Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Alangilan Campus
Golden Country Homes, Alangilan Batangas City, Batangas, Philippines 4200
Tel Nos.: (+63 43) 425-0139 local 2121 / 2221
E-mail Address: coe.alangilan@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph
College of Engineering - Department of Electronics Engineering

Project Title:

Pawsitive Collar Tracking System: A GPS-Based Pet Locator

A Group Project Paper Submitted to:
Engr. Gil Barte
Batangas State University - The NAtional Engineering University
Alangilan, Batangas City

In Partial Fulfillment
Of the Requirements for
ECE 415: Microprocessor and Microcontroller Systems and Design
Bachelor of Science in Instrumentation and Control Engineering

By:
Mark Zedrick Caya
Aira Rose Sarmiento
Gerome De Chavez

ICE-3203

Date:
May 23, 2025

A. Introduction

The problem of lost pets is a significant one, causing stress or anxiety for many owners. Our solution is the Pawsitive Collar Tracking System, a GPS-based pet locator that provides peace of mind and real time tracking to quickly locate their beloved companions to the pet owners. The system will utilize GPS technology integrated into a comfortable pet collar, offering a real time tracking and a range of features designed for both pet and owner convenience. Beyond location tracking, the Pawsitive Collar aims to improve pet safety and responsible pet ownership.

B. Objectives

General Objective:

To design, develop, and test a functional and user-friendly GPS-based pet tracking system (Pawsitive Collar) that provides real-time location data to pet owners via a mobile application, enhancing pet safety and responsible pet ownership.

Specific Objective:

a. Design and Implement a GPS Tracking Collar: Choosing appropriate hardware components(GPS module and battery), ensuring durability and comfort for the pet.
b. Select and Integrate with a Suitable Existing Mobile Application: Research and evaluate available pet tracking applications, selecting one that offers real-time location tracking.
c. Implement a reliable and secure method for transmitting location data from the collar to the mobile application, addressing potential issues like data loss and signal interference.
d. Design a power management system for the collar that maximizes battery life while maintaining consistent operation.
e. Testing and Evaluation: Conduct thorough testing and evaluation of the entire system to assess its functionality, accuracy, reliability, and usability.

This project aims to integrate a GPS tracking system, microcontroller and communication system to track the pet's real time location and transmit data to a mobile application.

C. Considerations

- **Hardware Considerations:**
    a. GPS Module Selection: Accuracy, power consumption, size and cost are all critical factors
    b. Microcontroller Choice (ESP32/Arduino): Processing power, memory, power consumption, and available libraries/support are key.
    c. Wireless communication: Bluetooth offers shorter range but lower power consumption, while Wi-fi or cellular provides longer range but higher power consumption.
    d. Power Source: Battery type. Consider the size and weight constraints of the collar

e. Collar Design: Material selection, size and adjustability for different pet size, and overall weight are essential for pet comfort and safety.

- **Software Considerations:**
  a. Firmware Development: The microcontroller firmware needs to handle GPS data acquisition, processing, and transmission.
  b. Mobile app: User interface design(easy to navigate), map integration and data visualization are crucial. Consider both iOS and Android platforms.
  c. Data Storage and Management: How will location data be stored on the phone?

- **Testing and Evaluation Considerations:**
  a. Accuracy Testing: thorough testing under various conditions like different environments and weather is needed to validate the GPS accuracy.
  b. Battery Life Testing: consider to test the system's battery life under typical usage scenarios to ensure it meets the expectations.
  c. Safety Testing: Ensure the collar is safe for the pet, considering potential hazards

D. Design Constraint

a. Timeframe: We have a limited time to complete the project, which makes it challenging to ensure everything is properly developed and tested. As the saying goes, "everything valuable takes time to build."

b. Skills: Since we are using Arduino and ESP32, coding is a major part of the project. We need to develop technical and analytical skills to troubleshoot issues and ensure smooth operation.

c. Software and Hardware Integration: The components we are using—Arduino, ESP32, and GPS—must be properly connected and programmed to work together efficiently for the best results.

d. Manufacturing and Assembly: The physical assembly of the collar is just as important as the programming. The product must be well-assembled and aligned with the intended design to ensure it functions correctly.

E. Working Principles

GPS Module with ESP32/Arduino: The collar contains a small GPS device that receives signals from satellites in space. These satellites provide information about their location, and the GPS in the collar uses this data to determine the exact position of the pet. The ESP32 or Arduino reads and processes this data through the code that is programmed into it.

Microcontroller: acts as the "brain" of the collar. It processes the location data from the GPS and decides what to do with it, such as storing it or sending it to a mobile phone. The microcontroller also manages power consumption to ensure the collar can last as long as possible on a single charge.Once the collar has the pet's location, it sends this information to a phone or an online server using Bluetooth or Wi-Fi. Bluetooth is more suitable for short-range tracking, while Wi-Fi or cellular networks allow for longer-distance tracking, depending on the design.

Mobile App: The pet's location can be tracked using a mobile app that displays the position on a map in real time. The app communicates with the collar through Bluetooth or Wi-Fi, enabling the user to see where their pet is, whether they are close by or far away.

As students, we may have limited knowledge about microcontrollers, GPS, and wireless communication. This project will help us gain hands-on experience in these areas, learning how to integrate components, write code, and troubleshoot.

F. Essential Components

For GPS Tracker:

- GPS MODULE (NEO-6M or NEO-8M) - This component will be used to get the location of the pet. This module receives signals from GPS satellites to determine its precise location.

NEO-6M GPS MODULE SPECIFICATIONS	
Parameter Name	**Parameter Value**
Receiver Type	50 channels, GPS L1(1575.42Mhz)
Horizontal Position Accuracy	2.5m
Navigation Update Rate	1HZ (5Hz maximum)
Capture Time	Cool start: 27sHot start: 1s
Navigation Sensitivity	-161dBm
Communication Protocol	NMEA, UBX Binary, RTCM
Serial Baud Rate	4800-230400 (default 9600)
Operating Temperature	-40°C ~ 85°C
Operating Voltage	2.7V ~ 3.6V
Operating Current	45mA
TXD/RXD Impedance	510Ω

- ESP 32:

ESP32 SPECIFICATIONS	
Parameter Name	**Parameter Value**
Operating Voltage	3.3 V
DC Current on 3.3V Pin	50 mA
DC Current on I/O Pins	40 mA
Maximum Operating Frequency	240MHz
Timers	2 x 64-bit Timers, 1 RTC Timer

- OLED - The OLED Display is used to visually present important system information. It shows the real-time location data from the GPS Module, confirms when SMS alerts are sent via the GSM Module.

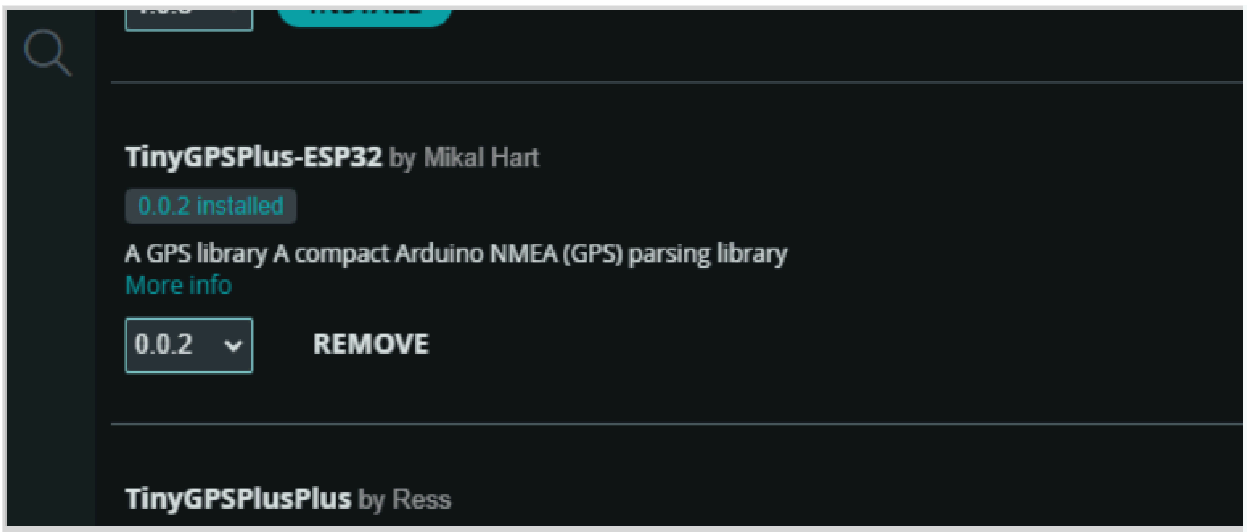
- Wires - These connect all the components, allowing data and power to flow between them.

G. Pseudocode:

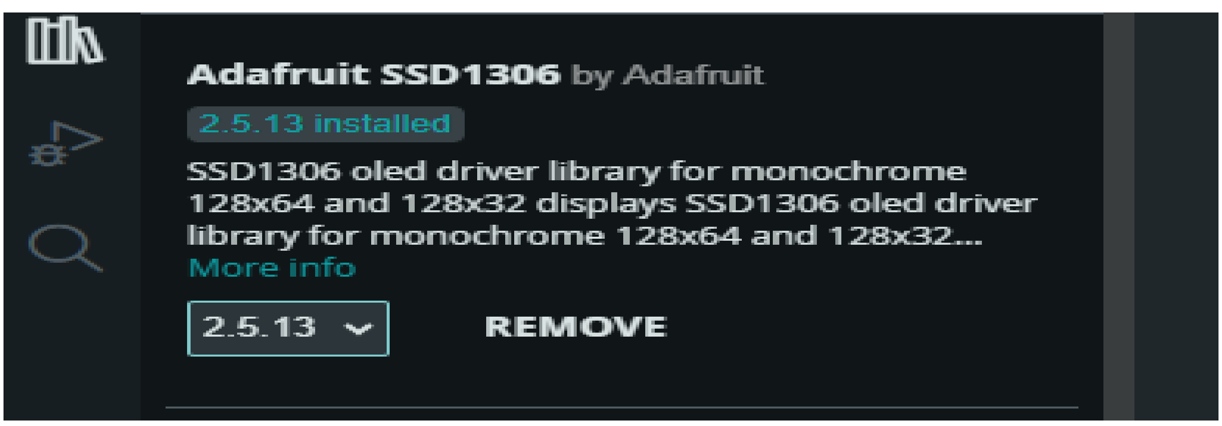
```
SETUP FIRST
INCLUDE all necessary toolboxes (libraries):
- Wire (for I2C communication)
- OLED display tools
- GPS decoder tools
- Hardware serial tools

DEFINE OLED screen settings:
- Width: 128 pixels
- Height: 64 pixels
- No reset pin needed

DEFINE GPS connections:
- GPS TX → ESP32 Pin 16 (RX)
- GPS RX ← ESP32 Pin 17 (TX)

CREATE objects:
- OLED display
- GPS decoder
- Serial port for GPS

/WHEN THE DEVICE STARTS
SETUP:
  BEGIN Serial Monitor for debugging at 115200 speed
  START GPS communication at 9600 speed

  TRY to initialize OLED display:
    IF display fails to start:
```

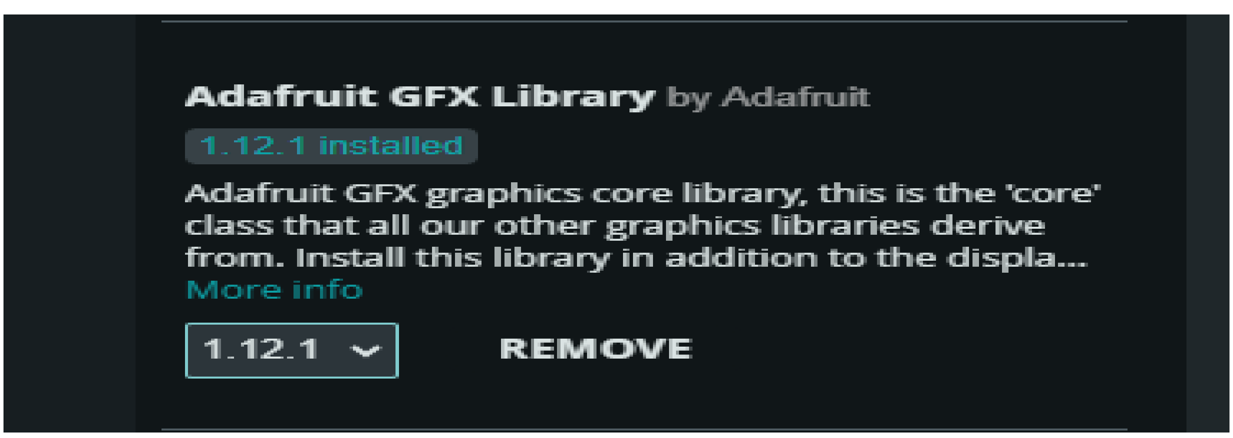
```
    PRINT "OLED failed!" to Serial Monitor
    FREEZE program (stop forever)
  ELSE:
    CLEAR display
    SET text size to normal
    SET text color to white
    SHOW welcome message: "GPS Debug Mode"
    SHOW instructions: "Press 'D' for raw data"

MAIN LOOP WOULD RUN FOREVER
LOOP:
 // Check if user wants to toggle raw data mode
 IF user presses 'D' in Serial Monitor:
   TOGGLE raw data mode ON/OFF

 // Process incoming GPS data
 WHILE GPS is sending data:
   READ one character from GPS

   IF raw data mode is ON:
     SEND the character to Serial Monitor (for debugging)

   TRY to decode the GPS data:
     IF decoding succeeds:
       UPDATE the OLED display with new info

 UPDATE DISPLAY FUNCTION :>
FUNCTION updateDisplay:
 CLEAR the OLED screen

 IF GPS has valid location data:
   DISPLAY "Lat: [latitude with 6 decimals]"
   DISPLAY "Lng: [longitude with 6 decimals]"
   DISPLAY "Sats: [number of satellites]"
 ELSE:
   DISPLAY "No GPS fix"
   DISPLAY "Chars received: [number of characters processed]"

 SEND the updates to the physical display
```

Downloaded Necessary Library



This library is used to read and interpret GPS data from modules that use NMEA sentences. It can read information like latitude, longitude, speed, time and more. You can use it with GPS modules like the NEO-6M.





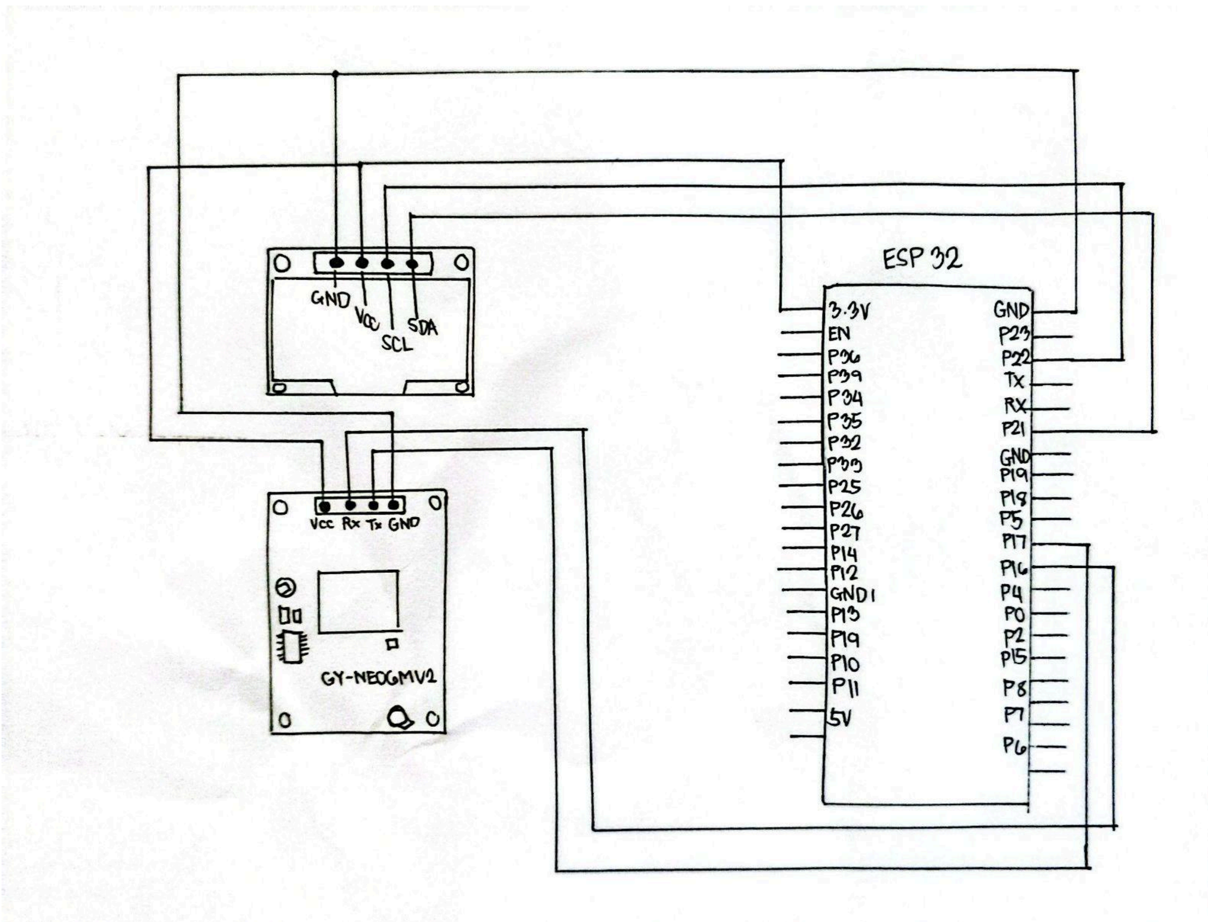
Updated Code:

```
#include <Adafruit_GFX.h>       // Graphics library for OLED
#include <Adafruit_SSD1306.h>   // OLED display library
#include <TinyGPSPlus.h>        // GPS data decoder
#include <HardwareSerial.h>     //ESP32 communicate with GPS
```

```
// OLED screen settings
#define SCREEN_WIDTH 128        // Screen is 128 pixels wide
#define SCREEN_HEIGHT 64        // and 64 pixels tall
#define OLED_RESET -1           // No reset pin needed
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
// Create OLED object

// GPS module connections
#define GPS_RX_PIN 16  // GPS TX wire → ESP32 RX (GPIO16) (GPS sends data here)
#define GPS_TX_PIN 17  // GPS RX wire ← ESP32 TX (GPIO17) (ESP32 sends commands
here)
HardwareSerial SerialGPS(1);
TinyGPSPlus gps;            // Create GPS decoder object

void setup() {
Serial.begin(115200);       // Start serial monitor for debugging

// Start communication with GPS module
SerialGPS.begin(9600, SERIAL_8N1, GPS_RX_PIN, GPS_TX_PIN); // 9600 baud speed

// Try to start OLED display
if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // 0x3C is OLED's address
Serial.println("OLED failed!");  // Show error if display doesn't work
while (1);                  // Freeze program if display fails
}

// Prepare the OLED display
display.clearDisplay();     // Clear any old content
display.setTextSize(1);     // Normal-sized text
display.setCursor(0, 0);    // Start writing at top-left corner
display.println("GPS Debug Mode");  // Show welcome message
display.println("Press 'D' for raw data"); // Instructions
display.display();          // Send text to display
}

void loop() {
static bool showRaw = false; // Tracks if we should show raw GPS data

// Check if user pressed 'D' in Serial Monitor
if (Serial.available()) {
if (Serial.read() == 'D') showRaw = !showRaw; // Toggle raw data mode
```

```
}

// Check for incoming GPS data
while (SerialGPS.available()) {
char c = SerialGPS.read(); // Read one character from GPS

// If raw data mode is ON, send GPS data to Serial Monitor
if (showRaw) Serial.write(c);

// Try to decode GPS data and update display if successful
if (gps.encode(c)) updateDisplay();
}
}

// Function to update the OLED screen with GPS info
void updateDisplay() {
display.clearDisplay();     // Wipe the display
display.setCursor(0, 0);    // TOP to LEFT ung sulat

// If GPS has valid location data
if (gps.location.isValid()) {
display.print("Lat: "); display.println(gps.location.lat(), 6); // Show latitude (6 decimal places for accurate output)
display.print("Lng: "); display.println(gps.location.lng(), 6); // Show longitude
display.print("Sats: "); display.println(gps.satellites.value()); // Number of satellites
} else {
// If no GPS signal
display.println("No GPS fix"); // Show error message
display.print("Chars: "); display.println(gps.charsProcessed()); // Show how much data was received
}
display.display(); // Send updates to screen
}
```

H. Schematic Diagram

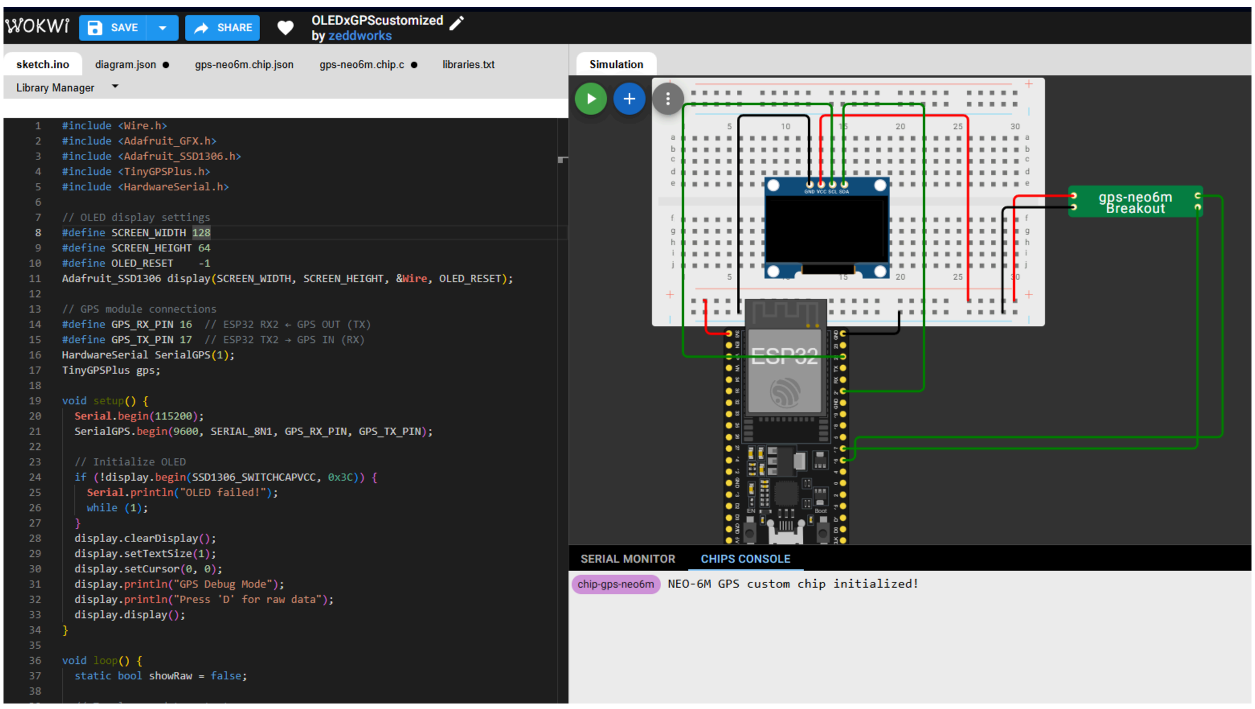


This schematic diagram shows the wiring connections between an ESP32 microcontroller, an I2C OLED display, and a GY-NEO6MV2 GPS module. The OLED display, which serves to visually show the real-time GPS data, is connected to the ESP32 using the I2C communication protocol. In the diagram, the OLED's GND pin is connected to one of the ESP32's ground pins, while its VCC pin is connected to the 3.3V power output of the ESP32 to supply power. The SDA (data) and SCL (clock) pins of the OLED are connected to GPIO 21 and GPIO 22 of the ESP32, respectively, which are the default pins for I2C communication.

The GPS module (GY-NEO6MV2) is powered by connecting its VCC pin to the 5V output of the ESP32 and its GND pin to one of the ESP32's ground pins. For serial communication, the TX (transmit) pin of the GPS module is connected to GPIO 16 (RX) of the ESP32, allowing the ESP32 to receive data from the GPS. The RX (receive) pin of the GPS is connected to GPIO 17 (TX) of the ESP32, enabling two-way communication if needed. This setup allows the ESP32 to collect GPS location data and display it in real time on the OLED screen, forming the core of the Pawsitive Collar tracking system. The diagram clearly illustrates how power and communication lines are managed between the components to ensure proper functionality.

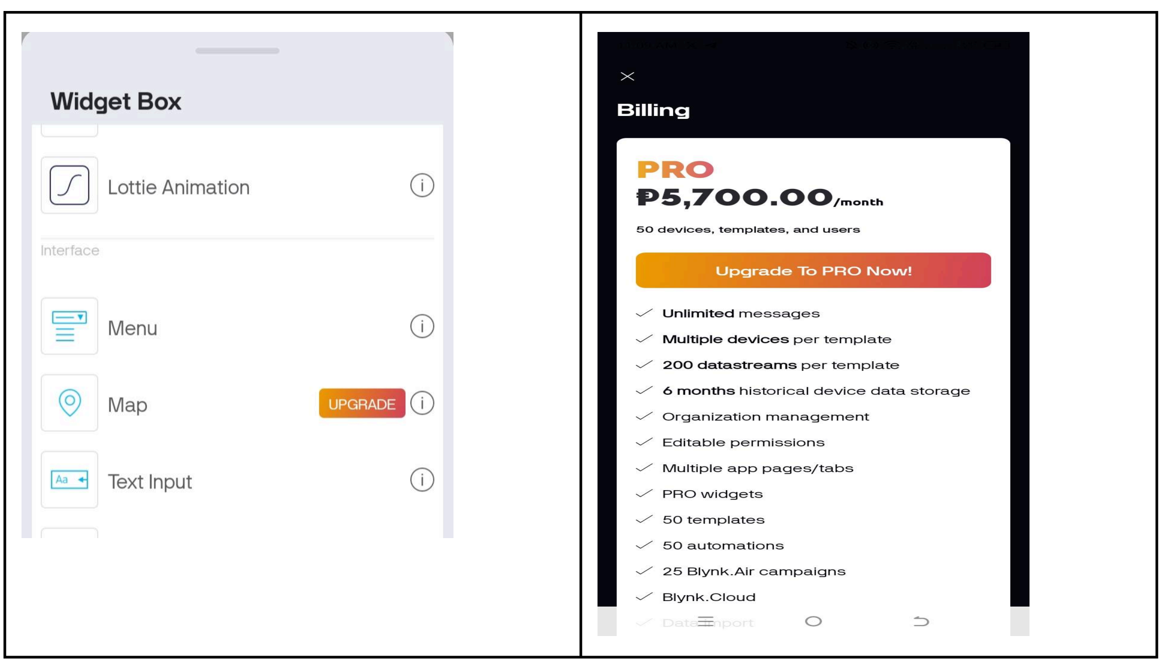
J. Simulation

https://wokwi.com/projects/432292837418023937



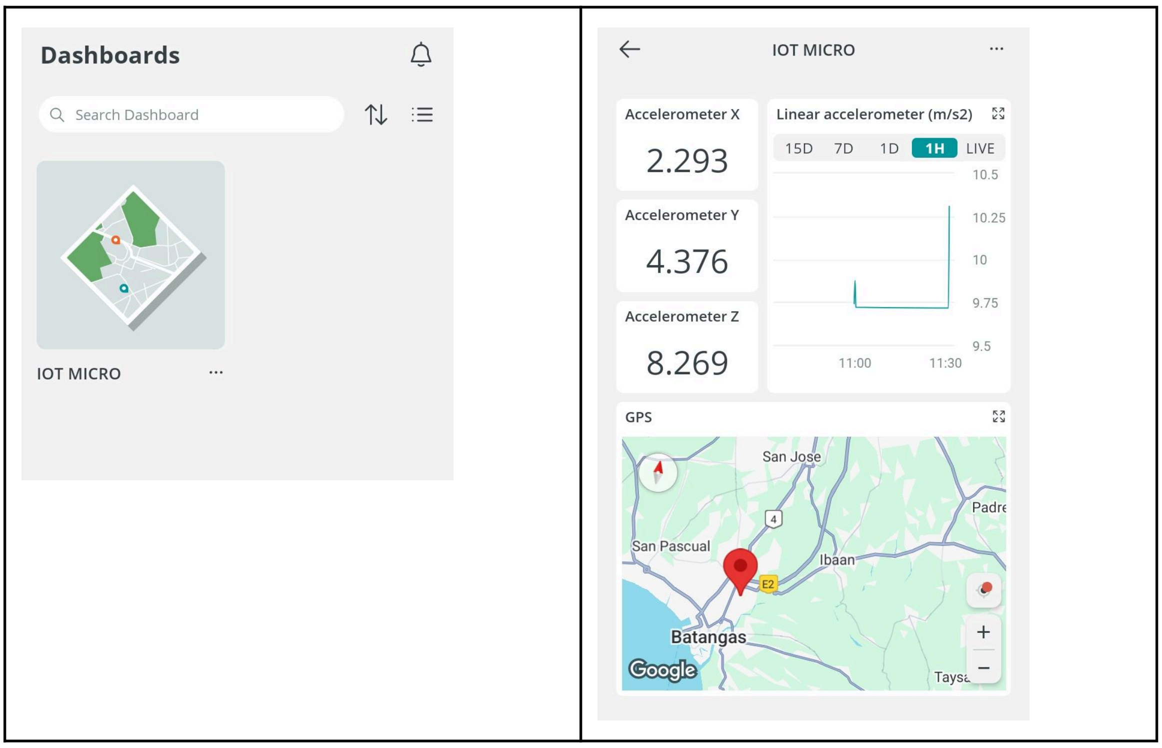
K. Interfacing IOT:

BLYNK APP



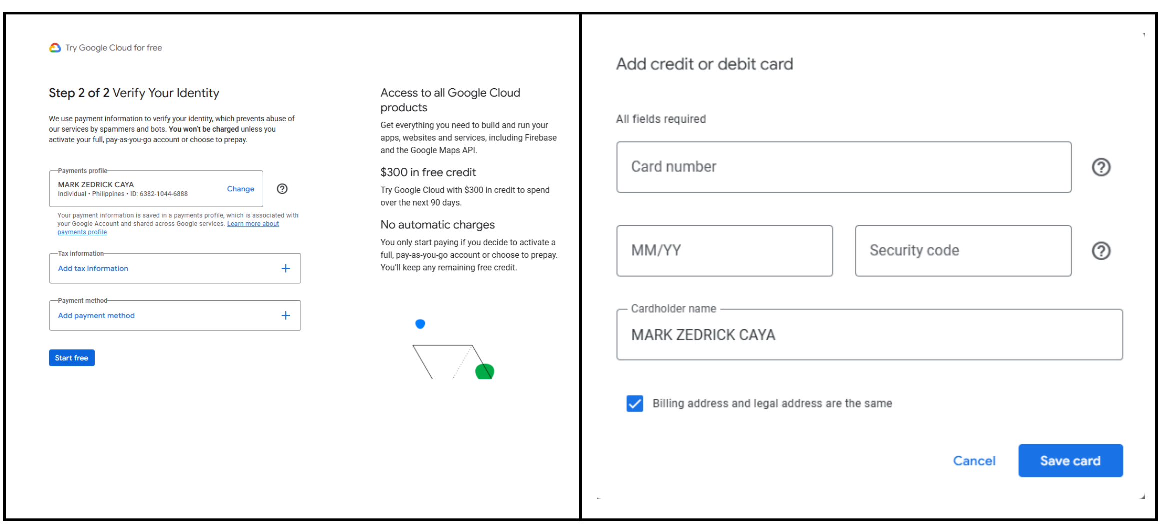
The primary reason we cannot use the Blynk app is that it requires a PRO subscription, which costs approximately $99.76. Additionally, the app lacks a feature that allows sending only latitude and longitude data

IOT REMOTE



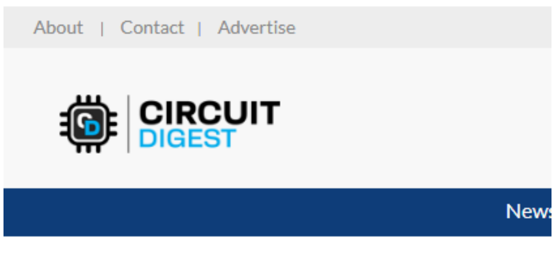
The app's name, 'IoT', doesn't quite reflect its functionality since it mainly uses the phone's built-in GPS rather than interfacing with external devices like Arduino. Moreover, it does not support direct connection with the Arduino IDE, limiting its use for hardware integration

CONSOLE CLOUD GOOGLE



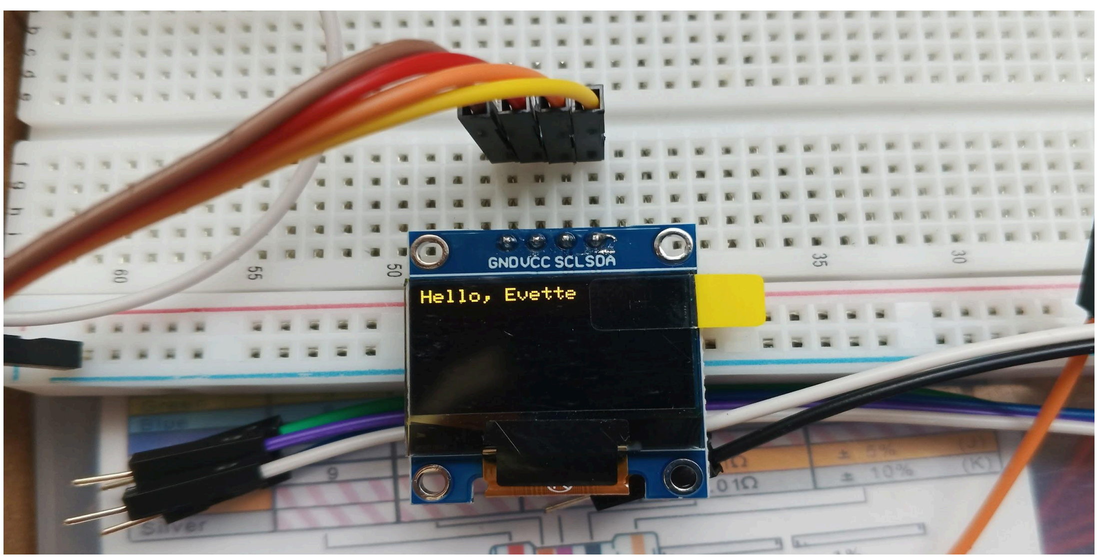
On this website, we don't have any card numbers to use. Also, it seemed fishy to ask for the security code.

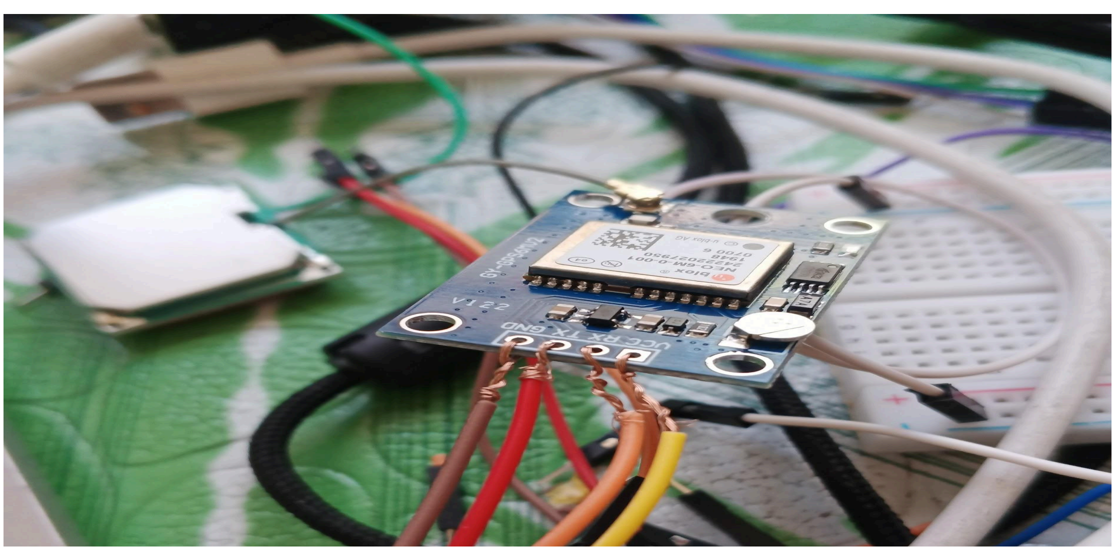
CIRCUITDIGEST.CLOUD

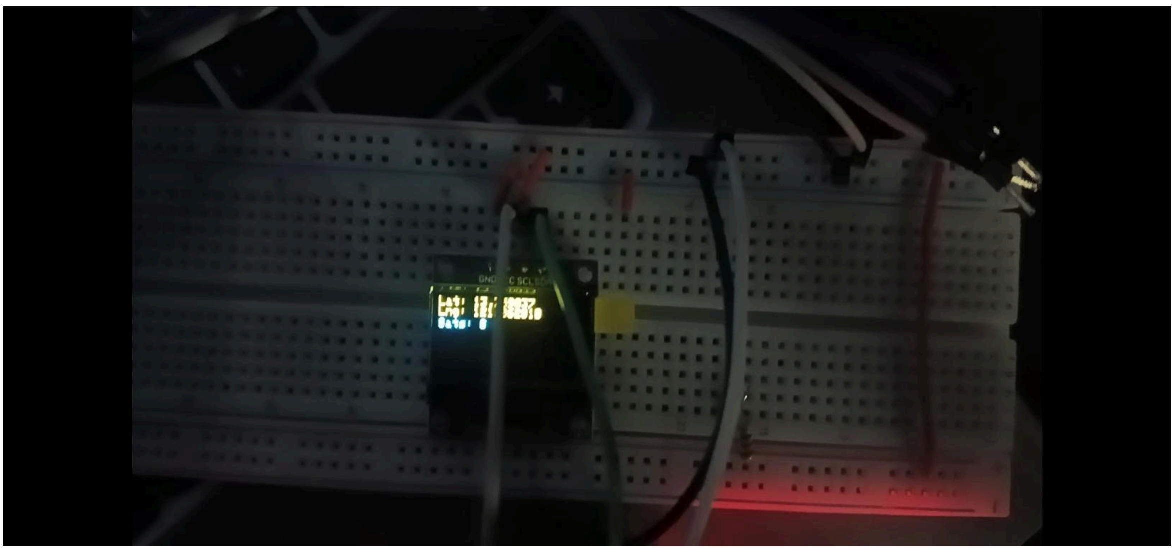
We followed all the steps on the website to obtain the API key—API KEY: IsDV6RULbRiV—but the server itself presents issues. Additionally, the API usage is limited to about 100 requests. Since our GPS sends signals every 0.5 to 1 second, using this API would only allow the system to function for roughly one and a half minutes before hitting the limit.

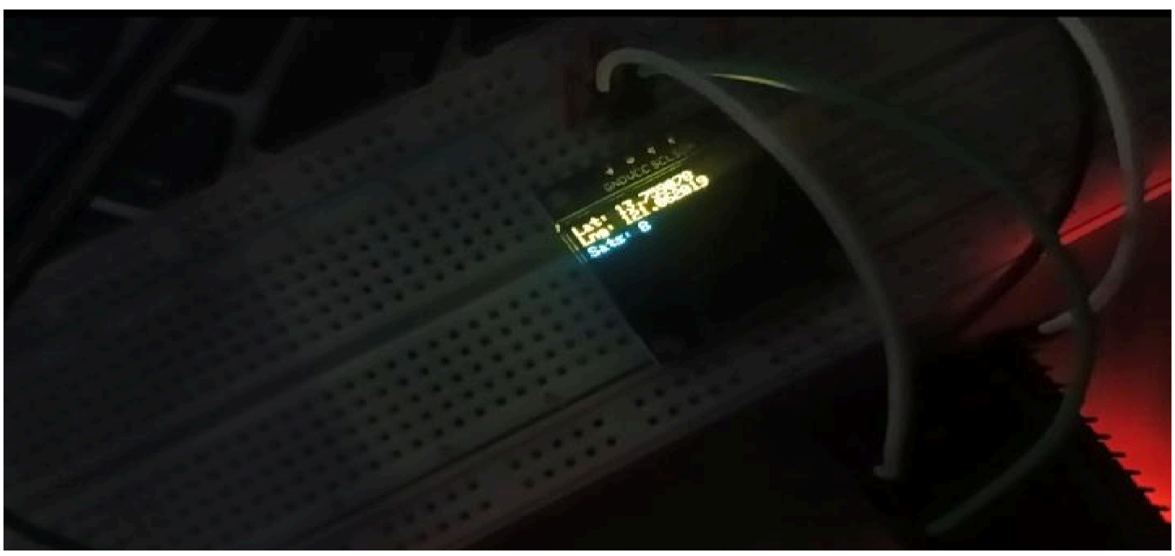
Hardwired Sample

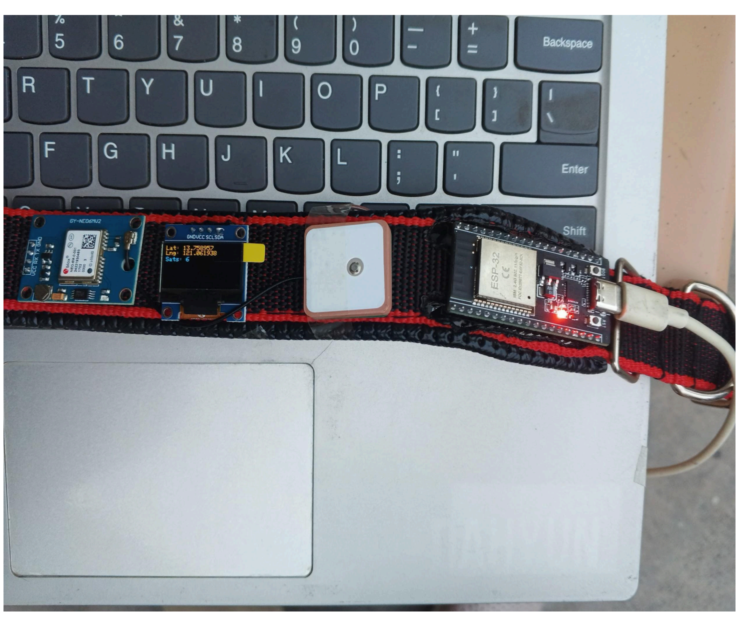


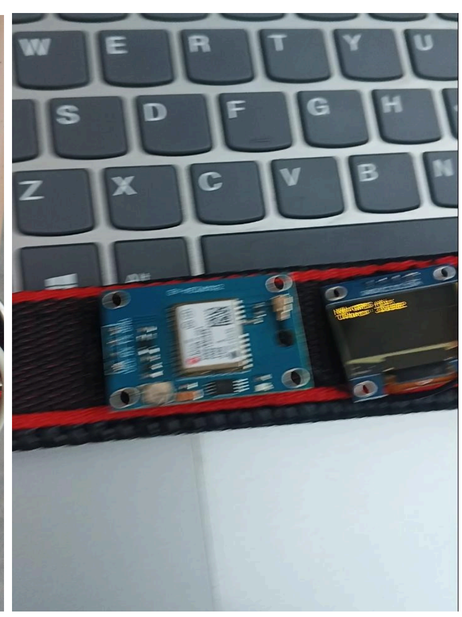


Documentation









Conclusion:

The development of the *Pawsitive Collar Tracking System* demonstrates the practical application of GPS technology in enhancing pet safety. Our team successfully designed and implemented a pet collar that accurately tracks real-time location data using a GPS module integrated with a microcontroller. This achievement reflects our understanding of embedded systems and real-world problem-solving using microprocessors.

However, despite our success in establishing reliable GPS tracking, the project encountered a major limitation: the integration of Internet of Things (IoT) functionality was not fully realized. Several platforms we explored either required expensive subscriptions (such as Blynk), lacked compatibility with Arduino IDE (such as IOT Remote), or presented security and usage constraints (such as Google Cloud and CircuitDigest). These barriers, especially budget and platform limitations, prevented us from successfully transmitting GPS data to a mobile application via the internet.

Nonetheless, this experience highlighted critical lessons in system design, troubleshooting, and the importance of aligning technical goals with available resources. While the current version of our collar functions well as a standalone GPS device, future iterations could incorporate IoT capabilities as platforms and tools become more accessible. Ultimately, the project laid a strong foundation for real-world applications and opened opportunities for future enhancements in pet tracking and smart wearables.