ParkPilot: Autonomous Obstacle-Avoidance & Self-Parking RC Car with Cloud Telemetry

EEC172 Final Project Proposal **Team:** Aktan Azat, Gezheng Kang **Date:** February 2026

1. Description

ParkPilot is an RC car controlled by a CC3200 that avoids obstacles, parallel-parks itself, and sends telemetry to AWS. An IR remote selects between manual drive, autonomous obstacle avoidance, and self-parking. Four ultrasonic sensors provide proximity data rendered as a radar map on a 128x128 OLED, while an Arduino co-processor handles sensor reads and motor actuation over UART.

AWS IoT is used for telemetry and parking guidance. The device shadow stores sensor distances, mode, and speed. An IoT Rule triggers a Lambda function that generates parking guidance stored in S3. A separate SNS rule emails collision alerts when the BMA222 accelerometer detects impact.

Existing hobby autonomous cars (Donkey Car, AWS DeepRacer) rely on cameras and ML on powerful compute. ParkPilot uses only ultrasonic ranging on a constrained MCU with classical control algorithms and cloud-assisted guidance.

2. Design

2.1 Functional Specification

The system operates in four modes selected by IR remote, with collision detection running in parallel across all active modes.

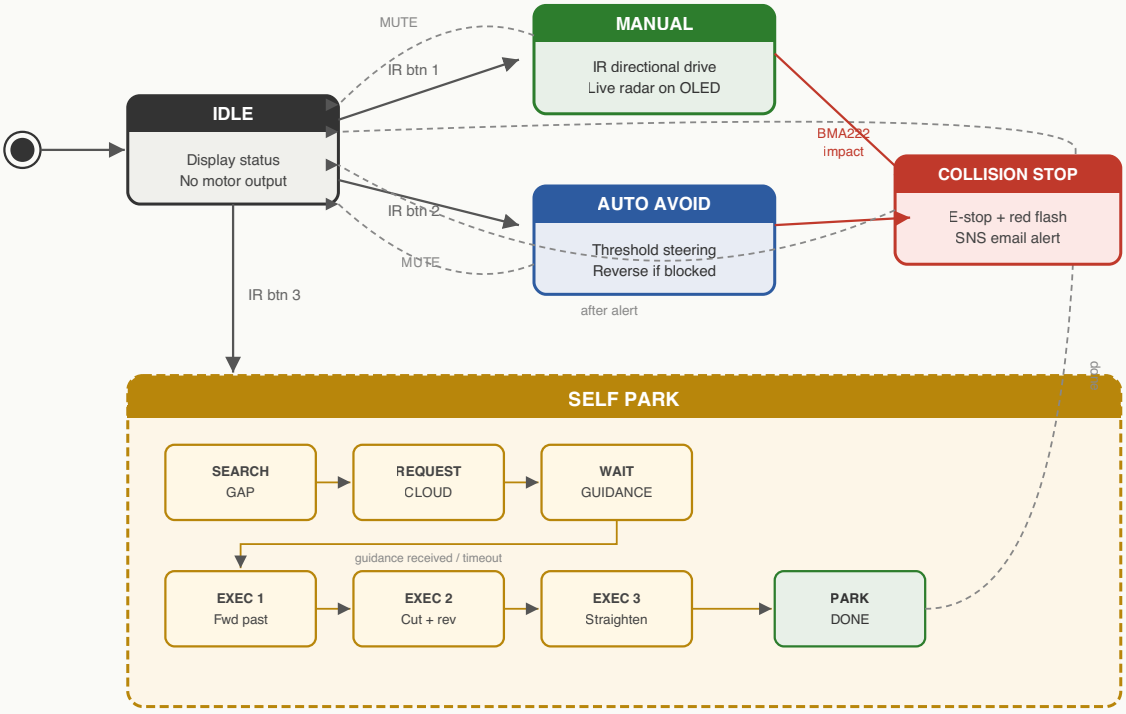

Figure 1: State Machine Diagram

IDLE (boot / STOP): Car stationary. OLED shows WiFi status, battery, mode prompt. **MANUAL** (IR btn 1): IR remote directional control with speed keys and live radar. **AUTO AVOID** (IR btn 2): Forward drive with threshold steering; reverse if all blocked. **SELF PARK** (IR btn 3): Wall-follow gap detection, cloud guidance request via Lambda/S3, then multi-stage parallel parking maneuver with local fallback. **COLLISION STOP**: BMA222 spike triggers e-stop, red flash, SNS email; returns to IDLE.

2.2 System Architecture

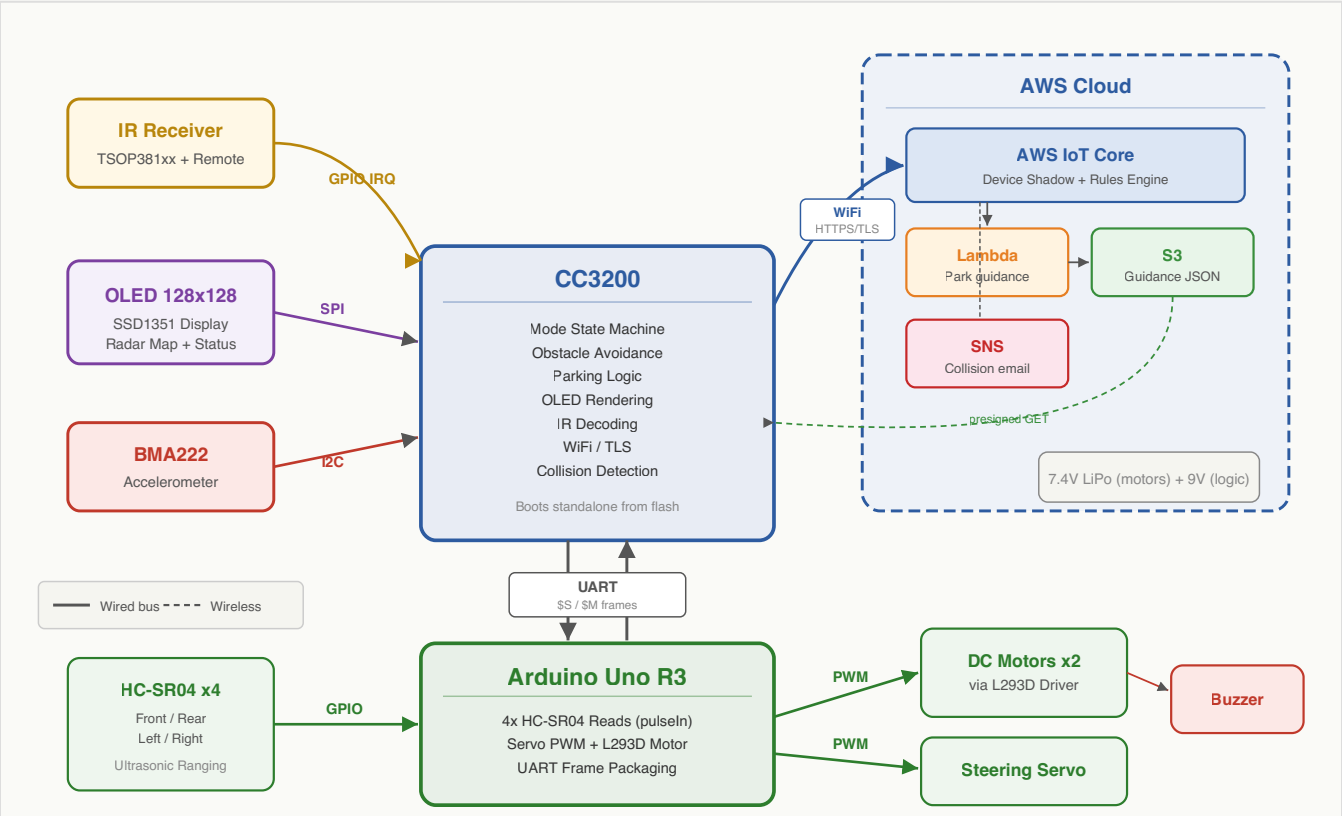

Figure 2: System Architecture Block Diagram

CC3200 (main MCU): Runs mode management, obstacle avoidance, parking state machine, SPI OLED rendering, I2C BMA222 collision monitoring, WiFi/HTTPS telemetry to AWS IoT, and UART command/sensor exchange with Arduino.

Arduino (co-processor): Reads 4x HC-SR04 ultrasonics at ~20 Hz, packages UART sensor frames (`$S,<front>,<rear>,<left>,<right>\n`), receives motor commands (`$M,<speed_pct>,<steer_angle>\n`), and generates PWM for servo and L293D.

Protocols: SPI (OLED), I2C (BMA222), UART (Arduino), GPIO IRQ (IR receiver), HTTPS/REST (AWS). **Sensors:** HC-SR04 x4 (proximity), TSOP381xx (IR remote), BMA222 (collision).

3. Implementation Goals

Minimal

- Car assembly with motors, battery, Arduino
- Ultrasonic UART transmission
- CC3200 UART parsing + OLED display
- IR mode switching + manual drive
- Autonomous obstacle avoidance
- AWS shadow telemetry
- Standalone boot from flash

Target

- Real-time top-down OLED radar map
- BMA222 collision-triggered SNS alert
- All three modes switchable by IR
- Self-parking with Lambda guidance
- Two-minute live demo sequence
- Clean wiring and packaging

Stretch

- Lambda parking arc trajectory + S3 image
- OLED overlay of fetched guidance image
- Speed-proportional avoidance
- Buffered telemetry replay after reconnect

4. Bill of Materials

ITEM	QTY	SOURCE	COST
CC3200 LaunchPad	1	Lab kit	$0
Adafruit OLED 1.5" (SSD1351)	1	Lab kit	$0
AT&T S10-S3 IR Remote	1	Lab kit	$0
TSOP381xx IR Receiver	1	Lab kit	$0
100 Ohm Resistor	1	Lab kit	$0
100 uF Capacitor	1	Lab kit	$0
Arduino Uno R3	1	Arduino kit	$0
HC-SR04 Ultrasonic Sensor	1	Arduino kit	$0
HC-SR04 Ultrasonic Sensor	3	Amazon	~$6
L293D Motor Driver IC	1	Arduino kit	$0
Servo Motor	1	Arduino kit	$0
Active Buzzer	1	Arduino kit	$0
830-point Breadboard	1	Arduino kit	$0
9V Battery + Snap Connector	1	Arduino kit	$0
RC Car Chassis + DC Motors	1	Amazon	~$12
7.4V Li-Po Battery (drive power)	1	Amazon	~$8
Jumper wires (M-M, M-F)	~30	Arduino kit	$0
Zip ties + double-sided tape	1 pack	Hardware store	~$3
Total			**~$29**