TrackMe

CSM117 Project by:
Kirk Patrick Rustia
Amir Ali Omidfar
Keith Akbik

Background and Motivation

TrackMe is a map-based tracking application that allows users to track the location of their belongings, preferably bicycles, as well as the distance between them, in case of conditions like theft. The app has been refined so that it parameters can be updated in real-time.

The motivations behind creating this app were to:

- Create an interactive wireless app easily applicable to everyday issues
- Understand the processes behind geomapping
- Understand the processes behind the interaction of data over a network
- Become knowledgeable in the use of accessing data over a server

Application Interface Setup

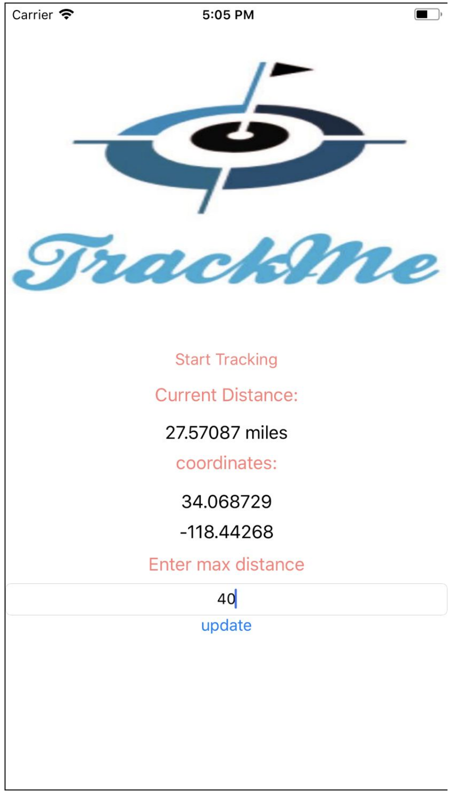
The application contains two different displays: home page and a map.

The home page shows the current distance between the bike and the user, it updates continuously every one second.

The home page shows the recent coordinates of the bike (taken from the server)

Also it allows the user to insert a maximum distance at which he would like to be notified when the bike exceeds this maximum limit

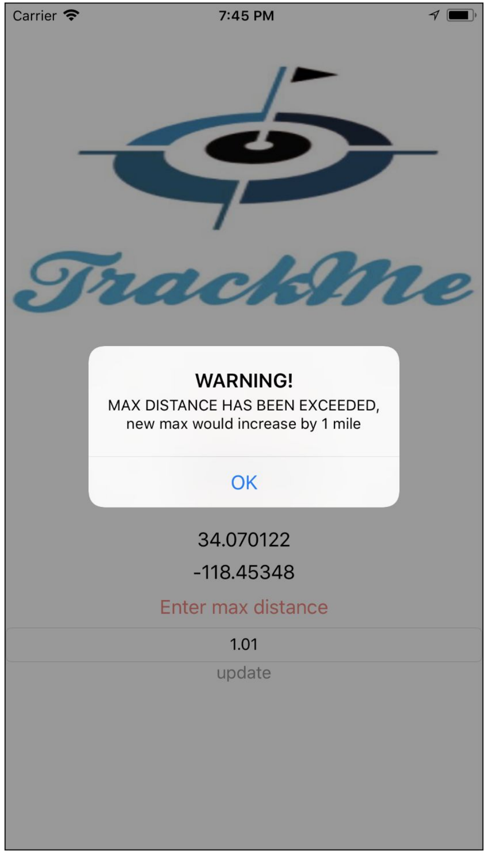
When the user clicks “Start Tracking” he will be shown a map with an annotation that indicates the current location of the bike which updates every second using the coordinates provided from the Arduino to the server.



On the left we see how the user gets notified if the bike has exceeded the preset max, and it asks the user to change the max distance if he think his bike is still within a safe range.

On the right we see how the user has the option to track his bike on the map by clicking "Start Tracking"

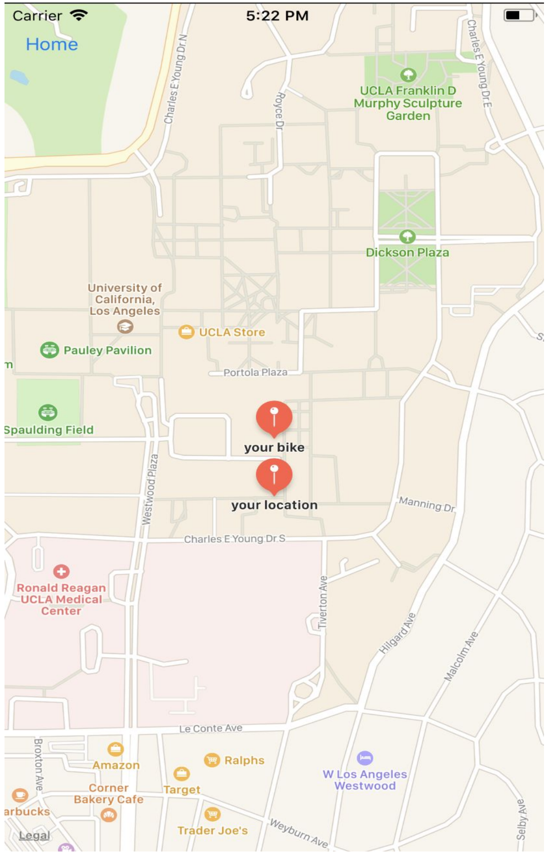
This option will show the bike location along with the user location


Carrier 7:45 PM

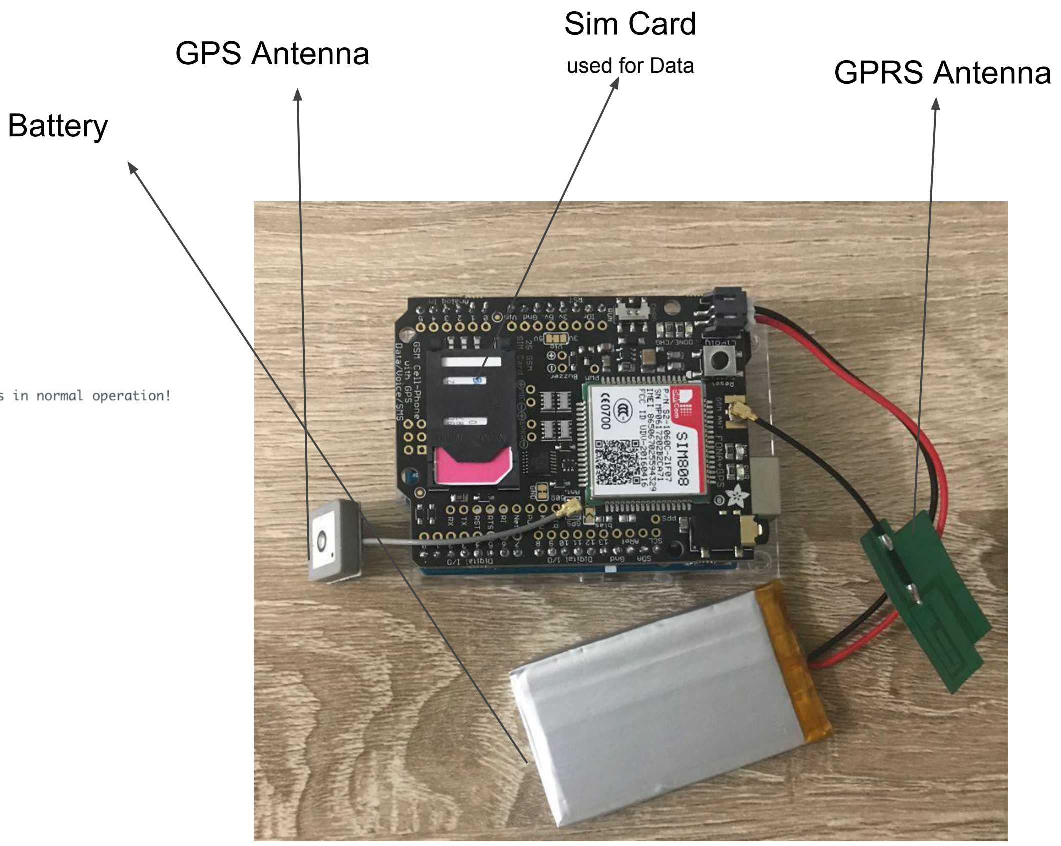
TrackMe

WARNING!
MAX DISTANCE HAS BEEN EXCEEDED, new max would increase by 1 mile

34.070122
-118.45348
Enter max distance
1.01



Hardware Setup (Arduino + Fona 808 Shield)



Arduino Code snippets:

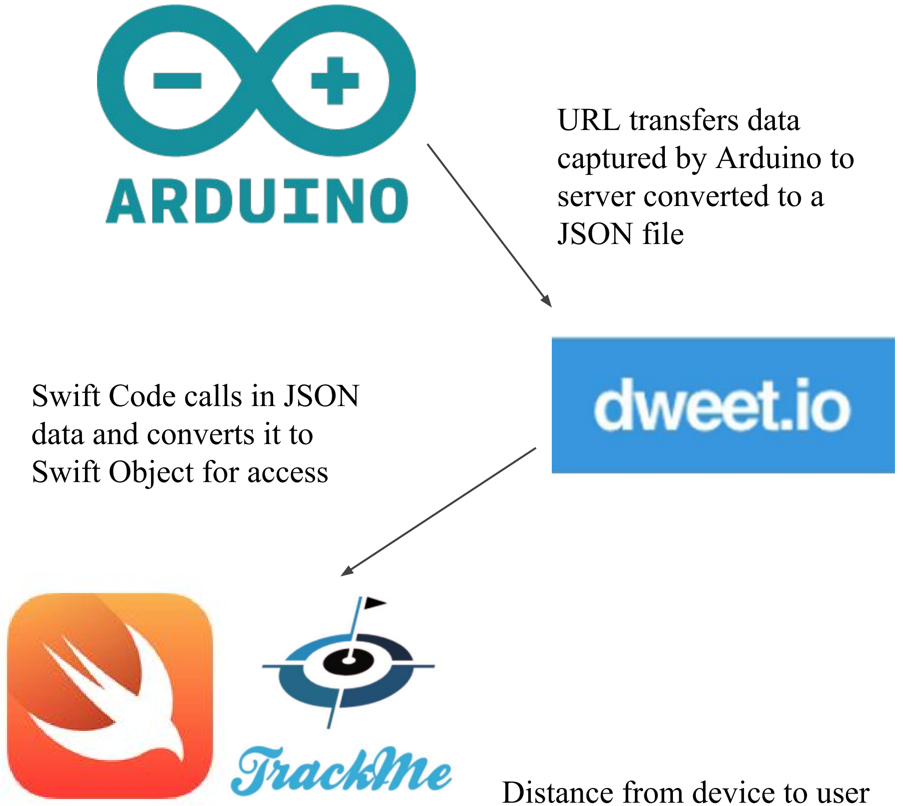
```
  // Libraries
  #include <Adafruit_SleepyDog.h>
  #include "Adafruit_FONA.h"
  #include <SoftwareSerial.h>

void loop() {

  // Watchdog reset at start of loop--make sure everything below takes less than 8 seconds in normal operation!
  //Watchdog.reset();

  // Grab a GPS reading.
  float latitude, longitude, speed_kph, heading, altitude;
  bool gpsFix = fona.getGPS(&latitude, &longitude, &speed_kph, &heading, &altitude);
```

```
// Prepare request
uint16_t statuscode;
int16_t length;
String url = "http://dweet.io/dweet/for/";
url += yourThing;
url += "?latitude=";
url += String(latitude,9);
url += "&longitude=";
url += String(longitude,9);
char buf[80];
url.toCharArray(buf, url.length());
```

Network Setup

The data transfer setup over our desired network framework was done as follows:

1. Using the GPRS/GPS shield of the Arduino, Ethernet connection is used to provide an URL to access the data (latitude and longitude from GPS) from a server.
2. Data sent to a remote server (Dweet) to be ready for access; data is converted over in JSON format.
3. Access data using our app code on Swift; convert data from JSON to readable swift values.
4. Distance between user and designated device displayed on app; map location updated



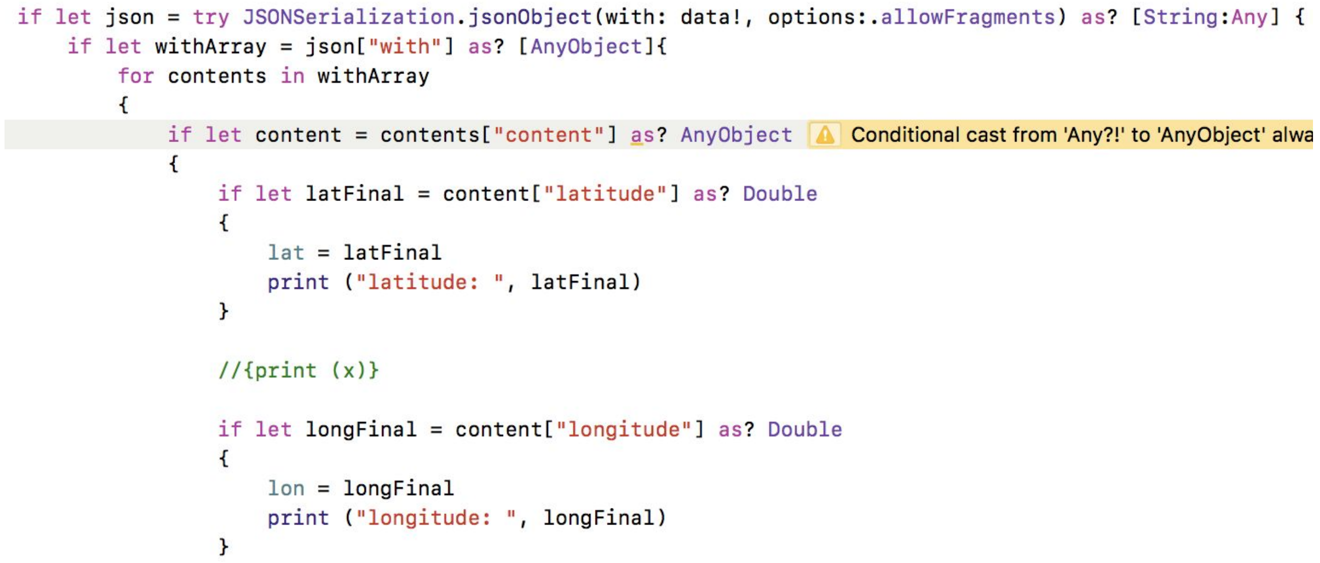
URL transfers data captured by Arduino to server converted to a JSON file

Swift Code calls in JSON data and converts it to Swift Object for access

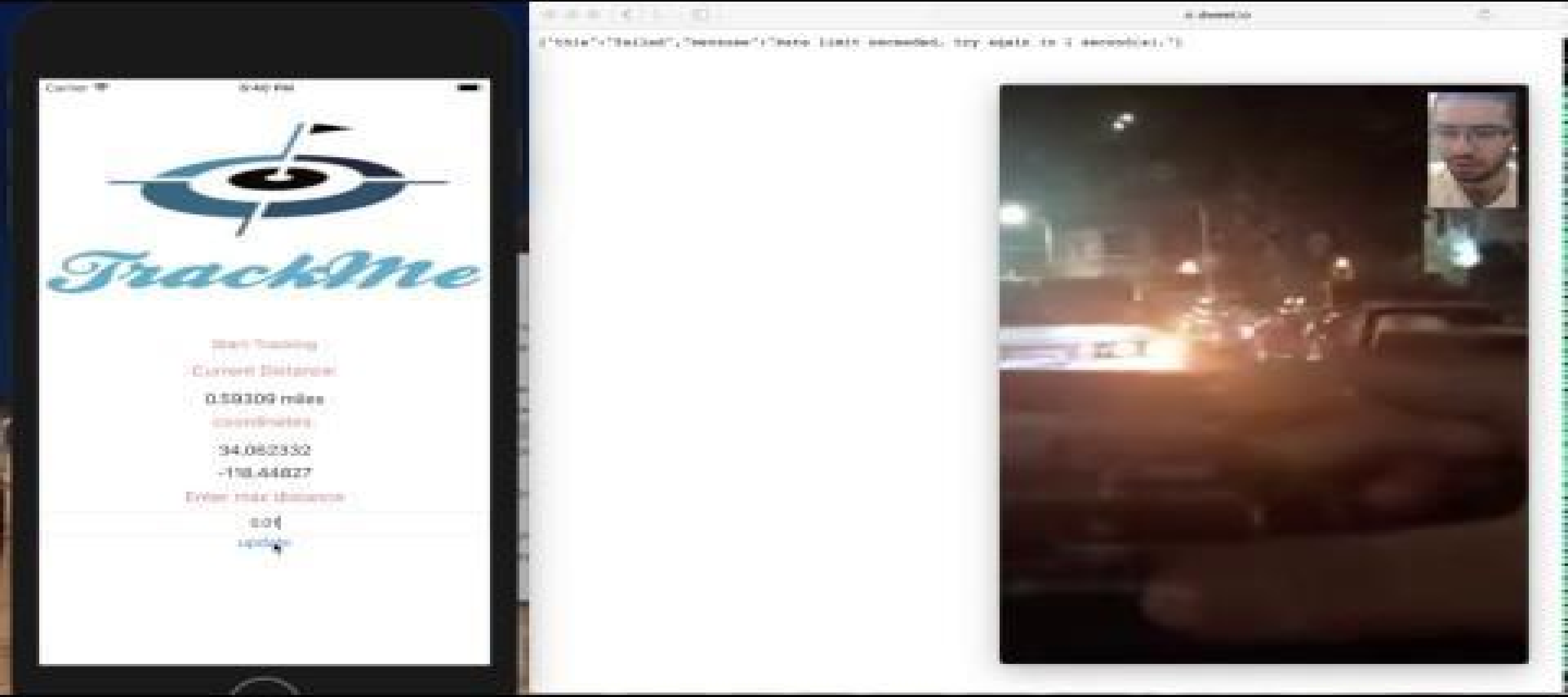
Distance from device to user displayed; map location updated with latitude and longitude update

Below is how the data looks on the server:

{"this":"succeeded","by":"getting","the":"dweets","with":[{"thing":"amirGps","created":"2017-12-04T22:50:07.097Z","content":{"latitude":34.068729,"longitude":-118.44268}}]}



```
if let json = try JSONSerialization.jsonObject(with: data!, options:.allowFragments) as? [String:Any] {
    if let withArray = json["with"] as? [AnyObject]{
        for contents in withArray
        {
            if let content = contents["content"] as? AnyObject   Conditional cast from 'Any?!' to 'AnyObject' alwa
            {
                if let latFinal = content["latitude"] as? Double
                {
                    lat = latFinal
                    print ("latitude: ", latFinal)
                }

                //{print (x)}

                if let longFinal = content["longitude"] as? Double
                {
                    lon = longFinal
                    print ("longitude: ", longFinal)
                }
```



Carrier

TrackMe

Start Tracking

Current Distance:

0.59309 miles

coordinates

34.052332

-118.44827

Enter max distance

0.0

update

```
{"title":"Failed","message":"Rate limit exceeded, try again in 1 second(s)."}
```

Summary of Results

- We were able to track the location of the bike over a constant time interval.
- We were able to show the distance between the bike and the user.
- We were to create an alert system that goes off when the distance surpases a certain threshold.
- Connection and data transfer is very good when working properly.

Challenges

- A team of three EE student doing a CS based project was the hardest challenge for us. Because we had to learn Swift from scratch.
- Finding an easy free online server that is compatible with receiving data directly from Arduino. We had a chance to physically learn how the wireless communication works in terms of server and client relationship as well as how a message is sent and received
- Extracting the data from the server by parsing multiple objects and arrays was not trivial.

Future Improvements and Implementation

- For mass production we need to lower the cost of our module (Arduino + GPS/GPRS shield). Right now being at $50 price range is not really trivial.
- Slightly improve the app interface, to display the route in a better way.
- Enable use for multiple devices for tracking.
- Make the device smaller and more compatible to be used in more often daily life.
- Improve so that the app will work when module is in an indoor environment.

Thank You!

Any Questions?