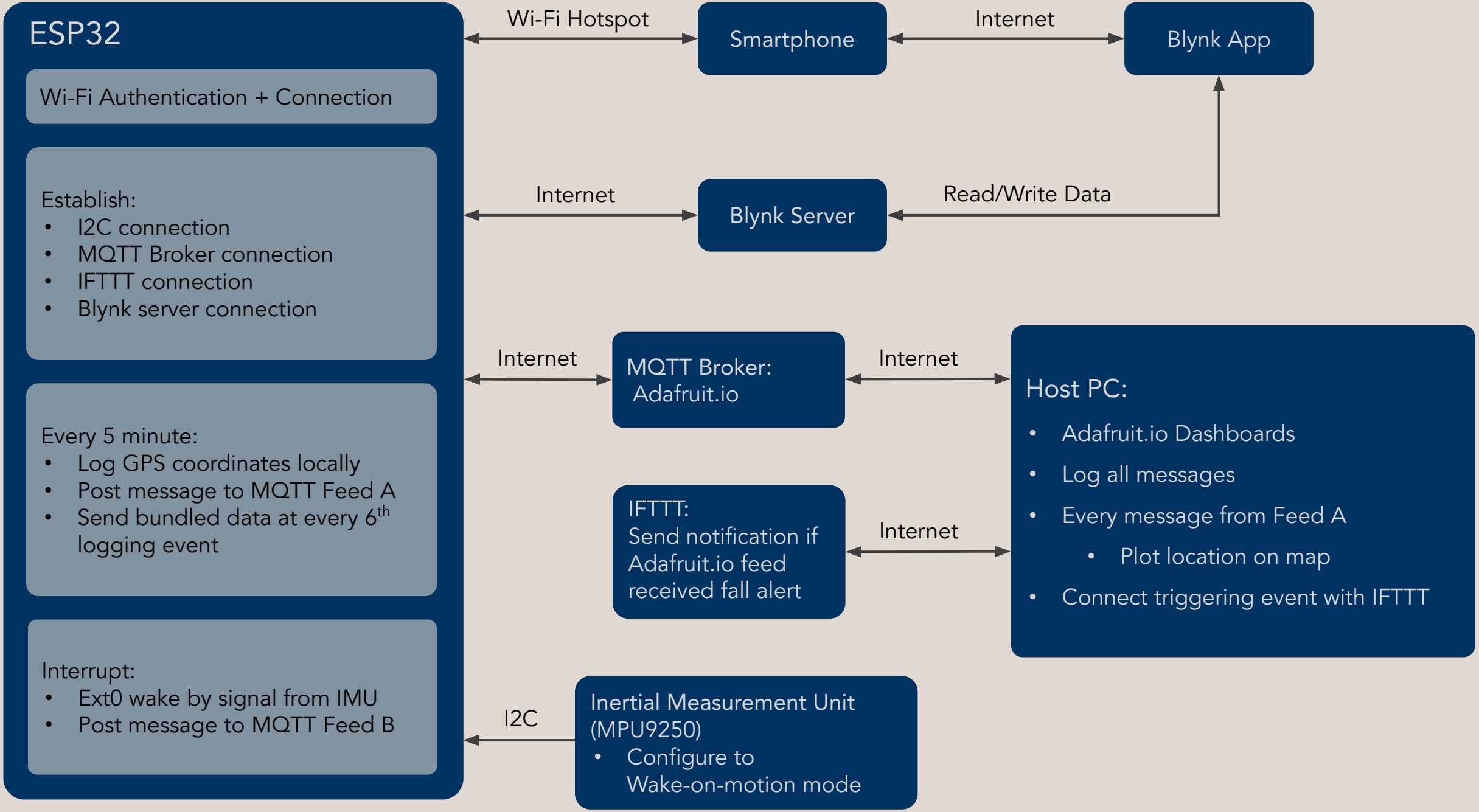
UCB FA20 ME100 Final Project

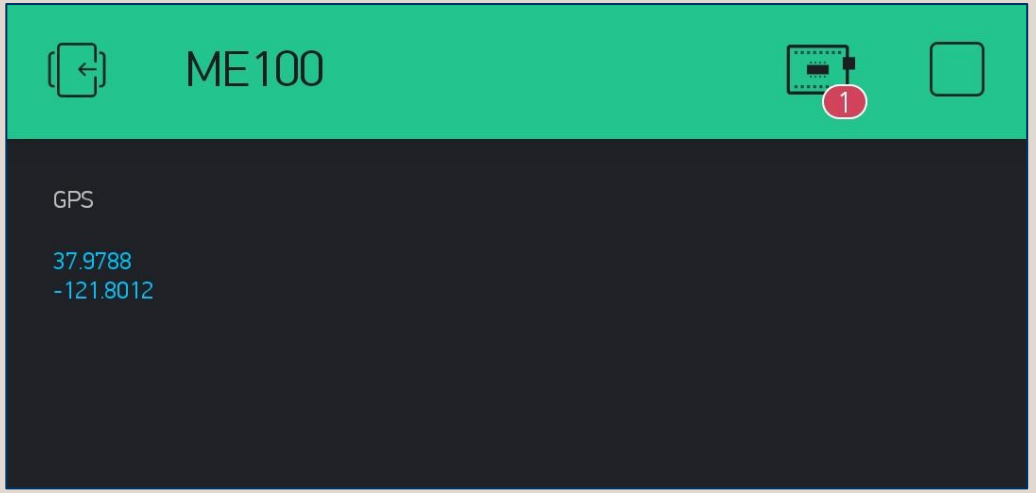
Elder Care Tracker

Video Demo: Click Here

Presented By: Chang Chen, Ivan Shao


ESP32
Wi-Fi Authentication + Connection
Establish:
• I2C connection
• MQTT Broker connection
• IFTTT connection
• Blynk server connection
Every 5 minute:
• Log GPS coordinates locally
• Post message to MQTT Feed A
• Send bundled data at every 6th logging event
Interrupt:
• Ext0 wake by signal from IMU
• Post message to MQTT Feed B
Wi-Fi Hotspot
Smartphone
Internet
Blynk App
Internet
Blynk Server
Read/Write Data
Internet
MQTT Broker:
Adafruit.io
Internet
Host PC:
• Adafruit.io Dashboards
• Log all messages
• Every message from Feed A
• Plot location on map
• Connect triggering event with IFTTT
IFTTT:
Send notification if Adafruit.io feed received fall alert
Internet
I2C
Inertial Measurement Unit (MPU9250)
• Configure to Wake-on-motion mode

Sensors



1. GPS

Access method: via Smartphone App (Blynk)

Description:

The Blynk App uses the Smartphone's GPS module to obtain the corresponding GPS coordinates. The data is uploaded to the Blynk server. The ESP32 then acquire the data from the Blynk server periodically.

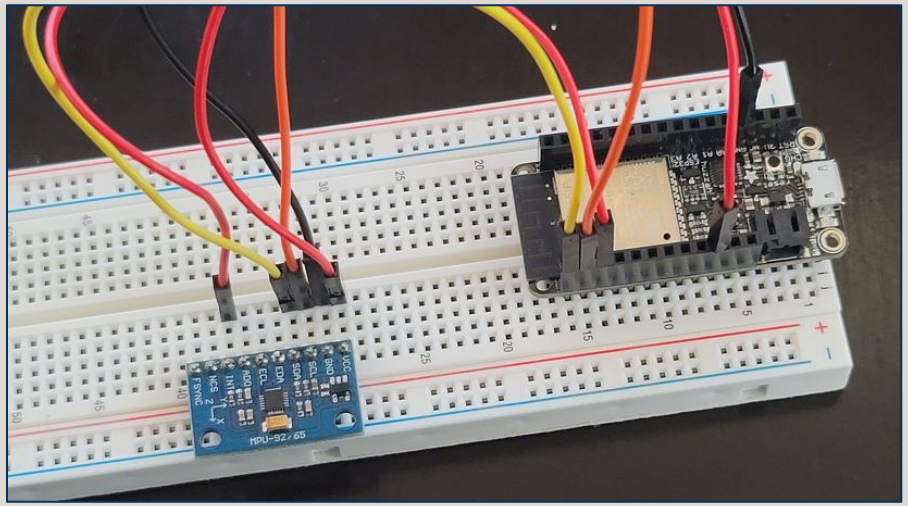
Performance:

The GPS sensor works as expected. However, the smartphone app only update the server when changes is detected.

Calibration:

The sensor accuracy within 10 meter. This error is within the acceptable range. No calibration is needed.

Sensors



2. IMU (MPU9250)

Access method: via I2C protocol

Description:

The MPU9250 is connected to the ESP32. Where pins VCC, GND, SCL, and SDA wired for I2C communication, and the INT is used to wake the ESP32 from deepsleep mode.

Configuration :

Followed the instruction in the MPU9250 Datasheet and the Register Map. We were able to configure the IMU into Wake-on-motion mode when the ESP32 enter deepsleep mode.

Performance:

The wake-on-motion function works as described in the datasheet.

Calibration:

The sensitivity of the motion sensor range from 0 to 1020 mg. We calibrated the sensitivity to the point where the sensor can be triggered when falling but not regular motion.

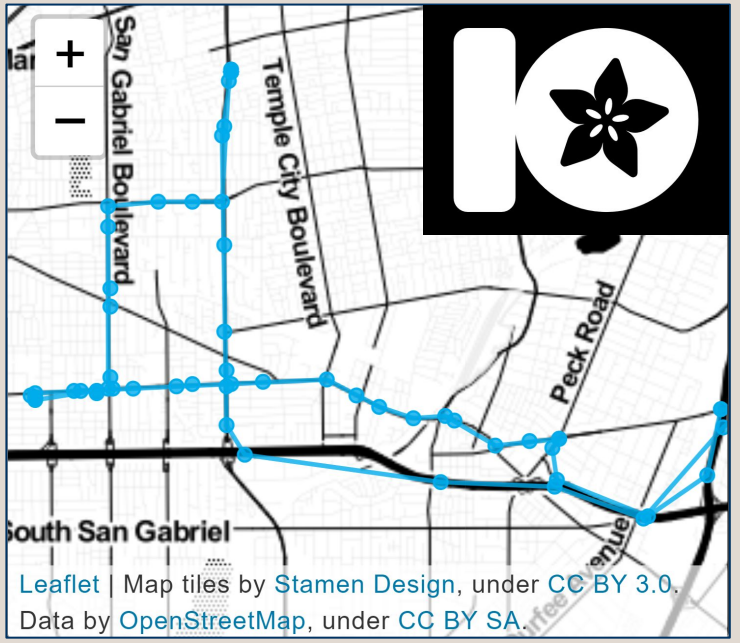
Outputs

Adafruit.io

Access method: via MQTT feeds

Output display:

Adafruit.io (Aio) can make visualized outputs with their Dashboard blocks.



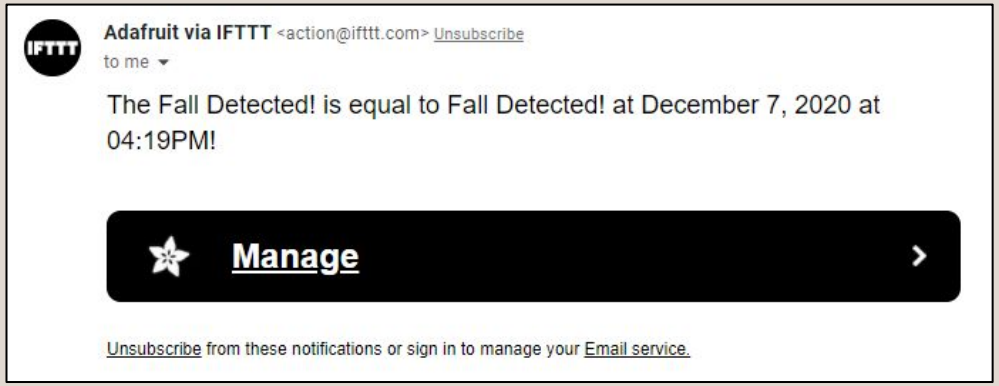
- The Map block plots the historical locations from the feed with timestamps.
- The Stream block monitors all messages updated to different feeds.

IFTTT

- IFTTT trigger will monitor the alert feed from Aio. When the Aio feed received the warning message, IFTTT will send an email to notify the host user.

Created at	Value
2020/12/07 3:09:37PM	Fall Detected!
2020/12/06 6:47:56PM	Fall Detected!
2020/12/05 11:42:50PM	Fall Detected!
2020/12/05 10:22:35PM	Fall detected!

Outputs



Application

- Data feed rate:

  Our device does not require high data feed rate to Aio. One locational data in every 5 minute unless there was an external wake event from the MPU9250.

- Data transfer latency:

  In a “Fall Detected!” event, the ESP32 is waken by the MPU and it will send a warning message to a designated alert feed on Adafruit.io. IFTTT monitors the connected feed and send email notification when the matching message was detected. We think the latency mostly occurs from when IFTTT send the email to the moment when the email account receive the notification.

Data Flow

Wi-Fi

The ESP32 establish personal hotspot connection with the smartphone carried by the user. Locational data is obtained from the Blynk server via the Internet.

MQTT

While the ESP32 is connected to the Internet, messages are communicated with Adafruit.io via MQTT protocol. Aoi MQTT server responds quickly to our message feed and it does not limit our input due to the low data feed rate of our device.

I2C

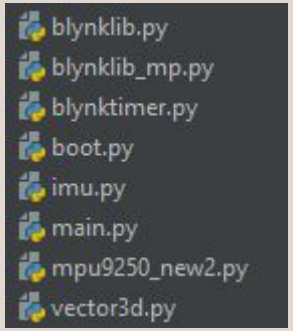
The ESP32 is also connected with the Inertial Measurement Unit (MPU9250). Configuration data is communicated via I2C protocol.

Coding

Language: Micropython (uPy)

Files:

- The device is controlled by boot.py and main.py.
- Uses Blynk uPy library (3 files) to establish the communication with Blynk server.
- Uses imu.py, mpu9250_new2.py, vector3d.py to establish the communication with the IMU.



Main code structure:

- Uses boot.py to establish Wi-Fi connection with smartphone.
- Main.py contains the codes to establish connection to Aio MQTT server, Blynk server, and an infinity loop to callback the main control function.
- Each time the main function is called, the EPS32 will read data from Blynk server. While it will send the data to Aio MQTT server, it will also log the data locally in case of power lost. After the data has been sent, the ESP32 will configure the IMU into wake-on-motion mode then enter Deepsleep mode itself.

- Configuration of the Wake-on-Motion interrupt (Low-power Accel. mode) was done in mpu9250_new2.py. We added a function object to change the register settings of the MPU9250.
- While the MPU is in Wake-on-Motion interrupt mode it can wake the ESP32 from Deepsleep move if the detected acceleration exceeds the threshold acceleration. Then the ESP32 will perform a reset and start the booting routine again.
- A flow control is used during each booting to determine if the ESP32 was waken by the external pin. If yes, the ESP32 will send an alert message to a different feed on the Aio MQTT server.

Power Management:

- ESP32 enters Deepsleep mode right after sending each data.
- MPU9250 operates at Low-power Accel. Mode.
- Device powered by battery.

Challenges:

- Learn Micropython
- Study and implement the Blynk library to use GPS data from the smartphone
- Finding the way to send data to Adafruit.io and have it recognized as GPS coordinates.
- Configure the MPU9250 into Wake-on-Motion mode
- Log data locally
- Merging all codes into main.py and have the device operates in loop.
- Troubleshooting power solution for the device

Summary

Outcome:

- We have all the planned essential functions working as expected
- Excluded “add-on” features such as Geofencing, Speed sensing and Alarm system.
- MPU9250 configuration could be more customizable, but it would require more Micropython programming knowledge .
- The IMU cannot distinguish the falling acceleration and impact. But it turns out impact detection is also a good feature.

Future development:

- Implement the add-on features.
- Modify the MPU9250 configuration function. Make possible for the host user to set sensitivity of the motion sensor from the host end.
- Add SIM module and GPS module to make the device independent from smartphone.
- Develop independent App to include all information and control interface.
- Reduce the size of the device
- Power harvesting from motion.