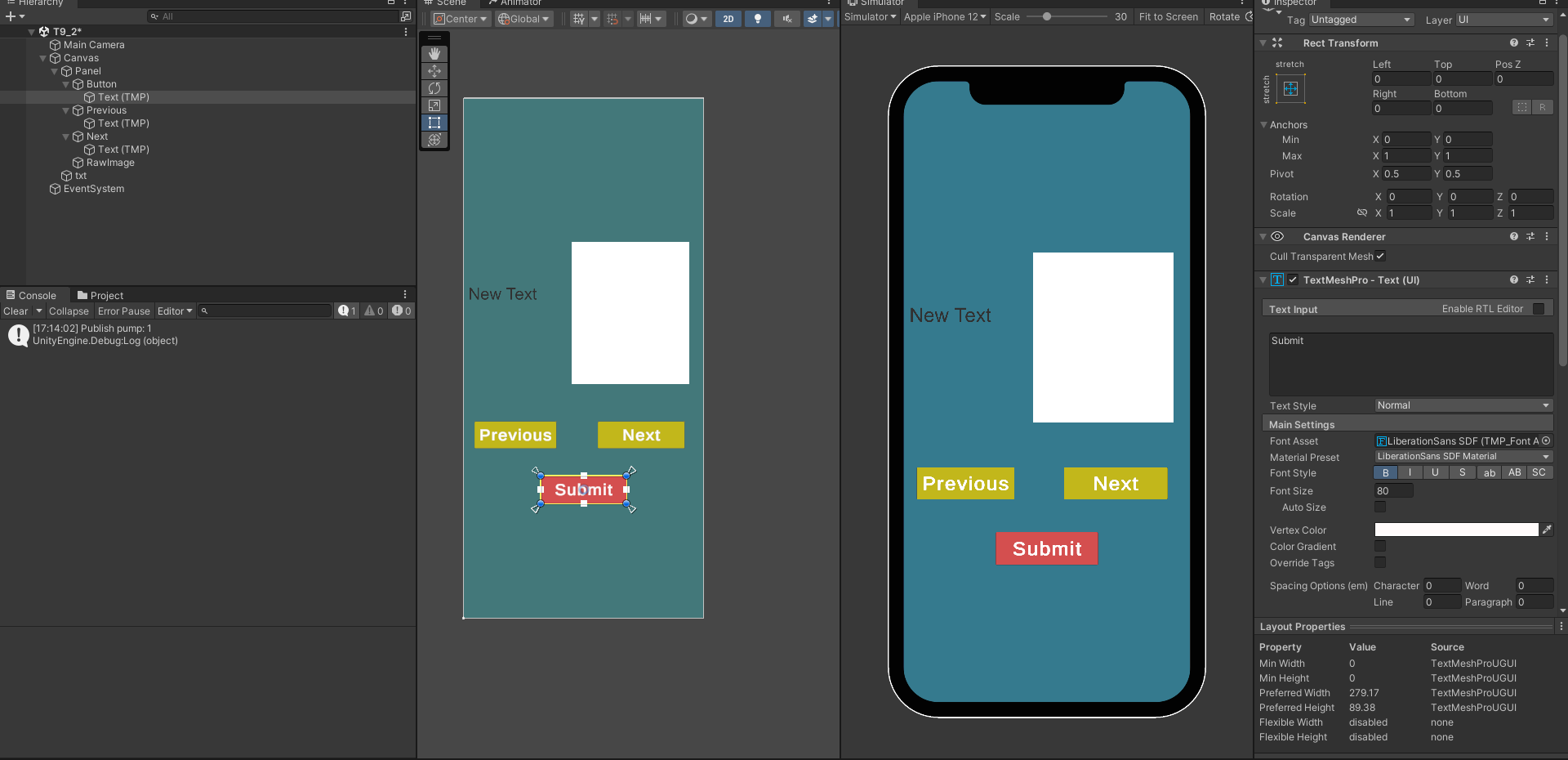
21015721 Nguyễn Thị Hồng Thắm

Đầu tiên tạo feeds và dashbroad

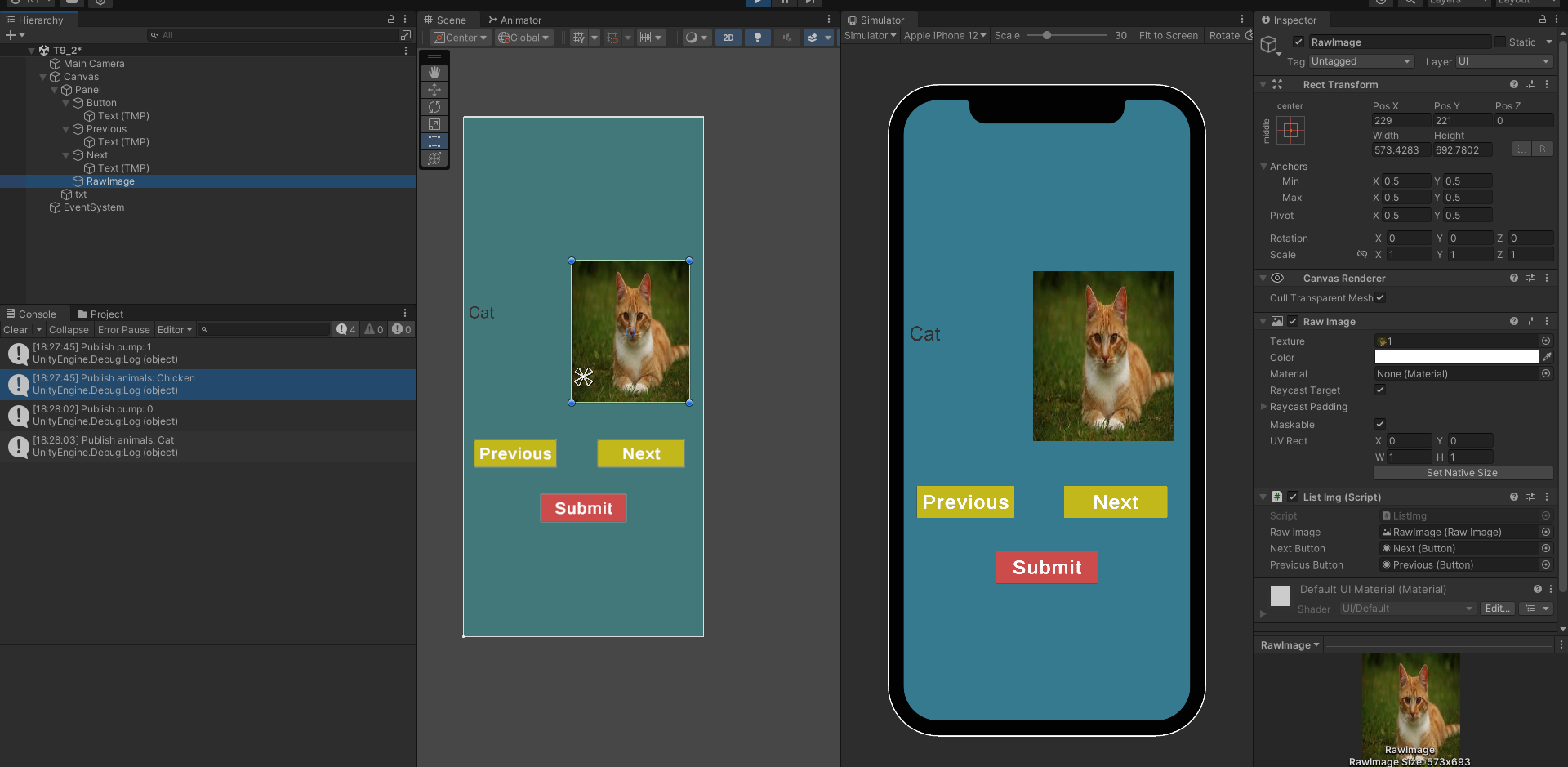




Sau đó tạo scence, có text để hiện thị tên loại đối tượng, rawimg để chứa mảng hình và button gồm 3 button: Previous hiện thị ảnh phía trước, Next hiện thị ảnh kế tiếp, Submit để nhận dạng con vật



* Tạo file script để truyền mảng img vào hình
* using System.Collections;
* using System.Collections.Generic;
* using UnityEngine;
* using UnityEngine.UI;
* public class ListImg : MonoBehaviour
* {
* public RawImage rawImage;
* public Button nextButton;
* public Button previousButton;
* private Texture[] images;
* private int currentIndex = 0;
* void Start()
* {
* images = Resources.LoadAll<Texture>("img/list");
* UpdateImage();
* nextButton.onClick.AddListener(NextImage);
* previousButton.onClick.AddListener(PreviousImage);
* }
* void UpdateImage()
* {
* if (images.Length > 0)
* {
* rawImage.texture = images[currentIndex];
* }
* }
* void NextImage()
* {
* currentIndex = (currentIndex + 1) % images.Length;
* UpdateImage();
* }
* void PreviousImage()
* {
* currentIndex = (currentIndex - 1 + images.Length) % images.Length;
* UpdateImage();
* }
* }
* Gắn file script đã tạo vào RawImg gán các thuộc tính vào inpector



* Tạo file script Detect\_animals\_MQTT để xử lý sự kiện trong từng button và đẩy dữ liệu lên adafruit

using System.Collections;

using System.Collections.Generic;

using System.Text;

using Unity.Barracuda;

using UnityEngine;

using UnityEngine.UI;

using uPLibrary.Networking.M2Mqtt.Messages;

using uPLibrary.Networking.M2Mqtt;

using Unity.VisualScripting;

public class Detect\_animals\_MQTT : MonoBehaviour

{

public Button button;

public Text txtCount;

public RawImage imgFace;

public NNModel modelAsset;

private MqttClient client;

private string brokerAddress = "io.adafruit.com";

private string username = "Nguyen\_Tham";

private string key = "aio\_XhAG272Co1QkzWifhdnfZZQkm02H";

private Model m\_RuntimeModel;

private IWorker m\_Worker;

private int count, channels;

private string[] feeds = { "pump","animals" };

private string[] label = { "Cat", "Chicken", "Dog" ,"Krixi","Mucca"};

private Queue<System.Action> actionQueue = new Queue<System.Action>();

void Awake()

{

m\_RuntimeModel = ModelLoader.Load(modelAsset);

m\_Worker = WorkerFactory.CreateWorker(WorkerFactory.Type.CSharp, m\_RuntimeModel);

channels = 3; // Number of color channels

}

void Start()

{

count = 0;

client = new MqttClient(brokerAddress);

client.MqttMsgPublishReceived += OnMessageReceived;

client.Connect(username, username, key);

client.Subscribe(new string[] { $"{username}/feeds/animals" }, new byte[] { MqttMsgBase.QOS\_LEVEL\_AT\_LEAST\_ONCE });

button.onClick.AddListener(OnButtonClick);

}

void OnButtonClick()

{

/\*count = count == 0 ? 1 : 0;

txtCount.text = count.ToString();

SendMQTTMessage("pump", count.ToString());

string predictedLabel = PredictImage();

txtCount.text = predictedLabel;\*/

count = count == 0 ? 1 : 0;

txtCount.text = count.ToString();

SendMQTTMessage("pump", count.ToString());

string predictedLabel = PredictImage();

txtCount.text = predictedLabel;

SendMQTTMessage("animals", predictedLabel);

}

string PredictImage()

{

if (imgFace.texture == null)

{

Debug.LogWarning("No image to predict.");

return "No Image";

}

Tensor input = new Tensor(imgFace.texture, channels);

m\_Worker.Execute(input);

Tensor output = m\_Worker.PeekOutput();

int[] maxIndex = output.ArgMax();

return label[maxIndex[0]];

}

void SendMQTTMessage(string topic, string value)

{

string formattedTopic = $"{username}/feeds/{topic}";

client.Publish(formattedTopic, Encoding.UTF8.GetBytes(value), MqttMsgBase.QOS\_LEVEL\_AT\_LEAST\_ONCE, false);

Debug.Log($"Publish {topic}: {value}");

}

void OnMessageReceived(object sender, MqttMsgPublishEventArgs e)

{

string topic = e.Topic;

string message = Encoding.UTF8.GetString(e.Message);

if (topic.Contains("pump"))

{

actionQueue.Enqueue(() => UpdateStatusText(message));

}

}

void UpdateStatusText(string text)

{

txtCount.text = text; // Update the displayed count

}

void Update()

{

while (actionQueue.Count > 0)

{

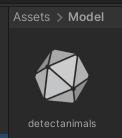
actionQueue.Dequeue().Invoke();

}

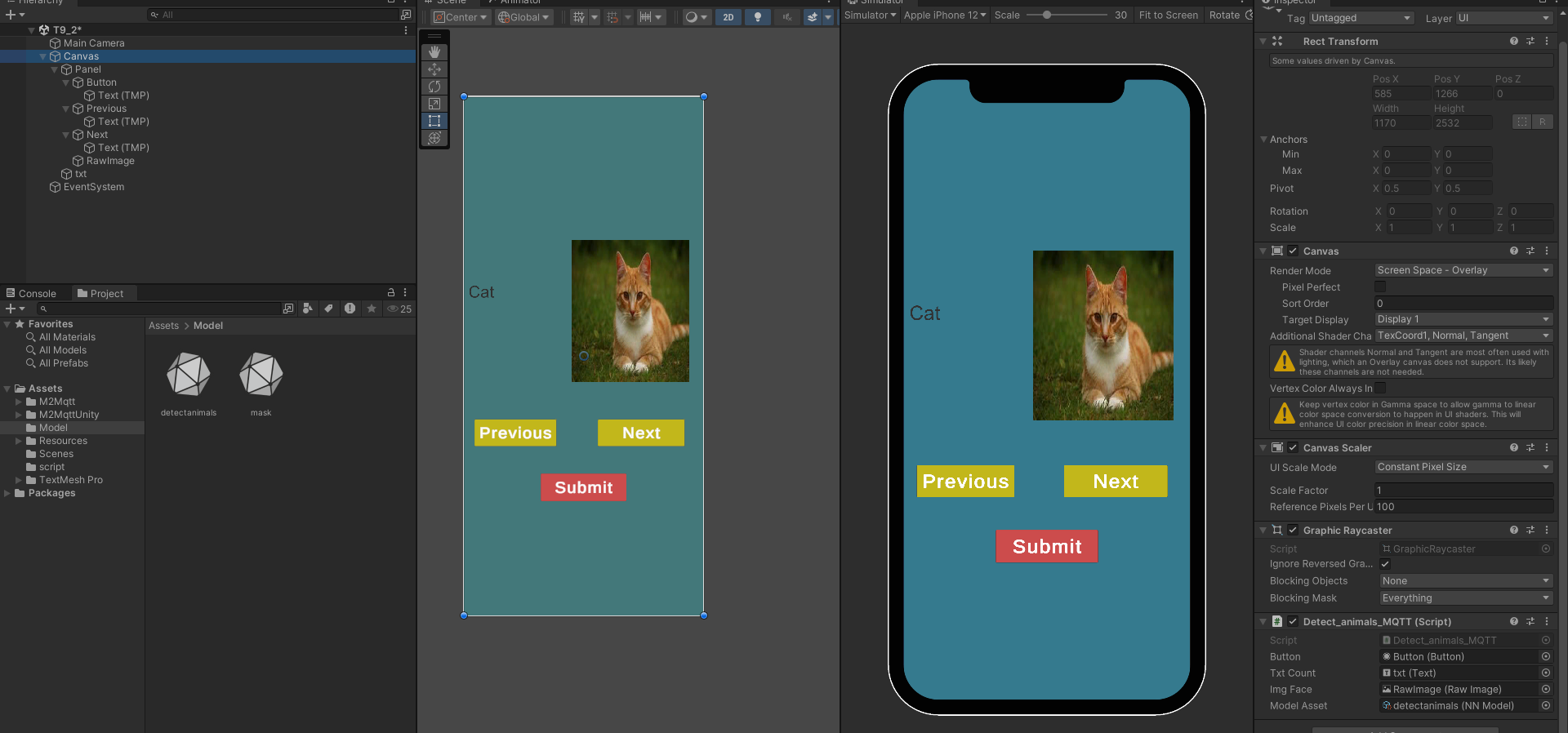
}

}

Import model đã train trên teachablemachine chuyển đổi h5 sang onnx

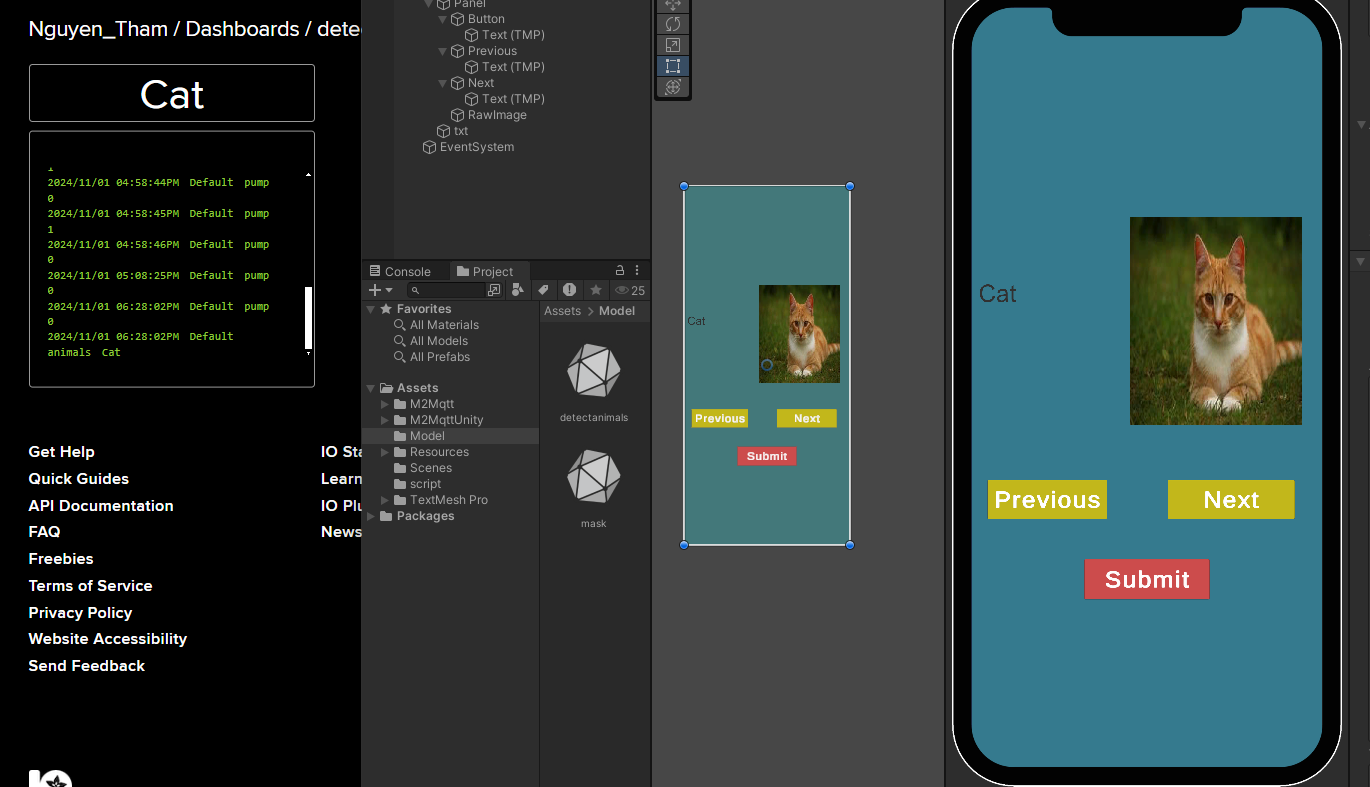


* Gắn file script Detect\_animals\_MQTT vào canva gán các thuộc tính vào inpector

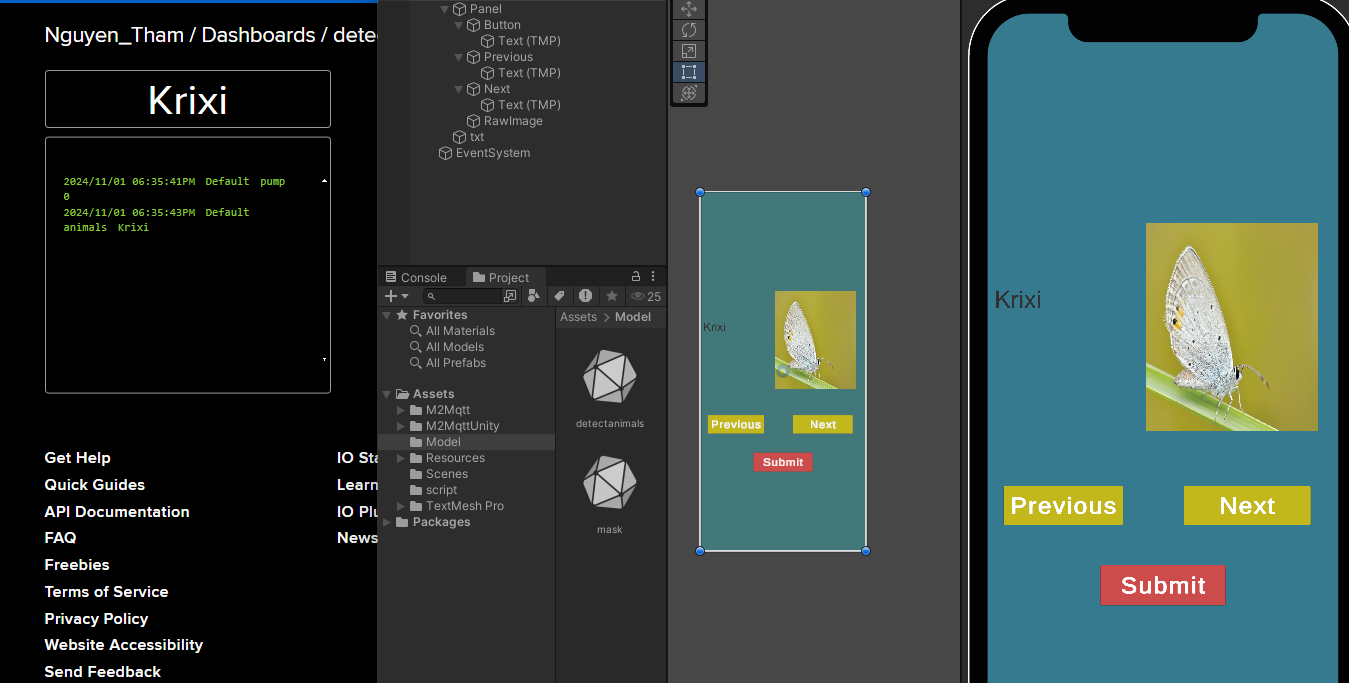


Kết quả

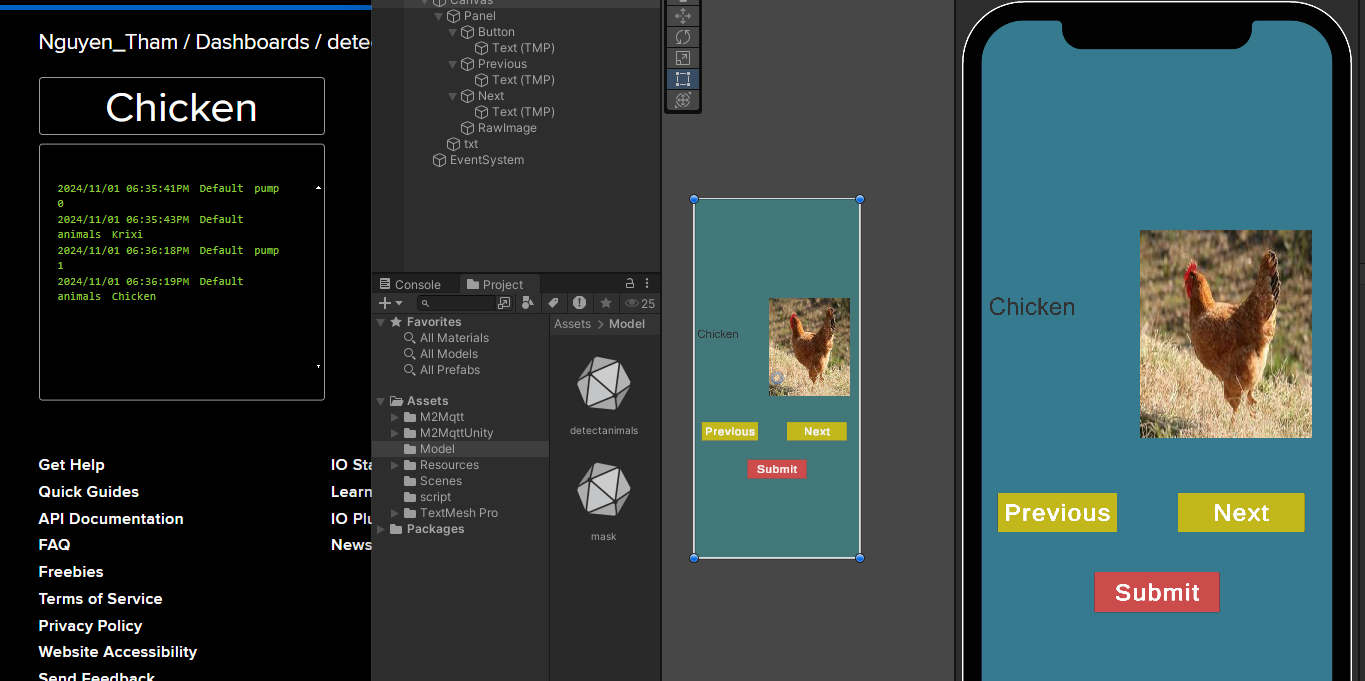
Nhận dạng Cat



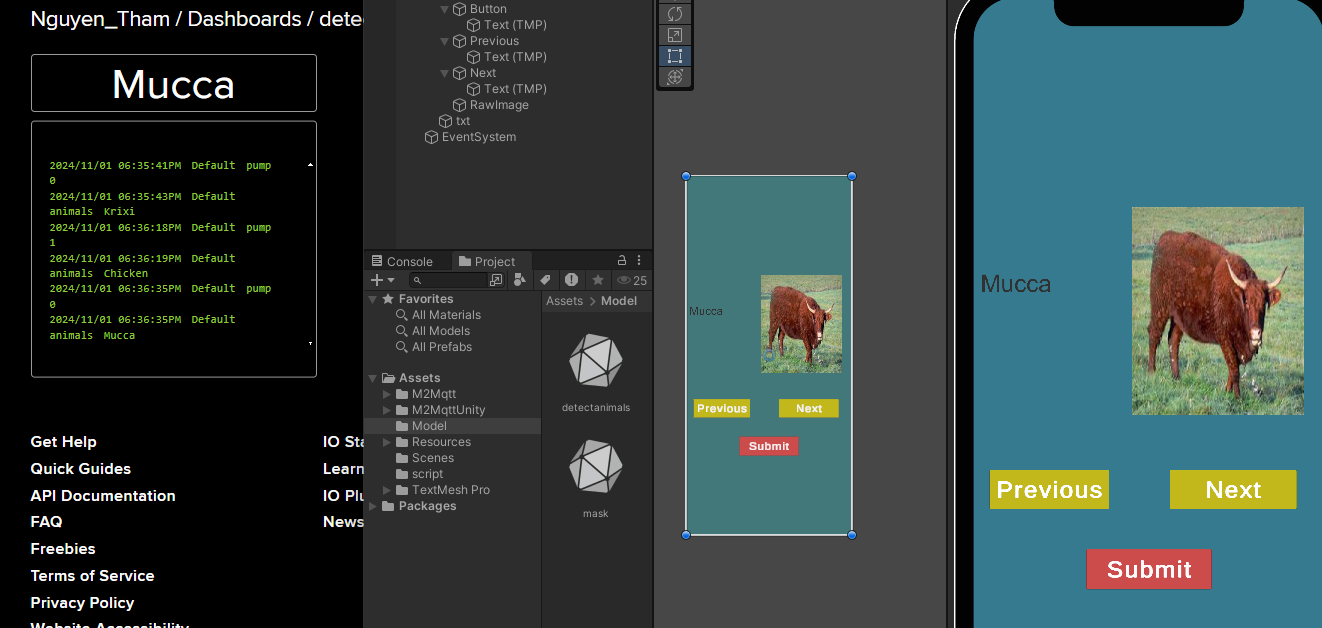
Nhận dạng Krixi



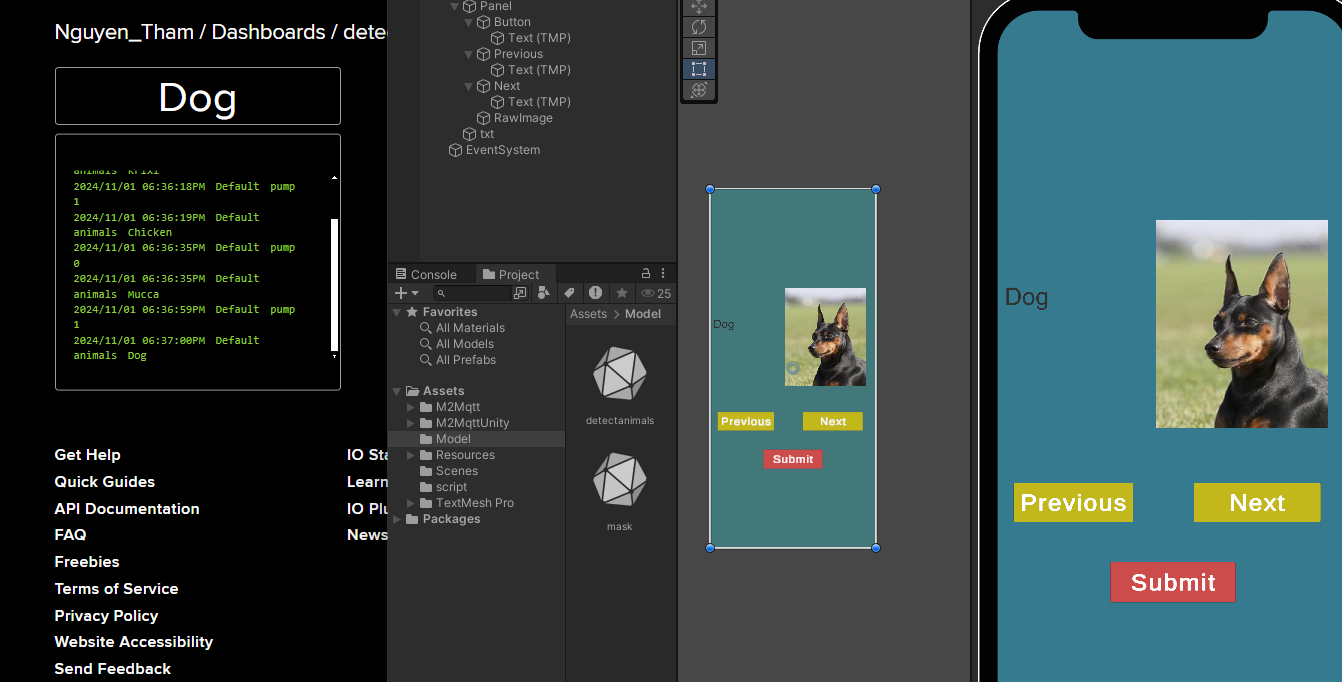
Nhận dạng chicken



Nhận dạng mucca



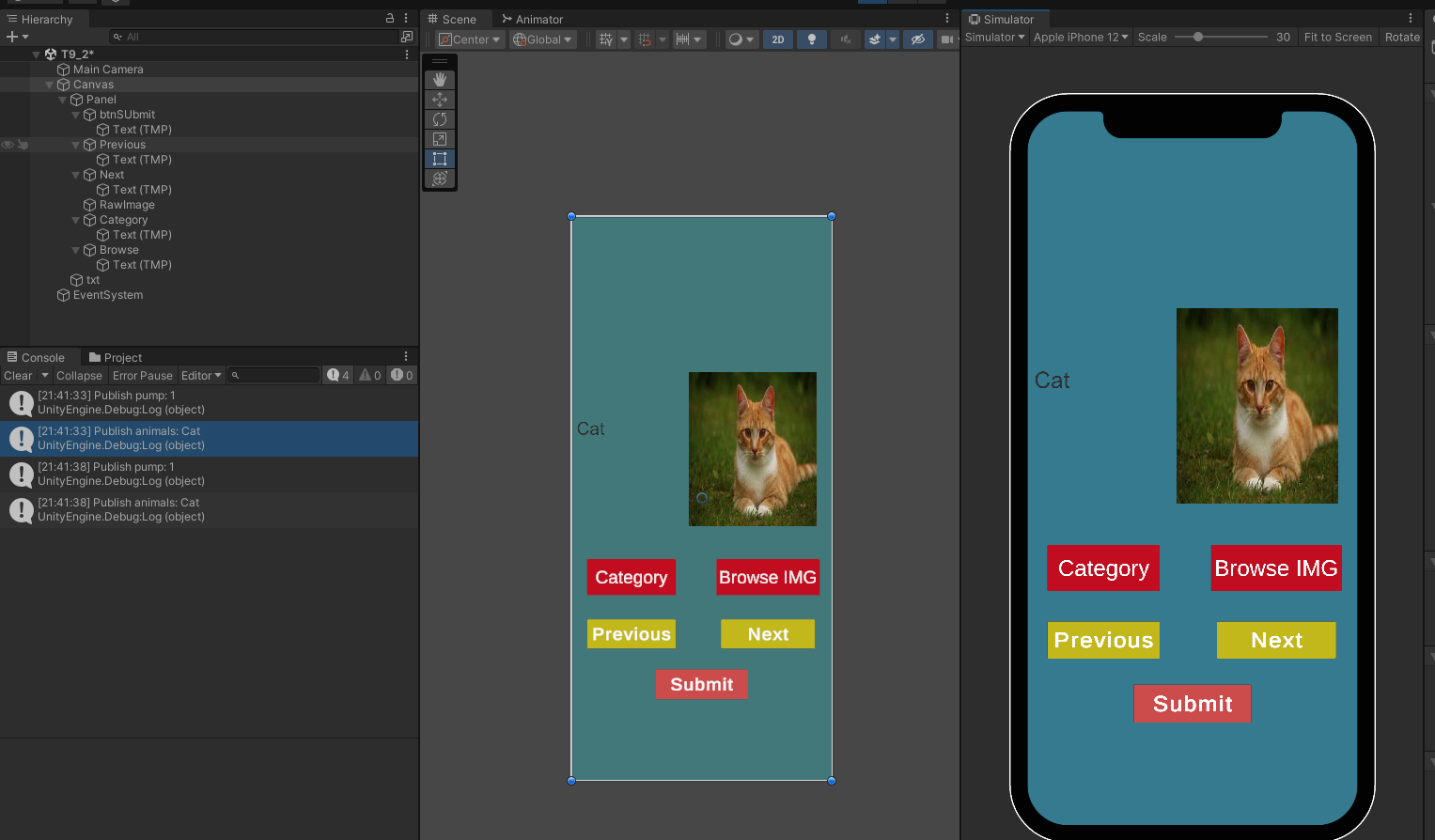
Nhận dạng Dog



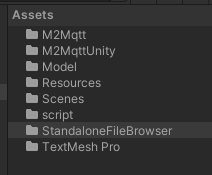
Nút Duyệt hình ảnh: chọn một hình ảnh từ thư mục trên điện thoại di động

Nút danh mục: trả về tên của đối tượng trong hình ảnh.

Tạo thêm 2 button Category, browse



Sau đó tải thư viện StandaloneFileBrowser trên git về và import vào assest



Sau đó code xử lý sự kiện 2 button

using System.Collections;

using System.Collections.Generic;

using System.Text;

using Unity.Barracuda;

using UnityEngine;

using UnityEngine.UI;

using uPLibrary.Networking.M2Mqtt.Messages;

using uPLibrary.Networking.M2Mqtt;

using Unity.VisualScripting;

using SFB;

public class Category\_browse : MonoBehaviour

{

public Button button;

public Button selectImageButton;

public Text txtCount;

public RawImage imgFace;

public NNModel modelAsset;

private MqttClient client;

private string brokerAddress = "io.adafruit.com";

private string username = "Nguyen\_Tham";

private string key = "aio\_XhAG272Co1QkzWifhdnfZZQkm02H";

private Model m\_RuntimeModel;

private IWorker m\_Worker;

private int count, channels;

private string[] feeds = { "pump", "animals" };

private string[] label = { "Cat", "Chicken", "Dog", "Krixi", "Mucca" };

private Queue<System.Action> actionQueue = new Queue<System.Action>();

void Awake()

{

m\_RuntimeModel = ModelLoader.Load(modelAsset);

m\_Worker = WorkerFactory.CreateWorker(WorkerFactory.Type.CSharp, m\_RuntimeModel);

channels = 3; // Số kênh màu

}

void Start()

{

count = 0;

client = new MqttClient(brokerAddress);

client.MqttMsgPublishReceived += OnMessageReceived;

client.Connect(username, username, key);

client.Subscribe(new string[] { $"{username}/feeds/animals" }, new byte[] { MqttMsgBase.QOS\_LEVEL\_AT\_LEAST\_ONCE });

button.onClick.AddListener(OnButtonClick);

selectImageButton.onClick.AddListener(OnSelectImageButtonClick);

}

void OnButtonClick()

{

count = count == 0 ? 1 : 0;

txtCount.text = count.ToString();

SendMQTTMessage("pump", count.ToString());

string predictedLabel = PredictImage();

txtCount.text = predictedLabel;

SendMQTTMessage("animals", predictedLabel);

}

void OnSelectImageButtonClick()

{

var extensions = new[] {

new ExtensionFilter("Image Files", "png", "jpg", "jpeg"),

new ExtensionFilter("All Files", "\*" ),

};

string[] paths = StandaloneFileBrowser.OpenFilePanel("Chọn ảnh", "", extensions, false);

if (paths.Length > 0 && !string.IsNullOrEmpty(paths[0]))

{

StartCoroutine(LoadImage(paths[0]));

}

}

IEnumerator LoadImage(string path)

{

using (WWW www = new WWW("file://" + path))

{

yield return www;

imgFace.texture = www.texture;

}

}

string PredictImage()

{

if (imgFace.texture == null)

{

Debug.LogWarning("Không có ảnh để dự đoán.");

return "No Image";

}

Tensor input = new Tensor(imgFace.texture, channels);

m\_Worker.Execute(input);

Tensor output = m\_Worker.PeekOutput();

int[] maxIndex = output.ArgMax();

return label[maxIndex[0]];

}

void SendMQTTMessage(string topic, string value)

{

string formattedTopic = $"{username}/feeds/{topic}";

client.Publish(formattedTopic, Encoding.UTF8.GetBytes(value), MqttMsgBase.QOS\_LEVEL\_AT\_LEAST\_ONCE, false);

Debug.Log($"Publish {topic}: {value}");

}

void OnMessageReceived(object sender, MqttMsgPublishEventArgs e)

{

string topic = e.Topic;

string message = Encoding.UTF8.GetString(e.Message);

if (topic.Contains("pump"))

{

actionQueue.Enqueue(() => UpdateStatusText(message));

}

}

void UpdateStatusText(string text)

{

txtCount.text = text;

}

void Update()

{

while (actionQueue.Count > 0)

{

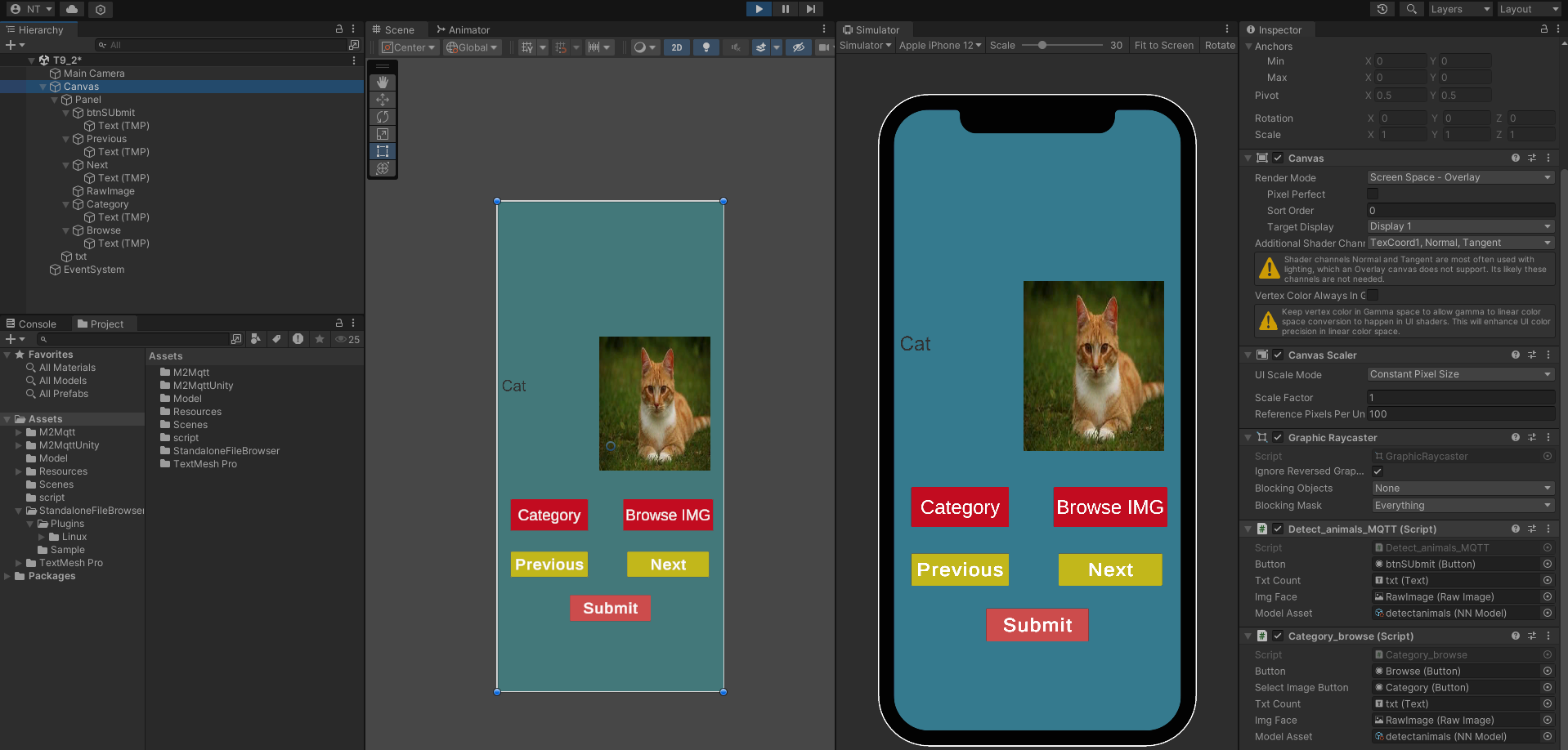
actionQueue.Dequeue().Invoke();

}

}

}

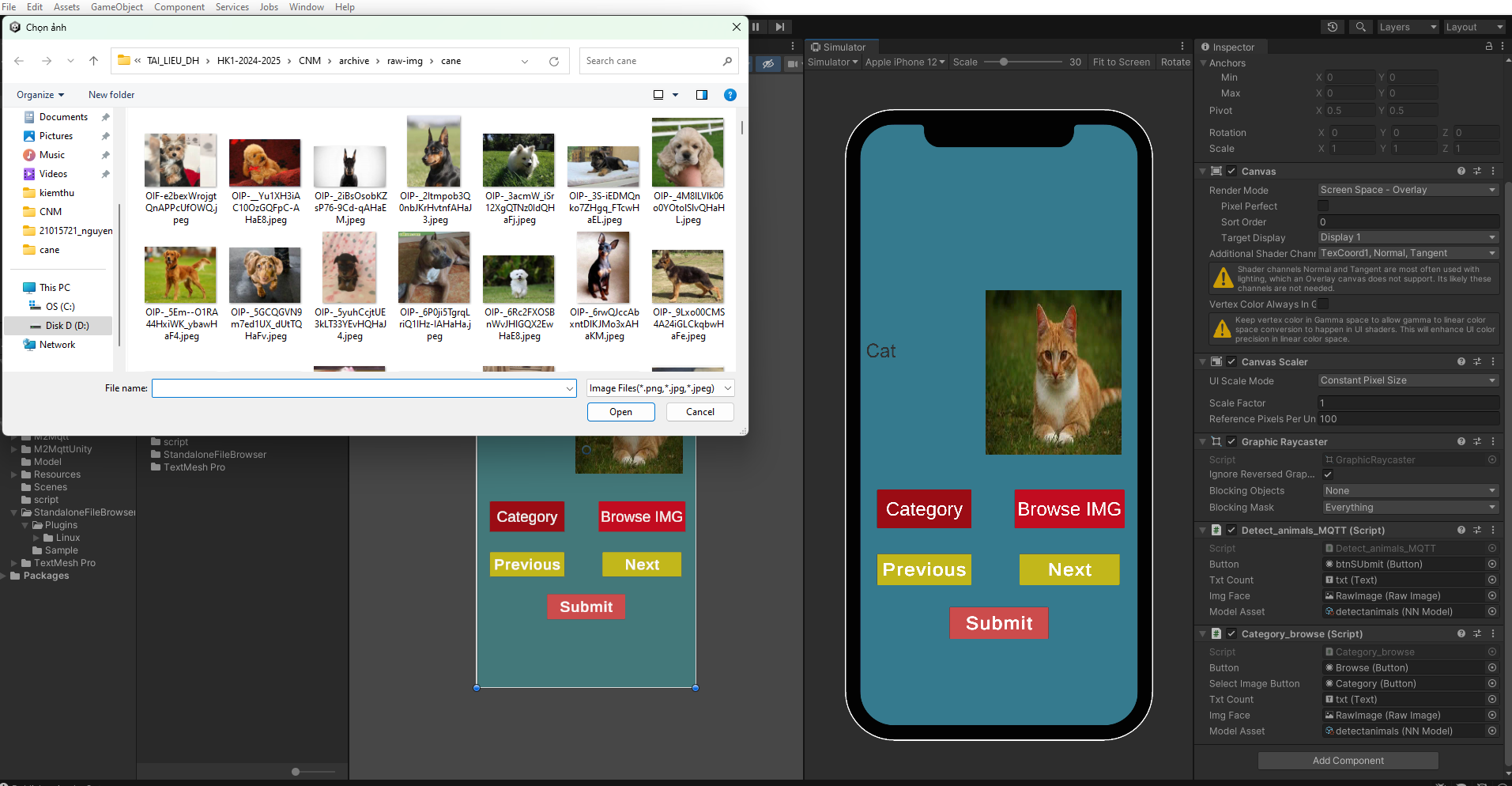
Gắn file script vào canva



Setting các thuộc tính phù hợp

Kết quả

bấm button Browse để chọn ảnh



Sau đó bấm button category để phân loại

