# IoT - Intelligent and Connected Systems
# EECS E4764 Fall' 22

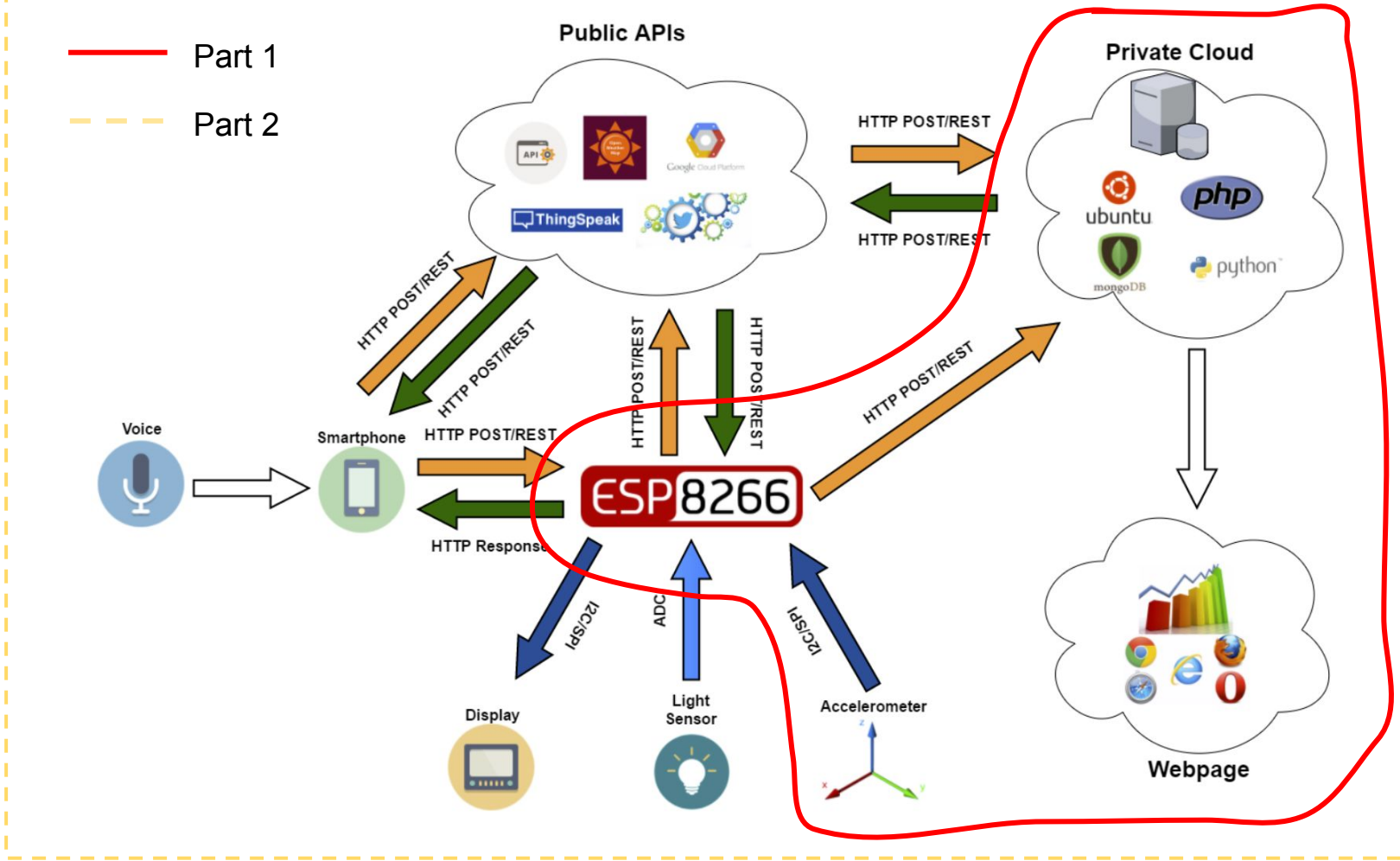
Lab 6

Junyi Wu
jw4173@columbia.edu

# Announcement

Project Proposal and BOM **due tonight at 11:59 pm**
Link / Submissions on *Courseworks - Final Project*
Proposal:

- Per the template on coursework, you should have
  - What? Why?
  - Feature
  - Block diagram: system architecture, HW/SW diagram
  - Component, cost, unit
  - Midpoints check Milestone


Part 1
Part 2
Public APIs
Private Cloud
HTTP POST/REST
HTTP POST/REST
HTTP POST/REST
HTTP POST/REST
HTTP POST/REST
HTTP POST/REST
HTTP POST/REST
HTTP Response
Voice
Smartphone
ESP8266
ThingSpeak
ubuntu
php
mongoDB
python
I2C/SPI
ADC
I2C/SPI
Display
Light Sensor
Accelerometer
Webpage

# 1

## One Last Feature:

## Database and Gesture Recognition

# Part 1

## Setup AWS server and Collect data

- Launch a virtual machine with EC2
- Setup MongoDB on AWS EC2 instance
- Transmit data from the board to EC2 and put it in the database

# Part 2

## Train a classifier for letter recognition

1. Separate your data into training sets and testing sets (e.g., 70%/30%)
2. Pre-process the data (cleaning, normalization, etc.).
3. Select your method (SVM, Neural Network, etc.) and parameters (kernel, layers, etc).
4. Train your classifier on the training sets and test it on the testing sets
5. If the testing result is not good, recollect data or go back to 3

# 2

## Aggregate everything together

# Lab 2 Review

GPIO (Setting Interrupt)


ADC


PWM


PWM

# Lab 3 Review

1. Display the time on the **OLED display**; the time should update like a regular watch. Be able to change time through the OLED display **buttons**.
2. Adjust **screen brightness** based on ambient light.
3. Add **alarm** functionality; Be able to set the alarm through OLED display and when the alarm goes off, a visual and audio notification should go off.

**New**: Now your watch should be able to calibrate time
Automatically given that it's connected to the Internet

# Lab 4 Review

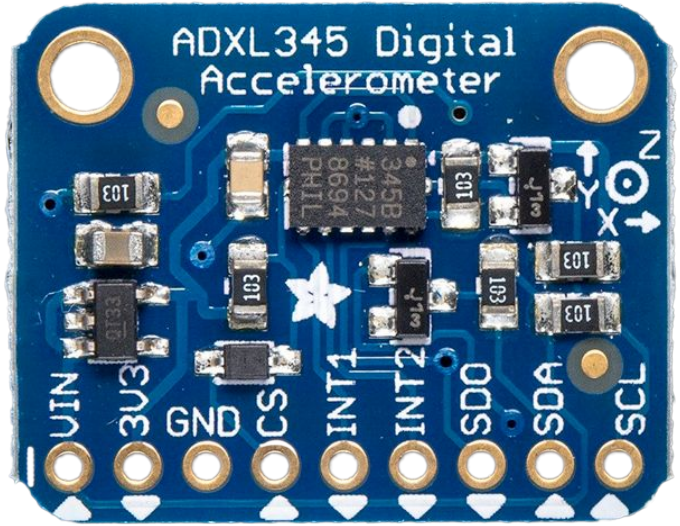
1. Text **scrolling** based on accelerometer readings.
2. Geolocation, weather and twitter **APIs**

# Lab 5 Review

1. Voice control

# Pin allocation

- Use alternate functions of pins
- Connect ADXL345 with I2C instead of SPI
  - I2C bus consists of two signals: SDA and SCL
  - SPI bus consists of four signals
- Turn some function to voice controlled
  - Alarm
- … …



# Part 3 Final integration

Three questions:

1. Does your system meet all function requirements?
2. Is your system robust enough?
3. How to make your system execute efficiently with little resources?

To answer these two questions, you need a good design

A good design needs two things:

1. Operation diagram:
    a. How your system should respond to each input given its state?
    b. How you handle I/O or computational delay and errors in each state?

1. Coding structure:
    b. Consistent while loop? Interrupt-driven design? Composition of both?
    c. Task scheduler
    d. Encapsulation and Modularity
    e. Error/exception handling
    f. Other small issues(variable scopes, garbage collection, etc.)

# Check off

Part 1:

1. Output "COLUMBIA" within 10 tries. (one letter each time)

Part 2:

1. Your system should be able to (we will ask you to perform these operations in random order):
    a. Show the time, set the time, set the alarm, and alarm sound (visual + sound) at the correct time
    b. Receive and execute voice commands from your mobile application
        i. Display weather
        ii. Send spoken tweets
        iii. Display time
    c. Display the weather information
    d. Display the last tweets you sent
    e. Always adjust brightness according to the ambient light
    f. Enter a recognition mode to output letter in the word "COLUMBIA" with your gestures

1. Your system should:
    b. Debounce all the buttons
    c. Be robust and able to handle any potential exceptions

1. You should, as always:
    b. Be able to explain any parts of your code and answer our questions

# Enjoy the Final Lab