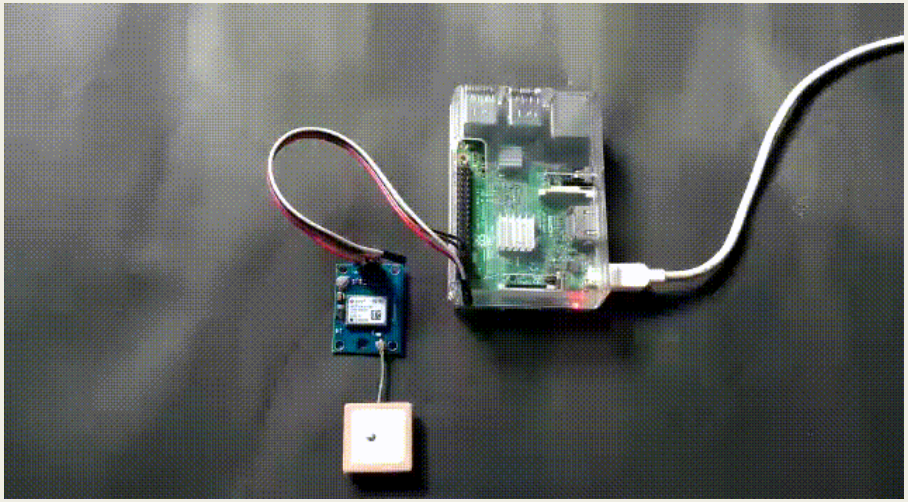
SKYMON: PI GPS TRACKING MODULE

Amechi Nwaeze, Roque De Leon, Sebastian Pena,
Michael Roller
CIS - 4355
Spring Semester

Executive Summary

- Objective:

This project aims to create a real-time GPS tracking system by integrating a Neo-6M GPS module with a Raspberry Pi 3 and Google Cloud. The system provides accurate, reliable tracking while ensuring seamless data management and mapping through the cloud platform.



Executive Summary

- Background:
  - *The Raspberry Pi 3, a versatile single-board computer, is known for its adaptability in various projects. Its GPIO pins enable easy connection with external sensors and modules, such as the Neo-6M GPS antenna, which provides accurate geographic coordinates. Together, with resources from Google Cloud Platform, they form a compact, customizable GPS tracking system that efficiently transforms data into useful software.*

Methodology

- GPS Module Setup:
  - *The first step involves configuring the Neo-6M GPS antenna module with the Raspberry Pi 3. The module will be connected via the GPIO pins, and appropriate drivers and software will be installed to ensure smooth communication between the hardware components.*
  - *Python libraries, such as gpsd, will be used to parse the GPS data from the Neo-6M module, enabling the retrieval of geographic coordinates.*

Methodology

- Google Cloud APIs and Services:
  - *A Google Cloud account is needed to build this project's underlying software and processing functions.*
  - *APIs and Permissions: Compute Engine API, bigquery, bigtable admin, bigtable data (readwrite), cloud database and debugger, cloud sql, compute engine(read write), Identity and API access, Vertex AI, Google Maps, Google Storage.*
  - *Cloud Buckets and Blobs to store data in multiple places*
  - *VPC networking*
  - *IAM roles and policy's to grant multi-user access*
  - *Compute Engine for virtual machines*

Milestone 1: Research

- Neo-6M GPS Module & Raspberry Pi 3: Investigate capabilities and requirements.
- APIs Exploration: Understand PubNub & Maps JavaScript API functionalities and integration methods.
- Requirements Analysis: Identify software and hardware needs.
- Virtual Machine (VM) Configuration: Discover deployment options.
- Google Cloud Efficiency: Maximize cost-efficient services for project success.

Milestone 1: Research

Enhancing the Setup:

- Google Maps API: Real-time streaming solution.
- Python Folium Library: Creates meaningful maps and diagrams.
- Vertex AI Workbench: Balances data processing workloads.

Exploration Objectives:

- Google Cloud Products: Store, protect, and transform data into valuable services.
- Google Cloud Solutions: Mapping, storage, connectivity, and computing power.

Milestone 2: GPS Module Setup

Connections:

- *Connect Neo-6M GPS module to Raspberry Pi 3 via GPIO.*
- *Install & configure drivers/Python libraries (gpsd, pynmea2) for accurate coordinate parsing.*

Data Verification:

- *Use cat command on TTYAMA0 to confirm GPS data flow.*
- *Focus on NMEA sentences: $GPGLL (latitude/longitude), $GPRMC (minimum GPS data).*

Milestone 2: GPS Module Setup

Data Collection Challenges:

- *Module requires a clear sky for optimal satellite signal.*
- *Basement testing led to scattered data output.*

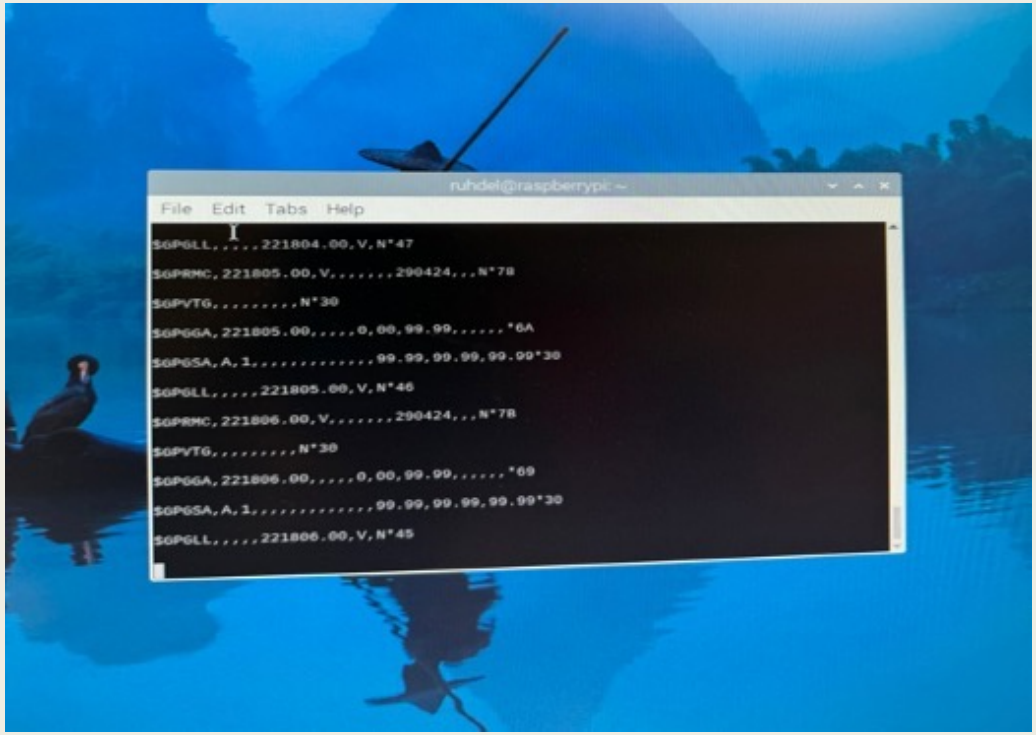
Data Processing & Transformation:

- *Develop a program to transform & store data in Google Cloud.*
- *Use pandas to convert coordinates into a DataFrame before uploading.*
- *Add run.py to /.xprofile for automatic execution.*
- *Set up Google Cloud project & obtain API keys.*
- *Utilize Jupyter notebooks for Python dependencies.*

Milestone 2: GPS Module Setup

```python
# Import necessary modules
import serial  # To handle serial port communication
import time    # For potential time-related functions (though not used directly
import string  # For string manipulation (also not directly used here)
import pynmea2 # For parsing NMEA data, specifically for GPS information

# Start an infinite loop to continually check for GPS data
while True:
    # Specify the serial port connected to the GPS module
    port = "/dev/ttyAMA0"

    # Create a serial object to communicate with the GPS module.
    # - `baudrate`: Speed of communication with the GPS module (typically 9600)
    # - `timeout`: Maximum waiting time for reading data (in seconds)
    ser = serial.Serial(port, baudrate=9600, timeout=0.5)

    # Initialize an NMEA stream reader to process incoming GPS data
    dataout = pynmea2.NMEAStreamReader()

    # Read a single line of data from the serial port (expected NMEA sentence fo
    newdata = ser.readline()

    # Check if the data is of type RMC (Recommended Minimum Navigation Informati
    if newdata[0:6] == "$GPRMC":
        # Parse the NMEA data using pynmea2 to extract useful GPS information
        newmsg = pynmea2.parse(newdata)

        # Extract latitude and longitude from the parsed message
        lat = newmsg.latitude
        lng = newmsg.longitude

        # Format the latitude and longitude values into a readable string
        gps = "Latitude=" + str(lat) + " and Longitude=" + str(lng)

        # Print the GPS coordinates to the console
        print(
```

Milestone 2: GPS Module Setup



Required APIs:

- *Google Cloud Platform*
- *Google Cloud Storage*
- *Google Maps*
- *IAM Management*
- *Cloud Storage Management*
- *Google Vertex AI*
- *Google Compute Engine*
- *Google BigQuery*
- *Cloud SQL*

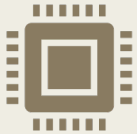
Milestone 3: API Integration

Milestone 3: API Integration

IAM Management:

Use IAM service to grant team access with Google Cloud emails.

Assign roles appropriately in the IAM page.



Google Maps API:

Obtain a separate key through the Cloud Console search bar.

Save for future use.

Service Account Keys:

Generate Raspberry Pi's service account key via IAM's "Create Service Account."

Save as ’cloud_key.json’ for program access.

VM service accounts created by default if VPC and location data match.

Milestone 4: Cloud Network Configuration

Bucket Configuration

- *Use Google Cloud's bucket storage service to store data from the satellite Python program.*
- *Create a bucket with a unique name in the same location as your VPC for optimal performance.*
- *Permissions:*
  - Allow universal read/write for service accounts.
  - Ensure proper resource access for VMs.

Google Cloud VMs

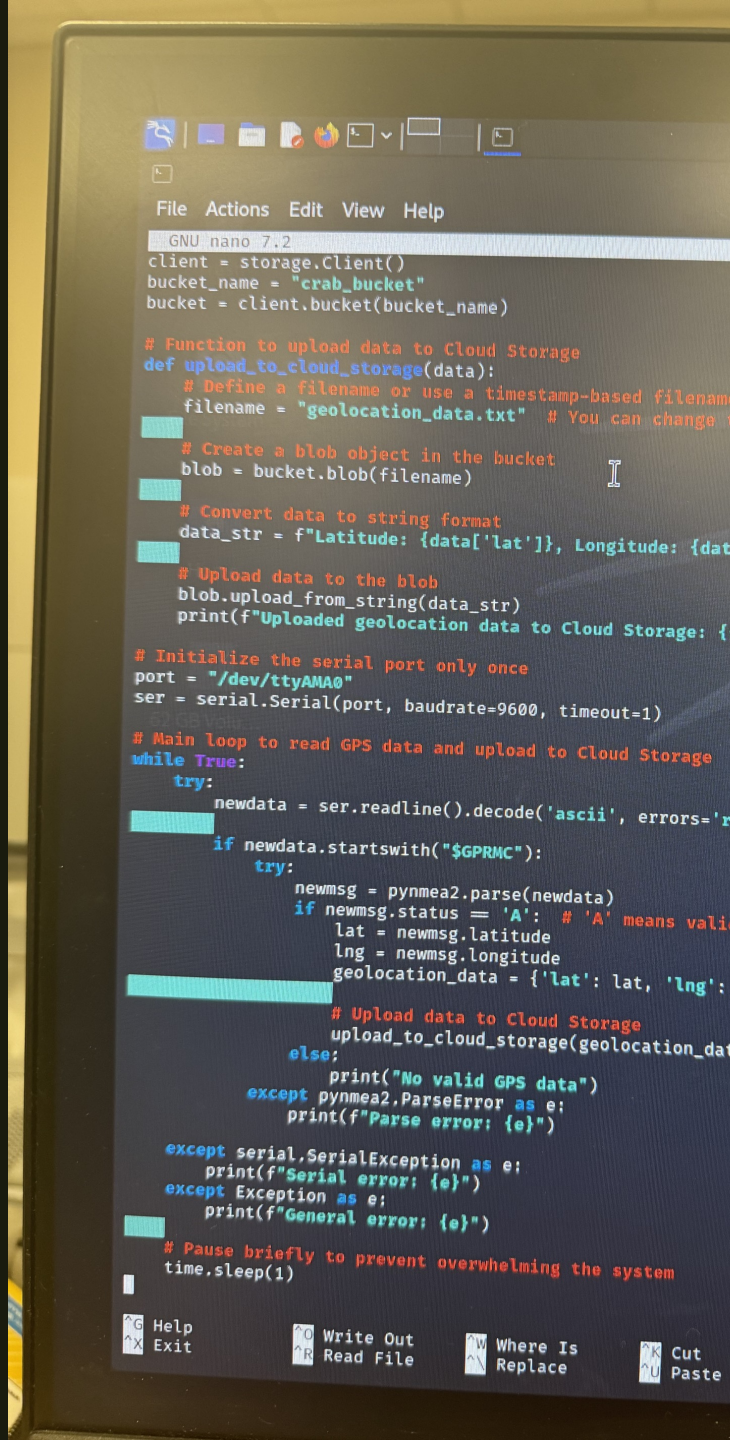
- *Deploy Python Compute Instances:*
- *Vertex AI Workbench: Create new instances, ensuring proper location and permissions.*
- *Use JupyterLab for step-by-step code testing.*
- *Google Compute Engine VM:*
  - Search "Compute Engine" and create a new instance.
  - Ensure location and zone align with your VPC.

Configuration Tips

- *Use project JupyterLab/Python apps to install Skymon/SkymonTracker.*
- *Focus on troubleshooting and optimizing configurations.*

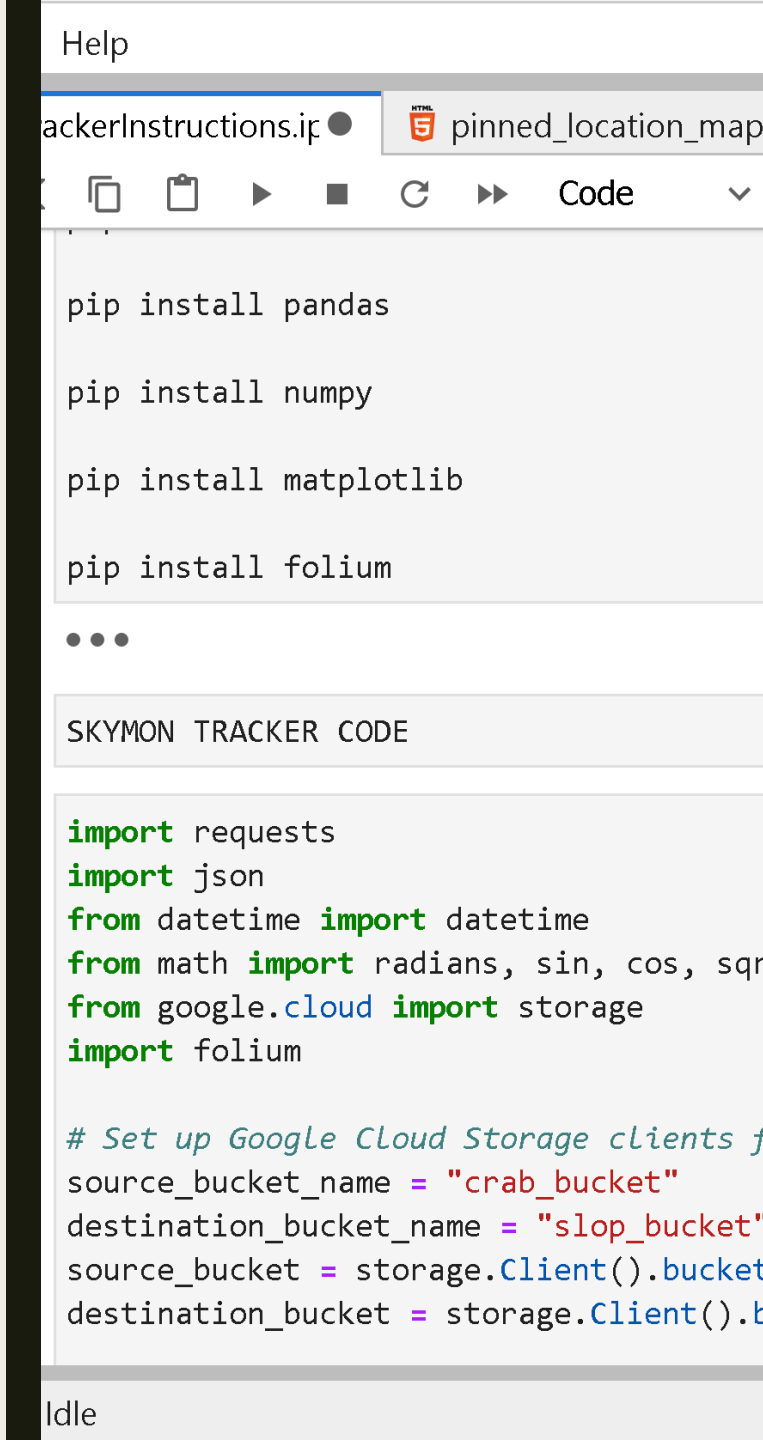
Milestone 4: Cloud Network Configuration

- Code to uploads geolocation data to cloud storage buckets
- Converts data to string because presented, unfiltered, data isn't formatted the way we want it
- Briefly pauses to prevent system overwhelming or crash



Milestone 4: Cloud Network Configuration

- Pandas library for fast and structured data structure
- Numpy in order to clean up data that is received
- Matplotlib in order to plot points and visualize on map
- Folium used in order to visualize manipulated data



Milestone 5: Testing and Troubleshooting

Testing Goals:

- *Identify and document errors or misconfigurations in the program.*
- *Ensure VMs communicate with the cloud and have correct access keys.*
- *Verify bucket configuration for data transfer.*
- *Ensure accurate data collection, transmission, and visualization.*

Milestone 5: Testing and Troubleshooting

Configuration for Skymon on Google Cloud VMs:

- Python Setup: Ensure Python 3, dependencies, and libraries are available for Skymon.
- Bucket Configuration: Use accurate bucket names and filenames.
- Environment Variable: Set Raspberry Pi variable for Google Cloud access via RSA key.
- API Configuration: Finalize API integration and explore new development areas.

Skymon Features:

- Retrieves raw data from the Raspberry Pi storage bucket.
- Creates interactive maps and visualizations of Pi's last location.
- Stores data with the ability to append files to the finalized cloud bucket.

Skymon Tracker Features:

- Receives coordinate input to monitor areas within a 5-mile radius.
- Logs coordinates when Pi is within user-defined zones, storing these for review.

Troubleshooting:

- *Secure all connections between Neo-6M GPS module and Raspberry Pi 3 GPIO pins.*
- *Install/update appropriate drivers.*
- *Test GPS module with a different Pi or GPIO pins to rule out hardware issues.*
- *Ensure stable soldering for proper power and connectivity.*
- *Operate in a clear-sky area for accurate satellite communication.*

Milestone 5: Testing and Troubleshooting

Conclusion

Final Accomplishments:

- Deployed the GPS tracking solution for real-world use, ensuring continuous operation with user feedback and testing.
- Skills acquired:
  - *Build an IoT device with a satellite receiver and cloud integration.*
  - *Plan and manage a cloud infrastructure.*
  - *Develop IoT software and applications for Google Cloud.*
  - *Manage and transform data using buckets and blobs.*
  - *Balance cost and time to meet project deadlines*

Result:

Achieved a fully functional GPS tracking system using a Raspberry Pi 3, Neo-6M GPS module, and cloud-based APIs, demonstrating the scalability and cost-efficiency of cloud technologies.

Conclusion

Issues and Future Development

- Issues:
  - *Need for outdoor monitoring to receive proper data via Neo-6M.*
  - *Outdoor power source required for Raspberry Pi, memorizing Python keystrokes.*
- Future Development:
  - *Portable case and power supply for easy satellite access.*
  - *Automated startup script to remove the need for a monitor.*
  - *Run cloud resources continuously for Skymon/Tracker.*
  - *More public infrastructure APIs for niche applications.*
  - *Enhanced UI with search queries and touch-selectable pin-drop locations.*
  - *Web application deployment for centralized control.*

References

- *GPS - NMEA Sentence Information*, Glenn Baddeley, aprs.gids.nl/nmea/. Accessed 6 May 2024.
- “Make a Realtime GPS Tracker Device with Raspberry Pi.” *SPARKLERS*, Arijit Das, 11 July 2019, sparklers-the-makers.github.io/blog/robotics/realtime-gps-tracker-with-raspberry-pi/.
- “Use NEO 6M GPS Module with Raspberry Pi and Python.” *SPARKLERS*, Arijit Das, 11 July 2019, sparklers-the-makers.github.io/blog/robotics/use-neo-6m-module-with-raspberry-pi/.
- “What Are Realtime Apis?” *PubNub*, 26 Apr. 2024, www.pubnub.com/guides/realtime-api/.

Cloud Documentation

Blobs/Objects

https://cloud.google.com/python/docs/reference/storage/1.25.0/blobs

Blob Storage (Read)

https://cloud.google.com/bigquery/docs/omni-azure-create-connection

Blob Storage (Write)

https://cloud.google.com/appengine/docs/legacy/standard/python/googlecloudstorageclient/read-write-to-cloud-storage

Bucket Storage

https://cloud.google.com/storage/docs/json_api/v1/buckets

Compute Engine

https://cloud.google.com/compute/docs

Compute Engine VM Pricing

https://cloud.google.com/compute/all-pricing

Identity and Access Management

https://cloud.google.com/security/products/iam

Regions and Zones

https://cloud.google.com/about/locations

Service Accounts

https://cloud.google.com/iam/docs/service-account-overview

Service Accounts (GCLoud VMs)

https://cloud.google.com/compute/docs/access/service-accounts

Vertex AI

https://cloud.google.com/vertex-ai?hl=en

VPC Networking

https://cloud.google.com/vpc/docs/vpc