CarControl_Evo

Danna Alejandra Sanchez Monsalve y Juan Pablo Vargas Jimenez

Sistema IoT para el control de un carro 2WD basado en ESP32, gestionado desde una aplicación móvil Android vía MQTT seguro (TLS) sobre AWS IoT Core.

Integración de Tecnologías Clave

Firmware ESP32

Controla el carro 2WD con sensor ultrasónico y luces.

Backend AWS IoT Core

Broker MQTT seguro con TLS para mensajería.

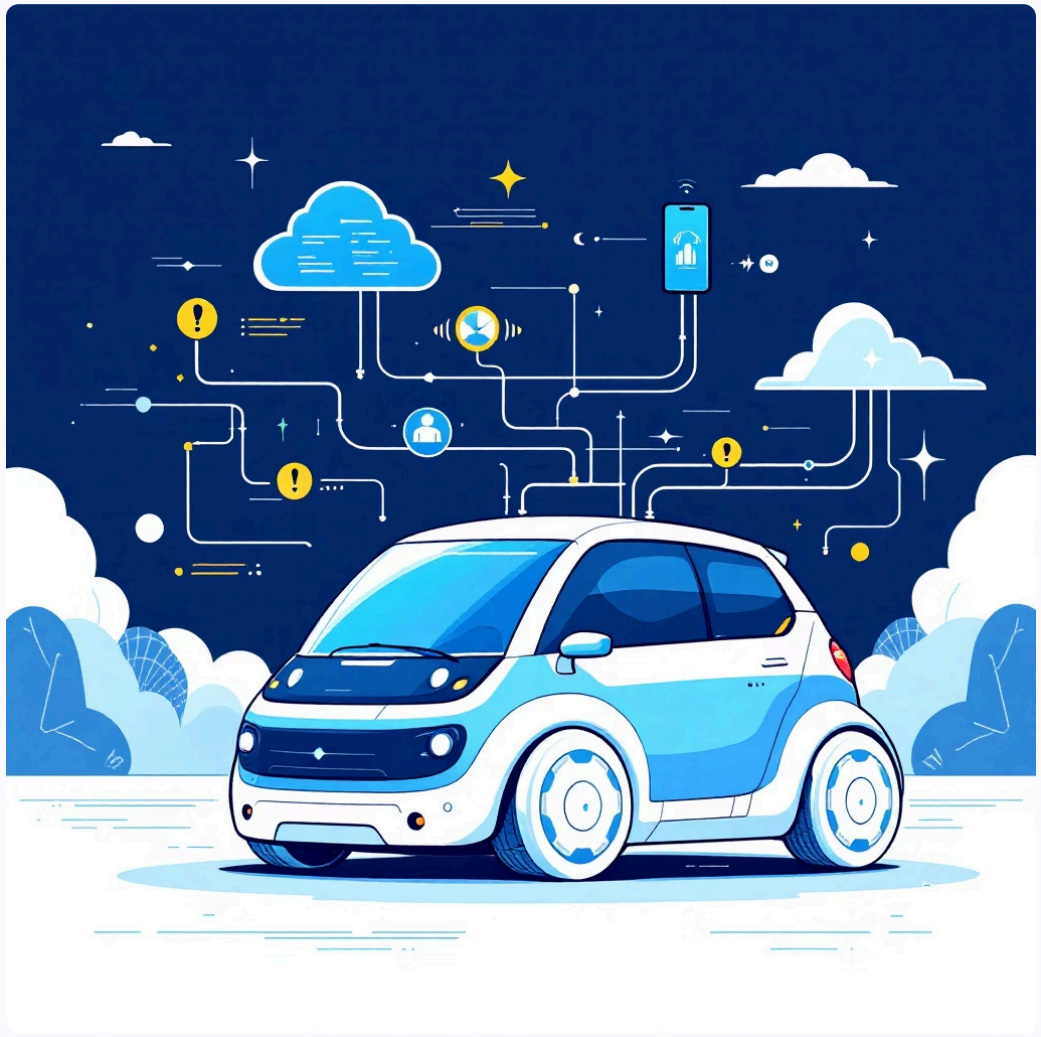
Aplicación Android

Control remoto y visualización de telemetría en Kotlin.

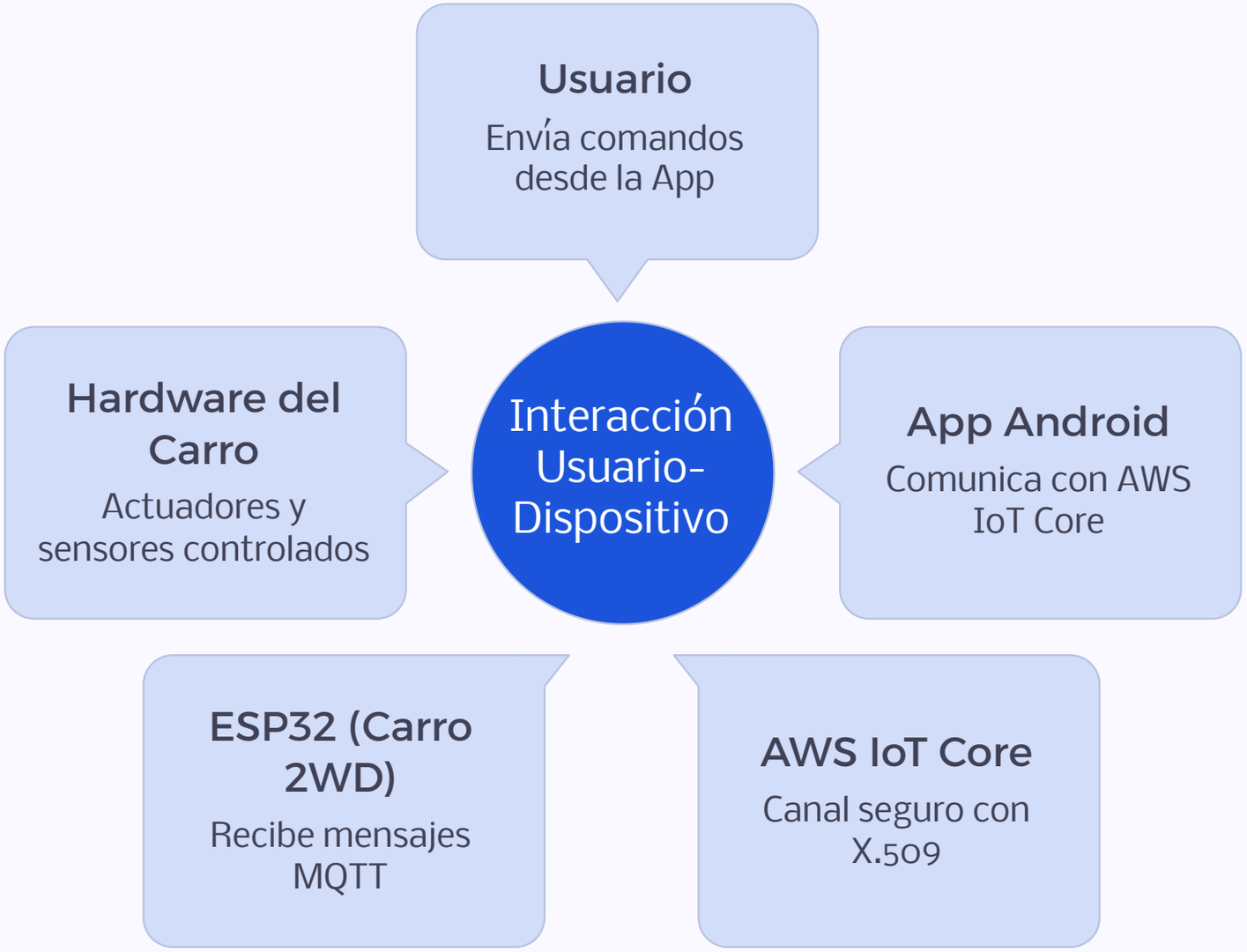
Descripción General del Proyecto

CarControl_Evo es un proyecto académico de IoT y robótica. Un carro 2WD con ESP32 se conecta a AWS IoT Core usando MQTT sobre TLS para publicar telemetría y recibir comandos.

La aplicación Android permite enviar comandos de movimiento, encender/apagar luces, y visualizar la distancia del sensor ultrasónico y el estado del carro en tiempo real.



Arquitectura de Alto Nivel



Todo el flujo de datos pasa por AWS IoT Core, utilizando certificados X.509 para asegurar el canal de comunicación.

Características Principales

Control Remoto Android

Botones táctiles para movimiento (adelante, atrás, izquierda, derecha, stop) y luces (ON/OFF).

Telemetría en Tiempo Real

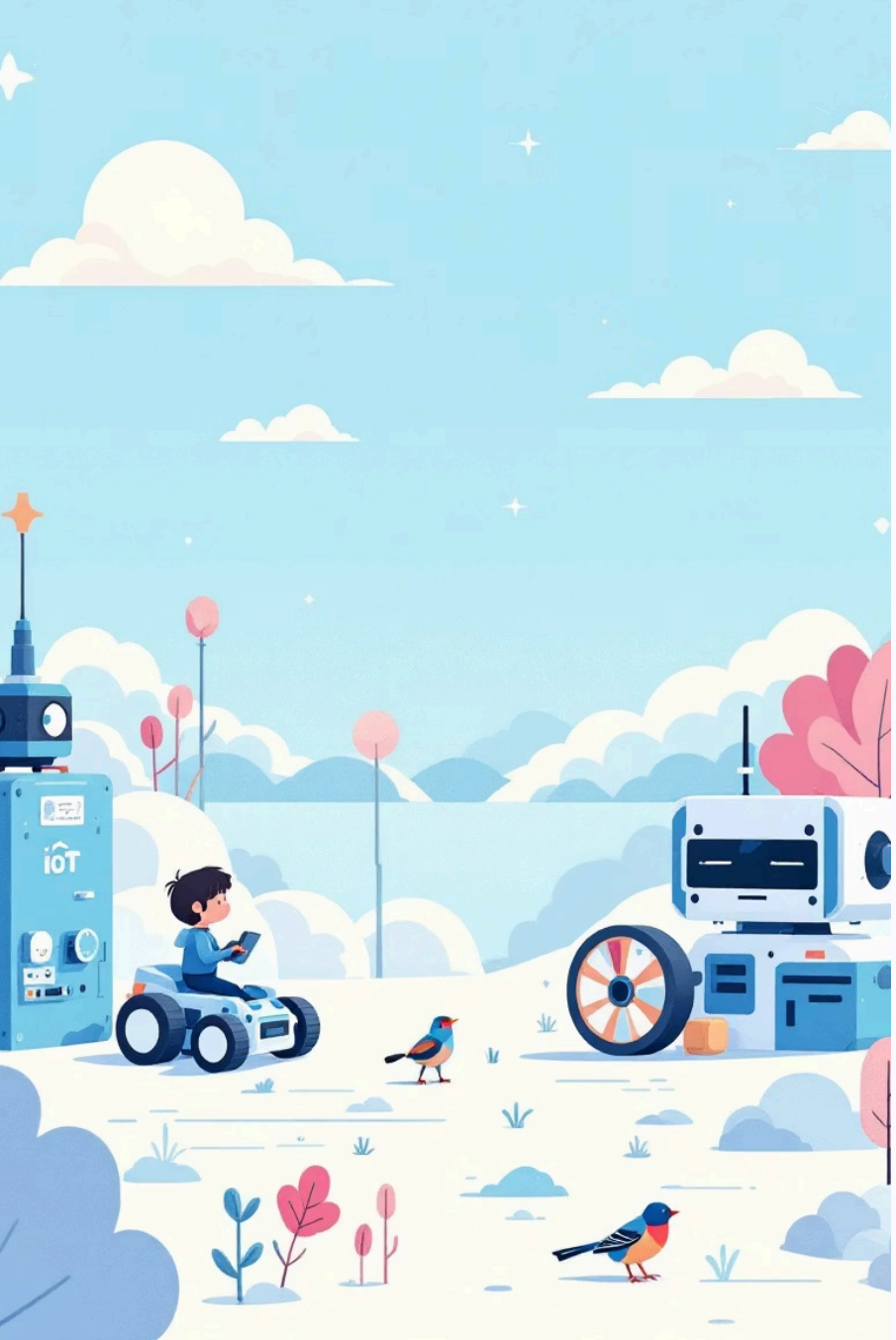
Lectura de distancia (cm) por MQTT y visualización del estado del carro.

Comunicación Segura

Conexión con AWS IoT usando TLS y certificados X.509.

Arquitectura Limpia

Estructura modular en Android (presentation / domain / data) con corrutinas y StateFlow.



Requisitos de Hardware y Software

Hardware

- 1x ESP32 (ESP-WROOM-32)
- 1x Chasis 2WD con motores y ruedas
- 1x Puente H (L298N)
- 1x Sensor ultrasónico (HC-SR04)
- LEDs, baterías, switch, cableado

Software

- PlatformIO + VSCode (firmware ESP32)
- Android Studio (app móvil Kotlin)
- Cuenta AWS con acceso a AWS IoT Core
- Certificados X.509

Configuración de AWS IoT Core

01

Crear "Thing"

Registrar el ESP32 en AWS IoT.

02

Generar Certificados

Crear certificados X.509 (dispositivo, clave privada, CA de Amazon).

03

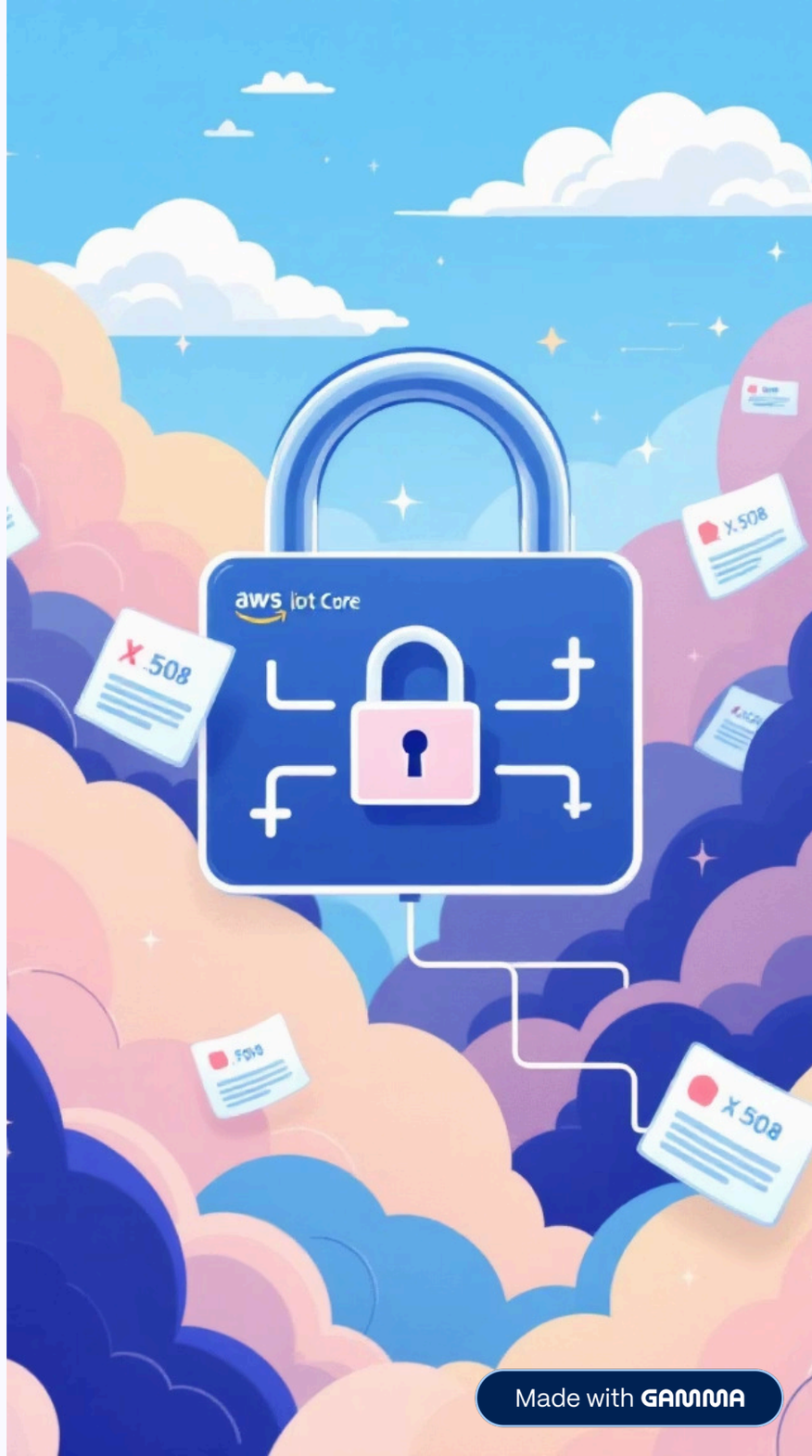
Adjuntar Política IoT

Definir permisos para conexión, publicación y suscripción a topics.

04

Descargar y Configurar

Copiar certificados al proyecto ESP32 y configurar la app Android con el endpoint MQTT.



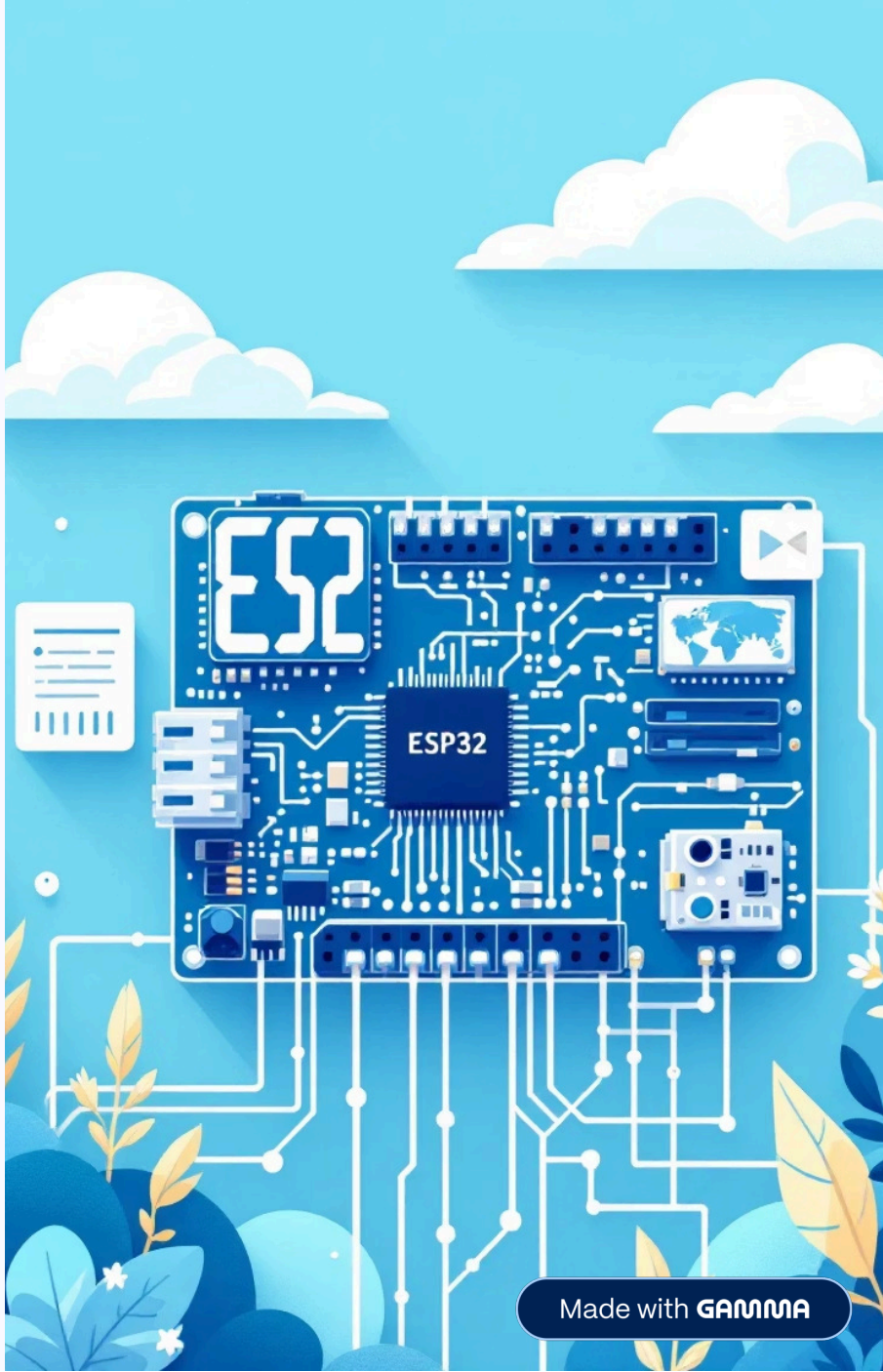
Firmware del ESP32: Configuración y Librerías

Variables del Preprocesador (config.h)

- Pines de motores, luces y auxiliares
- Configuración PWM para motores
- WiFi / Access Point
- Configuración MQTT / AWS IoT Core
- Servo de base rotatoria
- Configuración del sensor ultrasónico

Librerías Utilizadas

- PubSubClient (MQTT)
- ArduinoJson (JSON)
- WiFi (ESP32 core)
- WiFiClientSecure (TLS)
- Arduino Core para ESP32
- Random / esp_random



Aplicación Android: Estructura y Funcionalidad

Librerías Clave

Eclipse Paho MQTT, SslSocketFactory, Kotlin Coroutines/Flow, Jetpack ViewModel.

Paquete Principal

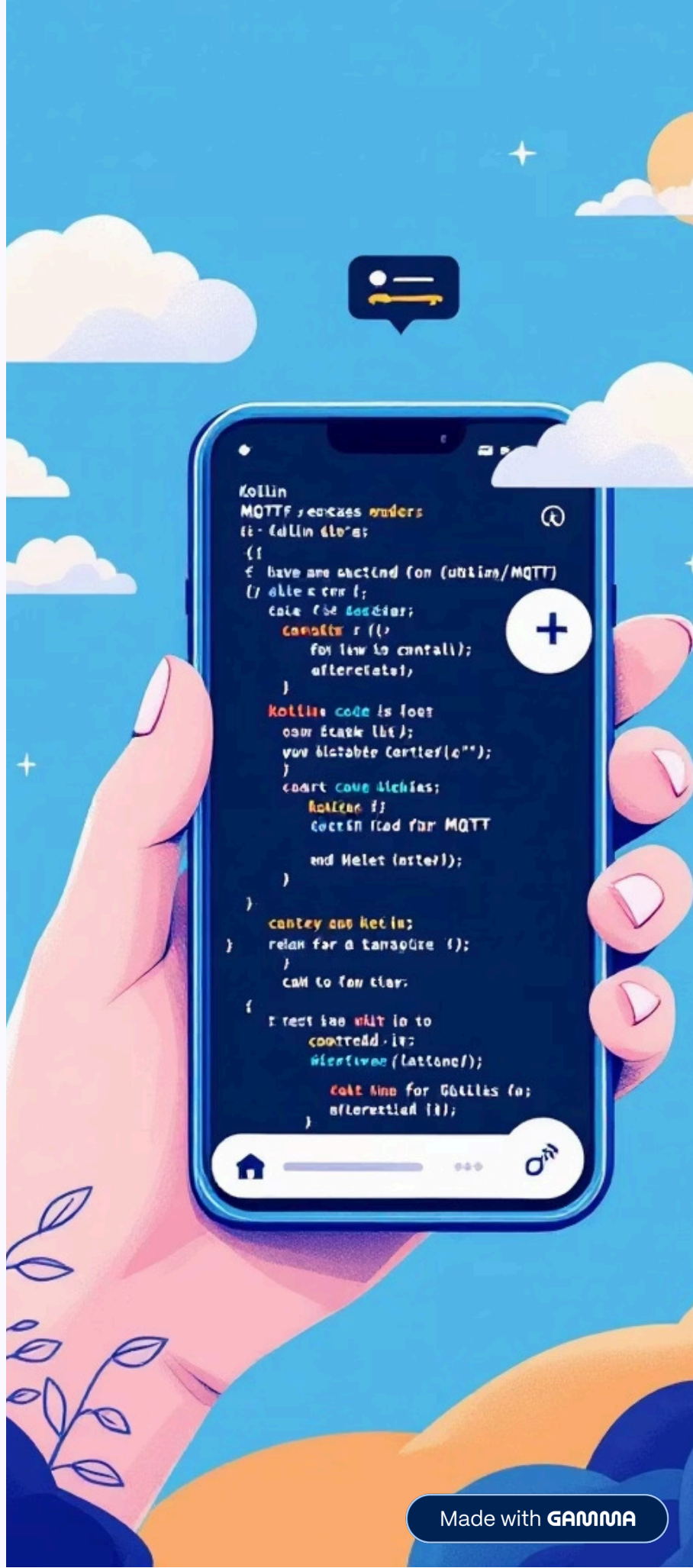
`com.jpvj.controlcarrito` con clases como `MqttManager`, `CarRepositoryImpl`.

Layout Principal

`activity_main.xml` define TextViews para estado/distancia y botones de control.

Topics MQTT y Mensajes

`carro/instrucciones` (String) y `carro/telemetria/distancia` (JSON).



Limitaciones y Futuras Mejoras

Limitaciones Actuales

- Dependencia total de Internet.
- Latencia del MQTT a través de AWS.
- Certificados embebidos en la app.
- Consumo energético del ESP32.
- Sensor ultrasónico limitado.
- No hay streaming de video.
- Sin autenticación de usuario.

Roadmap de Mejoras

- Añadir ESP32-CAM para video.
- Servidor web para control local.
- Dashboard web adicional.
- Detección de obstáculos automática.
- Guardar logs en DynamoDB.
- Internacionalización de la app (ES/EN).