Sedemac Hook

Development of a scalable prototype IoT (Internet of Things) system for automobiles

Goal

The project involves construction (either from scratch or by integrating devices available off the shelf) of a system which can acquire CAN (controller area network) data from a SEDEMAC automobile ECU and transmit the same to a server at SEDEMAC (through GSM for example). The idea is to explore the possibilities that *big data* offers for automotive data.

What exactly?

Logging CAN data from vehicles, analyzing the data and presenting the insights from data in such a way that is beneficial to automobile manufacturers

PLAN

1) Convert CAN signals to some things we can understand
2) Send that understood data to a secure place
3) Saving data in a proper format to bridge efficiently between 2) and 4)
4) Presenting data and making it accessible (Data Analytics).

Problem 1

Convert CAN signal to something you understand.

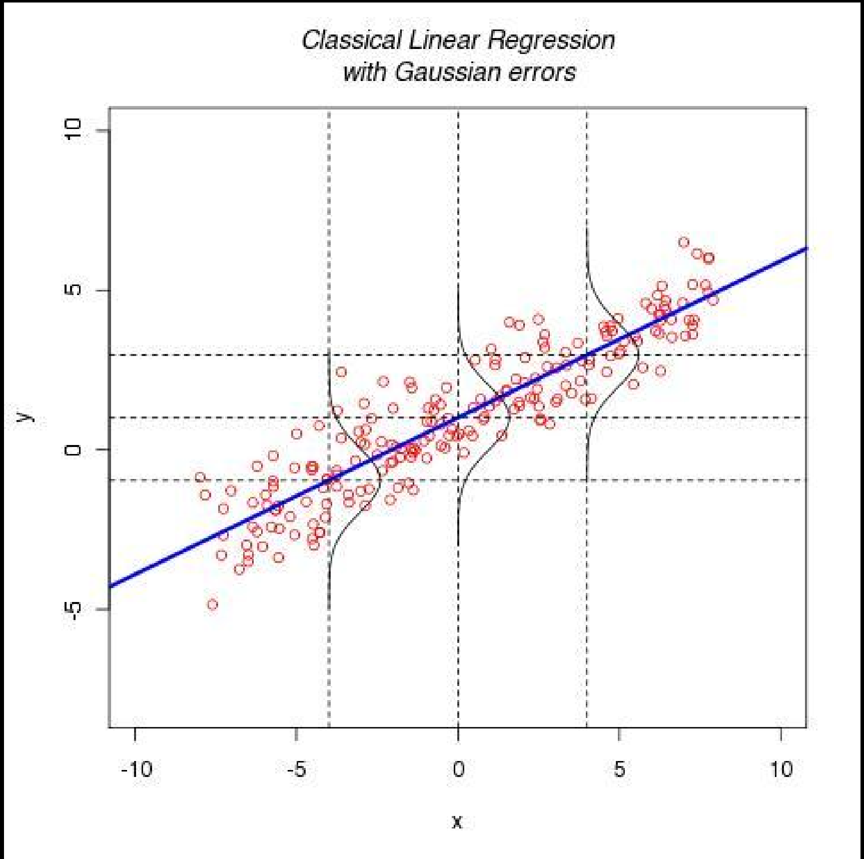
Sol: Connect the two CAN pins to an OBD port. Do some logic level shifts to the output of OBD port to make it Arduino understandable. You understand OBD signal format!

Problem 1.1 : No CAN Signal

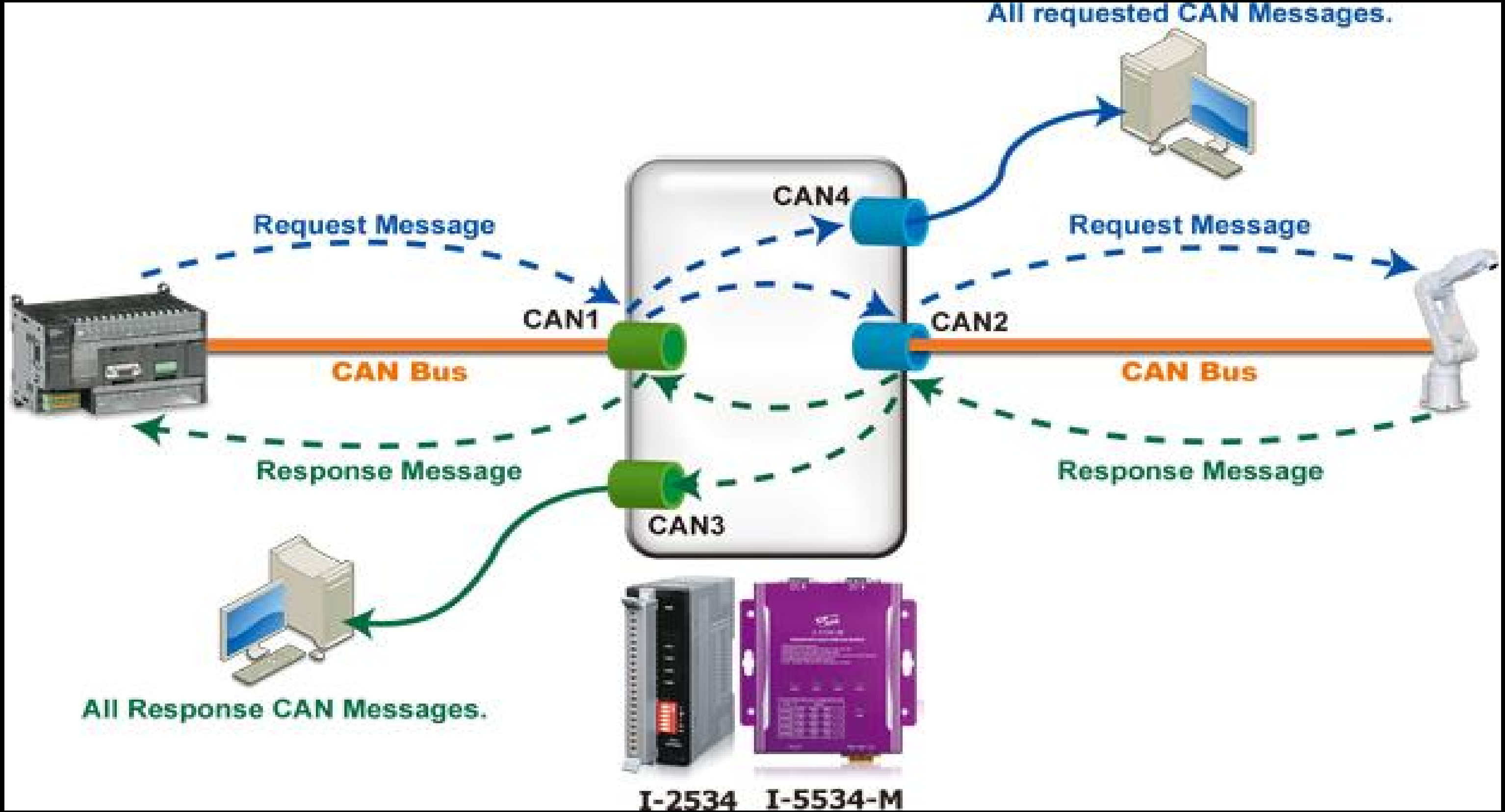
—

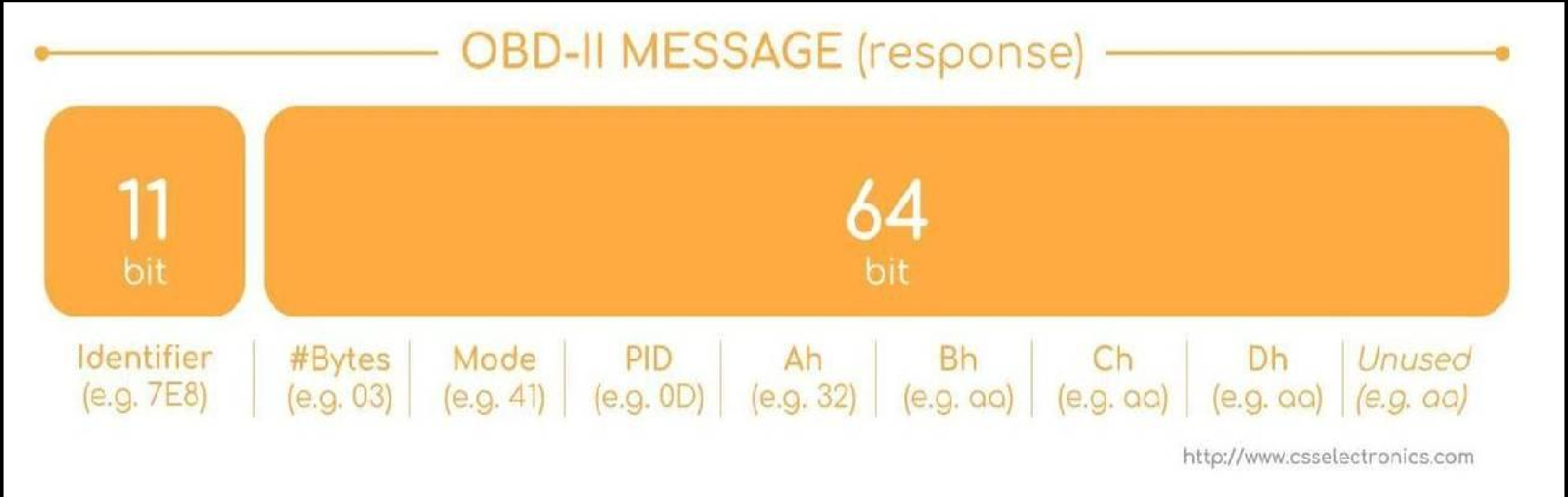
Sol : Emulator

Peak CAN

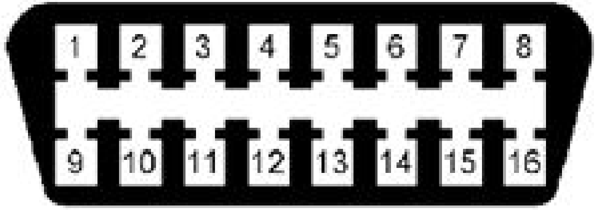


Classical Linear Regression
with Gaussian errors
10
5
0
-5
y
-10
-5
0
5
10
x


All requested CAN Messages.
CAN4
Request Message
Request Message
CAN1
CAN2
CAN Bus
CAN Bus
Response Message
Response Message
CAN3
All Response CAN Messages.
I-2534
I-5534-M



OBD-II MESSAGE (response)
11
bit
64
bit
Identifier
(e.g. 7E8)
#Bytes
(e.g. 03)
Mode
(e.g. 41)
PID
(e.g. 0D)
Ah
(e.g. 32)
Bh
(e.g. aa)
Ch
(e.g. aa)
Dh
(e.g. aa)
Unused
(e.g. aa)
http://www.csselectronics.com



PIN	DESCRIPTION	PIN	DESCRIPTION
1	Vendor Option	9	Vendor Option
2	J1850 Bus +	10	J1850 Bus –
3	Vendor Option	11	Vendor Option
4	Chassis Ground	12	Vendor Option
5	Signal Ground	13	Vendor Option
6	CAN (J-2234) High	14	CAN (J-2234) Low
7	ISO 9141-2 K-Line	15	ISO 9141-2 L-Line
8	Vendor Option	16	Battery Power

OBD-II Connector and Pinout

DTC	Description	Domestic RPO L82
P0101	Mass Air Flow (MAF) System Performance	A
P0102	Mass Air Flow (MAF) Sensor Circuit Low Frequency	A
P0103	Mass Air Flow (MAF) Sensor Circuit High Frequency	A
P0107	Manifold Absolute Pressure (MAP) Sensor Circuit Low Voltage	B
P0108	Manifold Absolute Pressure (MAP) Sensor Circuit High Voltage	B
P0112	Intake Air Temperature (IAT) Sensor Circuit Low Voltage	B
P0113	Intake Air Temperature (IAT) Sensor Circuit High Voltage	B
P0117	Engine Coolant Temperature (ECT) Sensor Circuit Low Voltage	B
P0118	Engine Coolant Temperature (ECT) Sensor Circuit High Voltage	B
P0121	Throttle Position (TP) System Performance	A
P0122	Throttle Position (TP) Sensor Circuit Low Voltage	A
P0123	Throttle Position (TP) Sensor Circuit High Voltage	A
P0125	Engine Coolant Temperature (ECT) Excessive Time to Closed Loop Fuel Control	B
P0131	HO2S Circuit Low Voltage Sensor 1	B
P0132	HO2S Circuit High Voltage Sensor 1	B
P0133	HO2S Slow Response Sensor 1	B
P0134	HO2S Circuit Insufficient Activity Sensor 1	B
P0135	HO2S Heater Circuit Sensor 1	B
P0137	HO2S Circuit Low Voltage Sensor 2	B
P0138	HO2S Circuit High Voltage Sensor 2	B
P0140	HO2S Circuit Insufficient Activity Sensor 2	B
P0141	HO2S Heater Circuit Sensor 2	B
P0171	Fuel Trim System Lean	B
P0172	Fuel Trim System Rich	B
P0201	Injector 1 Control Circuit	B
P0202	Injector 2 Control Circuit	B
P0203	Injector 3 Control Circuit	B
P0204	Injector 4 Control Circuit	B
P0205	Injector 5 Control Circuit	B
P0206	Injector 6 Control Circuit	B
P0230	Fuel Pump Control Circuit	D
P0300	Engine Misfire Detected	B
P0325	Knock Sensor Circuit	D
P0327	Knock Sensor Circuit Bank 1	D
P0336	18X Reference Signal	B
P0341	CMP Sensor Circuit Performance	B
P0401	EGR System Flow Insufficient	A

Problem 2 :

—

Reading & Sending Data.

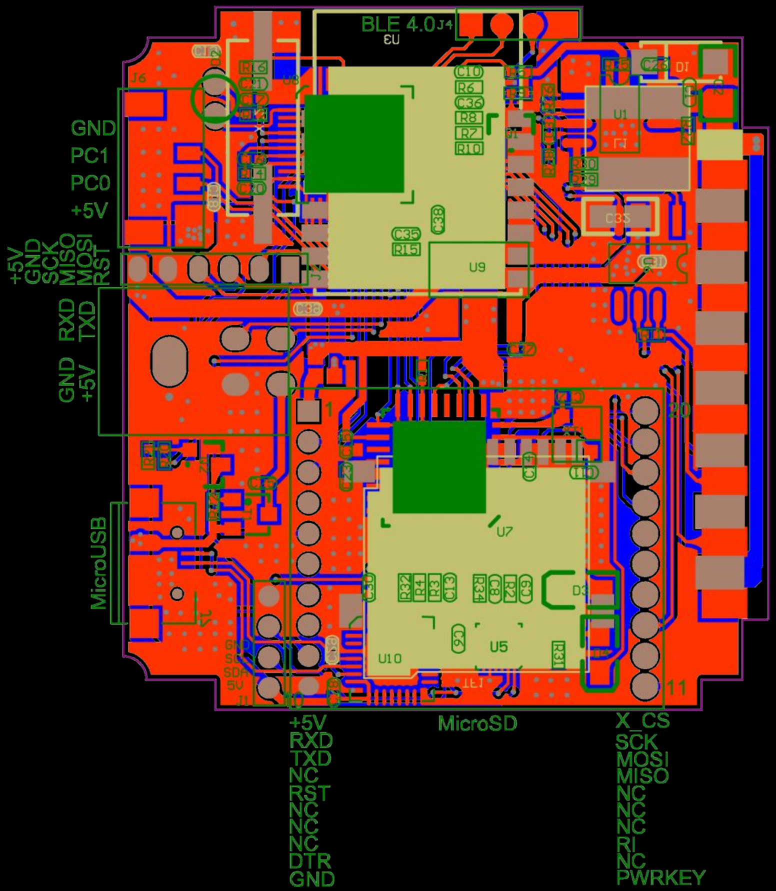
Solved

Features:

1) OBD port + GPS
2) Arduino
3) SIM800A
4) SD card

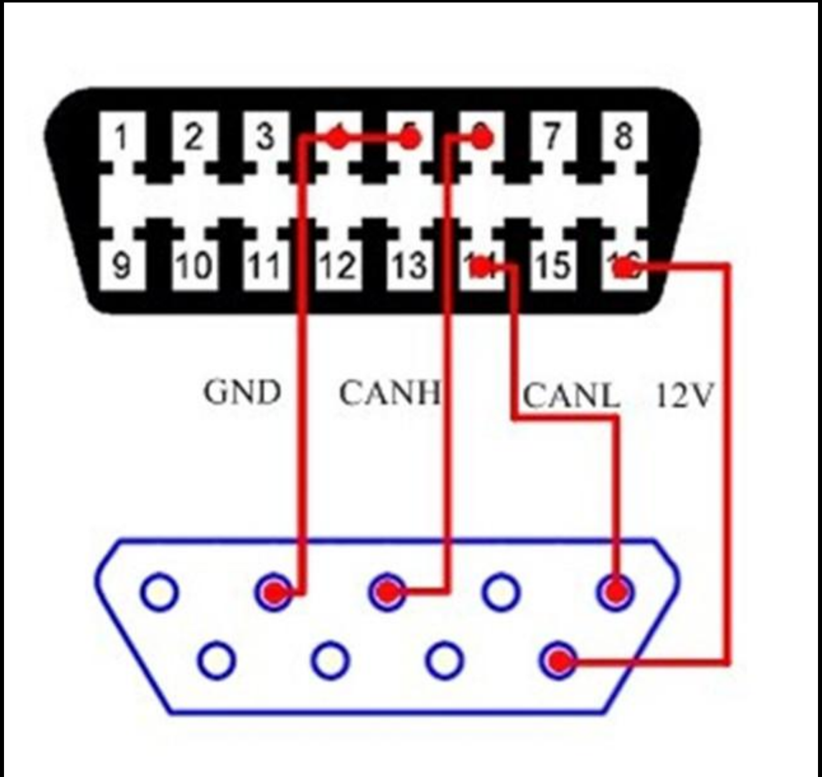
Inside

Circuit

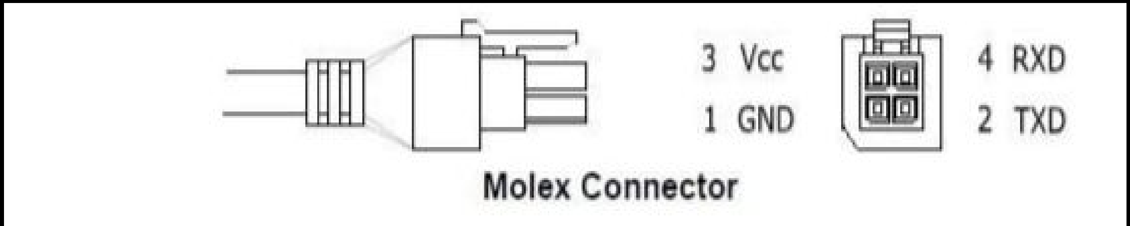

BLE 4.0
GND
PC1
PC0
+5V
+5V
GND
SCK
MISO
MOSI
RST
RXD
TXD
GND
+5V
MicroUSB
MicroSD
+5V
RXD
TXD
NC
RST
NC
NC
NC
DTR
GND
X_CS
SCK
MOSI
MISO
NC
NC
NC
RI
NC
PWRKEY

Connections :

1) Read CAN

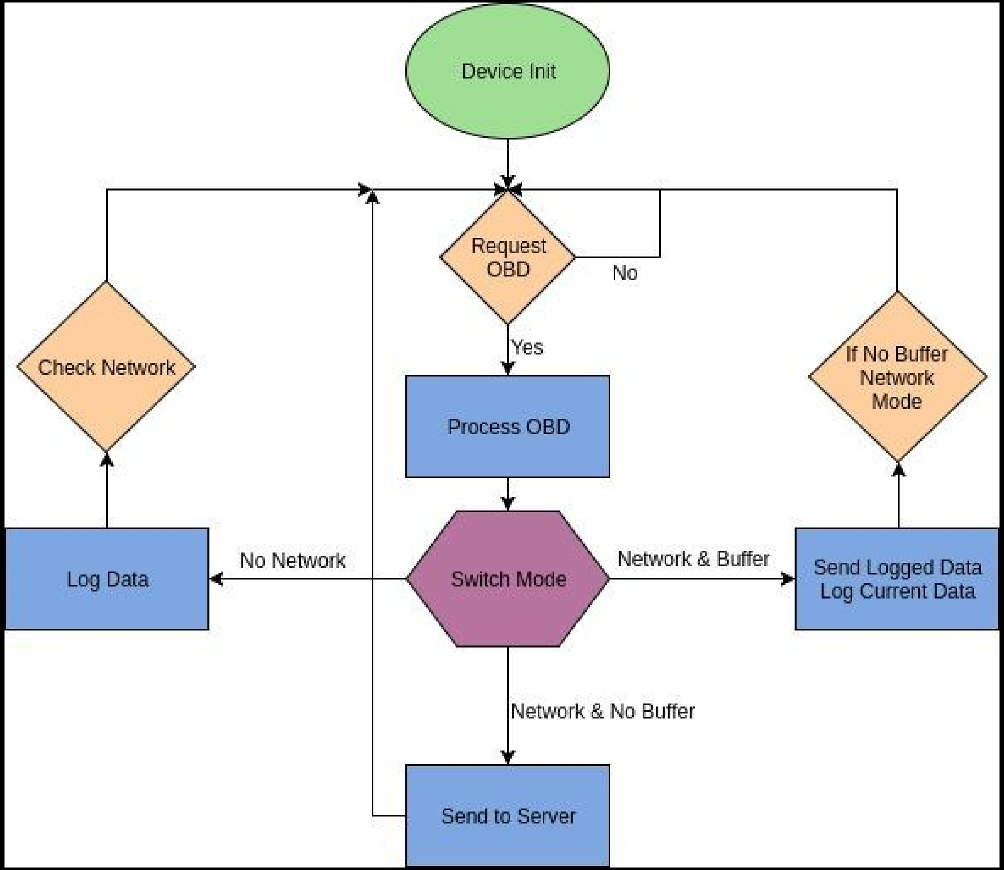


2) GPS



3) SIM & SD

Algo :



Device Init
Request OBD
No
Yes
Process OBD
Switch Mode
No Network
Log Data
Check Network
Network & Buffer
Send Logged Data
Log Current Data
If No Buffer Network Mode
Network & No Buffer
Send to Server

Complications :

—

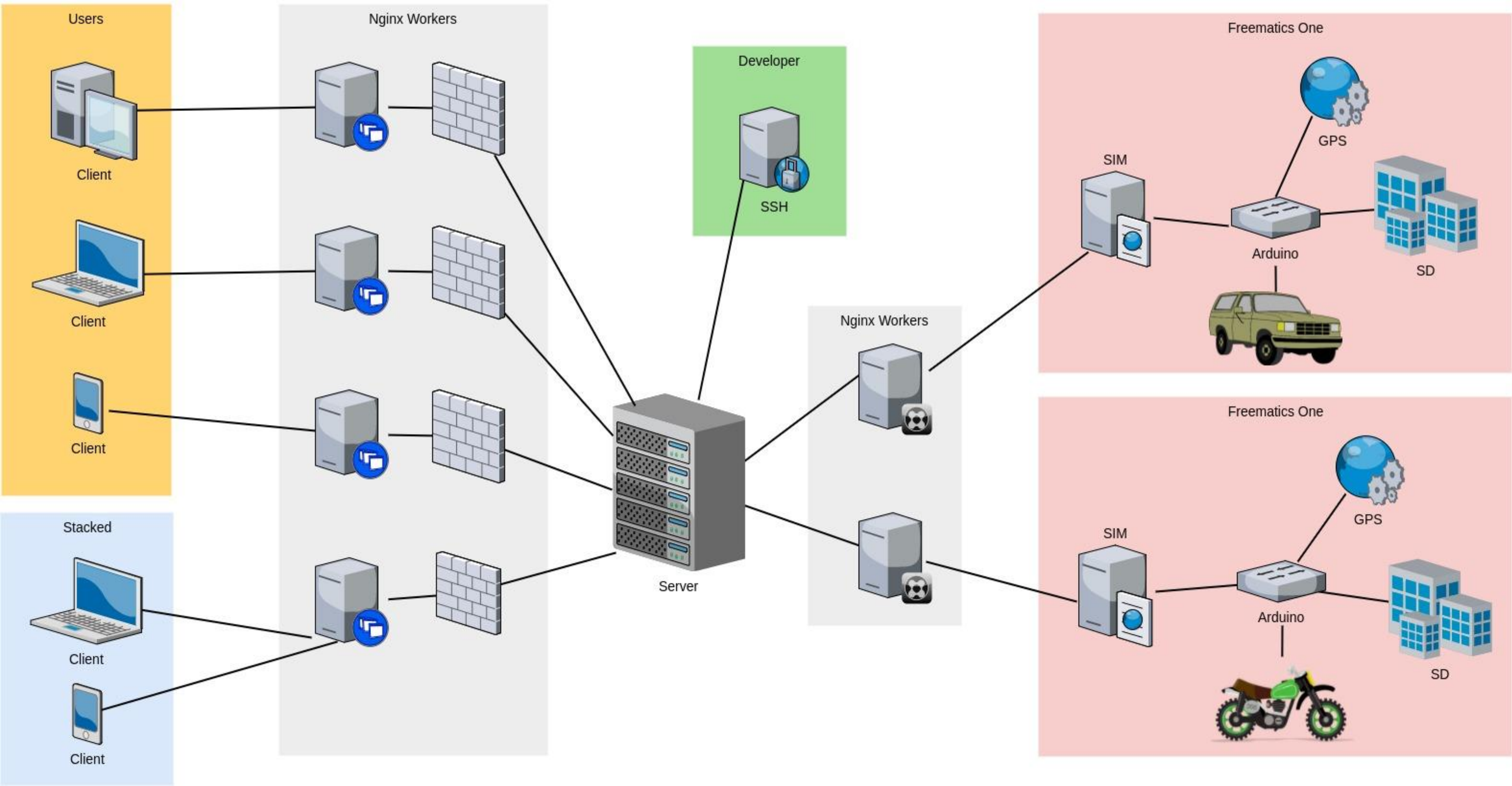
2kB dynamic memory.

SD card takes 512 bytes.

Device crashed.

GPS takes long to init.

Project Architecture



Server (Backend)

Objectives :-

I. Scalable backend to accommodate ever increasing number of users
II. Structured database for easier and quick SQL transactions
III. API endpoints to serve multiple purposes
IV. Easy development of frontend platforms over the backend
    A. Web-Frontend
    B. Mobile App
    C. Customised Scripts for Data Analytics

Server specifications (Backend)

I. Digitalocean cloud server
II. Nginx is used to deploy our Flask-App
III. Server side is developed on Flask (a python based framework)
IV. Database designed in postgreSQL
V. API endpoints developed to serve the required purposes

Database Structure

Aim

1. Easily understandable
2. Fast Query
3. Makes Data Analytics of individual as well as cumulative vehicles efficient

Development so far

1. Separate table for individual vehicles to store raw data directly
2. Separate table to measure overall performance of vehicles
3. All the data transactions are done in JSON format which is easy to understand and access

Objectives Achieved

1. Multiple devices : Nginx server with ability to handle multiple simultaneous requests is used to host our server
2. Fast Query : The database is structured in three layers for easier access and efficient data analytics
3. Data Analytics : API endpoints developed to serve data from database directly in JSON format for easy interpretation and custom scripts
   JSON Response Example: http://139.59.38.17/data/<device_id>/details
4. Python based Flask Framework is completely scalable

Three Layers of Table

1. Primary Table :- Used to directly store raw data coming from device. Every device has it's own primary table named as DeviceID. For example Device1 and Device2

2. Secondary Table :- Used to store trip wise details of a particular vehicle. Will be helpful to analyse trip parameters like average trip distance or average trip speed. Every device will have its own secondary table named as device_derivedID. For example device_derivced1, device_derived2

3. Tertiary Table :- Used to store cumulative record of all the vehicles together. Will make overall analytics of all the vehicles much more simpler and efficient. Each device will have one row for it in this table.

Data as sent by device

	Vehicle_1	Vehicle_2
erpm	2100	3200
engine_load	25	29
Vehicle_speed	32	39
Throttle_position	25	45
Runtime_crank	300	420
Latitude	18.5204	18.7904
Longitude	73.8567	73.9766
Altitude	350	450
data_time	12:10:34	12:10:45
data_date	29/06/17	29/06/17
New_data	1	0

Primary Table

Serial	erpm	engine_load	runtime	throttle_pos	latitude	longitude	altitude	speed	date	time
1	1200	23	460	33	18.2345	27.6789	540	43	29/06/17	13:10:00
2	1260	25	470	36	18.3012	27.7777	530	47	29/06/17	13:10:10
3	1300	29	480	41	18.3113	28.1234	550	49	29/06/17	13:10:20

Format in which Raw data is stored in Primary Table

Secondary Table

Serial	trip_duration	distance	avg_speed	avg_erpm	avg_load	latitude	longitude	altitude	start_time	date	end_time
1	00:50:00	10	460	2500	45	18.2345	27.6789	540	12:20:00	29/06/17	13:10:00
2	01:00:10	8	470	1900	34	18.3012	27.7777	530	12:10:00	29/06/17	13:10:10
3	00:23:00	25	480	3300	65	18.3113	28.1234	550	12:46:40	29/06/17	13:10:20

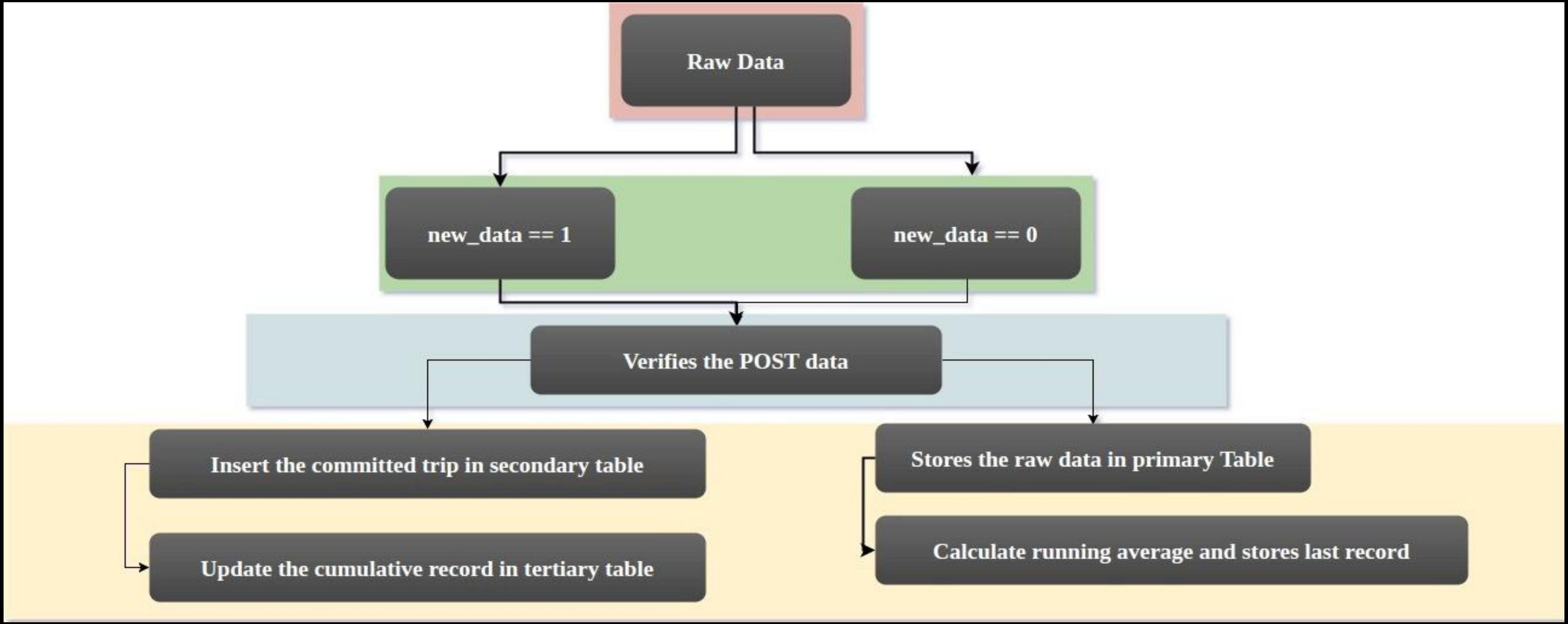
Format in which trip-wise details are stored in Secondary Table

Tertiary Table

ID	trips_no	avg_durtn	avg_dist	avg_speed	avg_erpm	avg_load	avg_throttle	latitude	longitude	total_dist	total_duratn
1	200	00:35:34	15	23	2500	45	32	18.2345	27.6789	3000	13:10:00
2	23	01:00:54	25	37	1900	34	41	18.3012	27.7777	575	13:10:10
3	134	02:01:41	33	41	3300	65	23	18.3113	28.1234	4422	13:10:20

Format in which cumulative records are stored in Tertiary Table

Data Flow Map



Server -API

Few examples :

Our URL : 139.59.38.17

API endpoint to fetch data :
139.59.38.17/data/<device_id>/charts/<chart_id>

API endpoint for location: http://139.59.38.17/data/location

API endpoint for last trip path : http://139.59.38.17/data/<device_id>/trip/<trip_id>

API endpoint to POST new data to server : 139.59.38.17/new

Problem 4

Sol: 139.59.38.17

Frontend

Phi-Matrix. Pixel Perfect.
Google Charts. Reliable.
Google Material Design. Responsive.

Track

Login.
Location.
Raw Data.
Visualization.

GITHUB

Github Repo : https://github.com/krpratik/sedemac-iot.git