Parki 2025

1. General

**🎯 Objective**

* Continuously monitor distance using the VL53L1CB (low power).
* If an object (vehicle) is detected within a certain range (e.g. < 1.5 meters):
  + Wake up camera
  + Take a picture
  + Optionally send it or store it

**🧠 System Logic (Simplified Flow)**

Loop:

├─ Read distance from ToF sensor (VL53L1CB)

├─ If distance < 1.5 m:

│ ├─ Capture image with camera

│ ├─ Store (or send) photo

│ └─ Optionally delay for a few seconds to avoid repeated shots

└─ Else:

└─ Sleep or wait before checking again

**🧰 Hardware Needed**

* ✅ Arduino Nicla Vision (built-in ToF + camera)
* 🔋 Li-Po Battery (3.7V 500–1000mAh) for portable setup
* 💾 Optional: SD card via breakout if you want to store images locally

**Tech Specs**

So yes — the Nicla Vision **does include a ToF sensor (VL53L1CB)**, and it:

* Acts like a **mini-LiDAR**
* Has a **FoV of 27°**
* Works well for parking lot occupancy detection if vehicle is **within 0.05–4 m**
* Works in **dark, indoor, outdoor, and bright light**

**🔌 Power Saving Options**

* Keep the **ToF sensor running**, it uses **< 20 mA**
* Only activate **camera + Wi-Fi** when needed
* Add LowPower.sleep() if you’re running on battery
* Use **LoRa or BLE** later to reduce transmission energy

**📦 Next Steps**

1. **Test the ToF triggering** using serial output
2. **Integrate image capture**
3. Add storage (SD card) or **Wi-Fi image sending**
4. Add power optimization and battery monitoring
5. **Webserver structure**

- Create table for Gate 1, Lot 1

CREATE TABLE Park1G1Lot1 (

    id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

    timeStamp TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

    status VARCHAR(50),           -- General status

    statusWeight INT,               -- General status weight

    statusS01 VARCHAR(50),         -- Sensor 01 status

    statusS01Weight INT,            -- Sensor 01 weight/priority

    statusS02 VARCHAR(50),         -- Sensor 02 status

    statusS02Weight INT,            -- Sensor 02 weight/priority

    statusS03 VARCHAR(50),         -- Sensor 03 status

    statusS03Weight INT,            -- Sensor 03 weight/priority

img1\_path VARCHAR(255),

img2\_path VARCHAR(255),

img3\_path VARCHAR(255);

    PRIMARY KEY (id)

);

**Typical architucture**

[Nicla Vision]

     ↓ HTTPS (Supabase REST API)

[Supabase Database + Storage]

     ↓ Web or Mobile App (optional)

[User Interface to see data + photos]

<https://supabase.com/dashboard/new/qzdduwnsdcqmkxxwptgv?projectName=ehabquran@yahoo.com%27s%20Project>

Parki Org, Parki Lots Project, DB pwd: Eaaq27276

Project ID:

Keukwkhoetrytwswayyt

URL:

<https://keukwkhoetrytwswayyt.supabase.co>  
  
anon public API keys

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImtldWt3a2hvZXRyeXR3c3dheXl0Iiwicm9sZSI6ImFub24iLCJpYXQiOjE3NTQ3MzcxMDEsImV4cCI6MjA3MDMxMzEwMX0.HVNaaobkqxP0pe\_vWNt\_LwJqh5YzarS\_naqEdHebn00

create table public."parkingLotEventsNV1" (

id bigint generated by default as identity not null,

timestamp timestamp with time zone not null default now(),

"lotId" text null,

status text null,

"statusWeight" bigint null,

"statusS01" text null,

"statusS01Weight" bigint null,

"statusS02" text null,

"statusS02Weight" bigint null,

"statusS03" text null,

"statusS03Weight" bigint null,

constraint parkingLotEventsNV1\_pkey primary key (id)

) TABLESPACE pg\_default;

Photos stores in (bucket name):

parki-photos

github

ghp\_PQCK4Lj4yAv0dkBVKH1ZVndzJG5IYy4UXhJE

ghp\_PQCK4Lj4yAv0dkBVKH1ZVndzJG5IYy4UXhJE

render, db internal url: postgresql://parki\_db\_t2rj\_user:nxiIBVWpFddCR2LFc5J5efkN76AZcKOg@dpg-d2bu2rre5dus738j7q9g-a/parki\_db\_t2rj

1. h
2. Kjk

On Render server, Service Name:  
1. **parki-db**  
2. **parki-server (using sorce code from github:** [ehab / parki-servermain](https://github.com/ehab/parki-server/tree/main))

database name is: parki\_db\_t2rj

table name is: parking\_data

host: [dpg-d2bu2rre5dus738j7q9g-a.oregon-postgres.render.com](http://dpg-d2bu2rre5dus738j7q9g-a.oregon-postgres.render.com)

posrt: 5432

authentification, username: parki\_db\_t2rj\_user

authentification, password: nxiIBVWpFddCR2LFc5J5efkN76AZcKOg

url: jdbc:<postgresql://dpg-d2bu2rre5dus738j7q9g-a.oregon-postgres.render.com:5432/parki_db_t2rj>

psql command: PGPASSWORD=nxiIBVWpFddCR2LFc5J5efkN76AZcKOg psql -h dpg-d2bu2rre5dus738j7q9g-a.oregon-postgres.render.com -U parki\_db\_t2rj\_user parki\_db\_t2rj

i added the folowing to the environment variables:

DATABASE\_URL : postgresql://parki\_db\_t2rj\_user:nxiIBVWpFddCR2LFc5J5efkN76AZcKOg@dpg-d2bu2rre5dus738j7q9g-a/parki\_db\_t2rj

https://parki-server.onrender.com/

curl -X POST https://parki-server.onrender.com/parking \

-H "Content-Type: application/json" \

-d '{"spot\_id": 1, "status": "occupied"}'

**Summary for your workflow:**

1. Modify code files locally
2. git status to see changes
3. git add . (or specific files)
4. git commit -m "meaningful message"
5. git push origin main
6. **How to check your branches**
7. bash
8. *# See all branches (current branch has a \* next to it)*
9. git branch
10. *# See which branch you're currently on*
11. git branch --show-current
12. *# See remote branches too*
13. git branch -a
14. **Basic branch commands**
15. bash
16. *# Create a new branch*
17. git branch feature-login
18. *# Switch to a branch*
19. git checkout feature-login
20. *# or newer syntax:*
21. git switch feature-login
22. *# Create and switch to new branch in one command*
23. git checkout -b new-feature
24. *# or*
25. git switch -c new-feature
26. *# Push to specific branch*
27. git push origin branch-name
28. **Simple workflow example**
29. bash
30. *# You're on main branch*
31. git branch *# shows: \* main*
32. *# Create branch for new feature*
33. git checkout -b add-user-profile
34. *# Make changes, commit them*
35. git add .
36. git commit -m "Add user profile page"
37. *# Push this branch*
38. git push origin add-user-profile
39. *# Later, switch back to main*
40. git checkout main
41. When you do git push origin main, you're telling Git: "push my commits to the remote repository (origin) on the main branch

**DB/Table Name:**

**parkingLotEventsNV1**

id

timestamp

lotId

sensorId

status

statusWeight

statusS01

statusS01Weight

statusS02

statusS02Weight

statusS03

statusS03Weight

snapshot1Url

snapshot2Url

snapshot3Url

snapshotAiUrl

statusAccuracyPercent

statusS01AccuracyPercent

statusS02AccuracyPercent

statusS03AccuracyPercent