Project Report:

Real Time GPS Tracking

Group Members:

#	Name	Roll Number	Major
1	Muhammad Hassan Nadeem	15100063	EE
2	Massab Ahmad	15100107	EE
3	Danial Nawaz	15100177	EE
4	Soban Shahid	15100083	EE

Project Objective & Introduction:

The aim of the project is implement industry standard GPS tracking system, which would be capable of real time tracking and plotting the travel history on Google maps.

Components Used:

Component	Quantity
PIC 18F46K22	1
GSM Module (SIM 900D)	1
GPS Module	1
Bluetooth Module	1
Raspberry Pi	1
12 V LiPo Battery	1
5V Voltage Regulator (LM7805)	1
Variable Voltage Regulators (LM317)	2

What did we do?	
1.	Created a website to map the location of the object being tracked.
2.	Created a database to store the location of the object being tracked.
3.	Interfaced the website with Google Maps to display the coordinates on the map.
4.	Interfaced Microcontroller with Google Maps to get direction to specified location.
5.	Wrote php scripts to add records to the database / query records from the database
6.	Interfaced GPS module with the microcontroller.
7.	Parsed GPS sentences and extracted relevant information from it.
8.	Interfaced Bluetooth module with the microcontroller.
9.	Interfaced SIM900D module with the microcontroller.
10.	Send/Receive text messages from the sim module.
11.	Post coordinates to database using GPRS of SIM900D module.
12.	Interfaced Raspberry Pi with the microcontroller.
13.	Wrote routines for the Raspberry Pi so that it could be used in embedded application.
14.	Wrote routines for the Pi so it could replace SIM900D module.
15.	Interfaced Cameras to the Raspberry Pi
16.	Wrote routines for Raspberry Pi to detect the roads.

List of Languages used for this project.	
1.	HTML
2.	CSS
3.	PHP
4.	JavaScript
5.	C++
6.	Python
7.	SQL

Special Features	
1.	Three UARTs were implemented for this project. 1. For the Raspberry Pi/SIM900 Module (Hardware UART) 2. For the Bluetooth (Hardware UART) 3. For the GPS (Software UART)
2.	Hardware UARTs were interrupt driven. i.e everything sent to the UART would be received and processed, so that no exception generated by the SIM900 and Raspberry Pi wouldn't go unnoticed.
3.	Special routines were written to wait and confirm the AT command's responses with a timeout.

4.	No delays in the program….
5.	Internal Pull-Ups of the PIC were used on Port B.
6.	Two external interrupts of the PIC were used. 1. To safely shutdown the Raspberry Pi. 2. To check if the Raspberry was online.
7.	Special routines were written to safely turn off the Raspberry Pi.
8.	Two PWMs were generated using CCP to drive the motors.

Details

In order to make the website we needed access to Google maps so that we could plot the co-ordinates on to it. They we provided free to us under the Google maps API which has a wide array of programming interfaces that enabled us to embed Google maps into our website. The process begins with acquiring the API key which our website should mention in its code to access the maps. There are basically four steps in making the website.

1) Writing Html, CSS, JavaScript and php code for the website.

2) Setting up the database to store the latitude, longitude and time that is sent by the micro-controller through the raspberry pi.

3) Establishing a link between the database and the php script.

4) A link between the database and the raspberry pi through python script.

1) **Setting up the website:**

The website was written in HTML and designed in CSS with embedded JavaScript to access the Google mas through their api. The code in JavaScript consisted of a function named *initialize*() which makes an object of kind map and provides it with specific co-ordinates(latitude and longitude). In order to embed the map onto the website the maps object has to be displayed in a div. Now another function named add() is created to plot the current position i.e the last co-ordinate in the database. The add function simply updates the marker position on the map to show the current location. Since JavaScript is client side language, it cannot access the contents stored in the database. In addition to this, JavaScript is loaded only once at the start and does not reload afterwards. So, in order to dynamically access the database and plot the gps co-ordinate onto the map, AJAX was used. AJAX provided the leverage to get the gps co-ordinate form the database through php and plot it on the map without needing to reload the page. In

addition to this, another function was implemented which accessed all the co-ordinates in the database as an array and plotted then onto the Google maps to provide all the travel history.

2) **Database:**

As mentioned before, a database is needed to save all the co-ordinates for plotting. For this purpose MYSQL was used. MYSQL is a very popular database which runs on a server and used with PHP. All the co-ordinates from the micro-controller are stored in the database and PHP is used to query these co-ordinates and plot pass them onto JavaScript through AJAX. A query is simply a question or a request. The database and the website are uploaded onto the web through a free web hosting service.

3) **Link between Database and website:**

As mentioned before, the link between the website and the database is established through AJAX. AJAX is a group of inter related web development techniques used on client side to create asynchronous web applications.

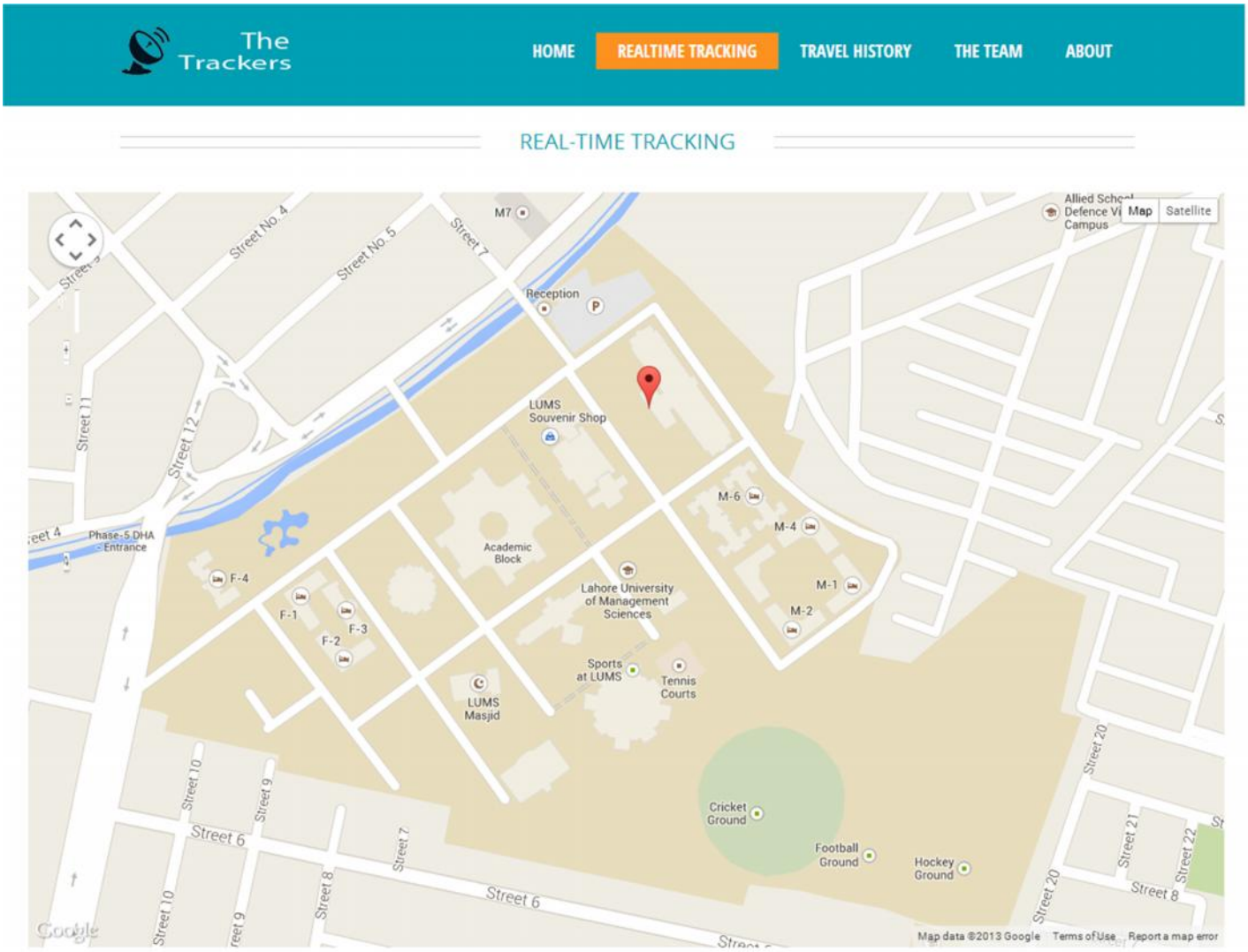
4) **Posting Co-ordinates on the database:**

In order to post the data onto the Internet, the raspberry pi uses python http library. An httpGet() function is created which creates a connection with the database and the uses the https get method to post the data onto the database. The raspberry is turned on by the micro-controller. On startup it runs a python script through crontab which gets co-ordinates from the micro-controller and posts it on the internet.

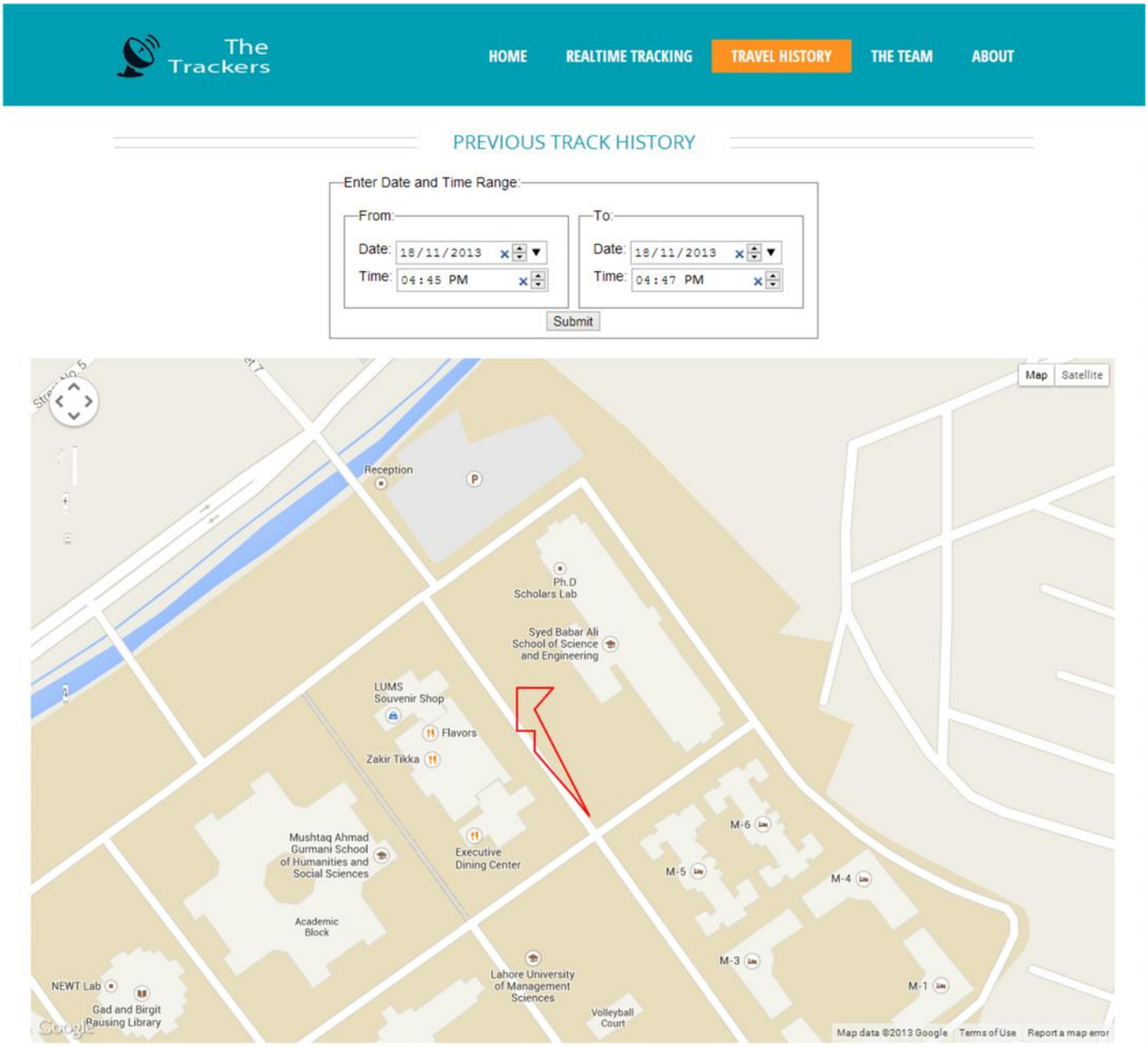
Homepage


The Trackers
HOME
REALTIME TRACKING
TRAVEL HISTORY
THE TEAM
ABOUT
Street 9
Street 11
Street 12
Street 4
Phase-5 DHA - Entrance
MCB Bank ATM
Gloria Jean's Coffees
Academic Block
Lahore University of Management Sciences
Sports at LUMS
Tennis Courts
LUMS Masjid
Suleman Dawood School of Business
F-4
F-3
F-1
F-2
M-6
M-4
M-3
M-2
Street 20
LUMS MICRO-CONTROLLER INTERFACING [EE-324] PROJECT
REAL-TIME GPS TRACKING SYSTEM.
SERVICES
TRACKING
Real Time GPS Tracking
TRAVEL HISTORY
Access to Unlimited Historical Data.
NAVIGATION
Automatic Navigation to specified location.
SALIENT FEATURES
Integrated with Google Maps, see live location of your vehicle.
Lorem ipsum dolor sit amet, consecteturdiing elit. Ut sit amet lorem sit amet nunc mattisrt imperdiet ac sit amet dui
Copyright (C) 2013-15 | Real Time Tracking | Track History | The Team | About
GPS Tracking for LUMS [EE-324] MICRO-CONTROLLER INTERFACING

Real Time Tracking Page, showing position outside SSE Building during project demonstration.



Track History Page, showing route taken during project testing on 18 November 2013.



Issues Faced.

1. Lack of reliable internet connection: Our project depended heavily on stable internet connection. Lack of budget did not allow us to get PTCL EVO Wi-Fi cloud so we had to create our own wifi hotspot using PTCL USB borrowed from a friend through our laptop. The hotspot wasn't perfect and it dropped approximately 15 percent of the connections.
2. Lack of stable web hosting: Due to lack of budget we had to use free web hosting, the website would occasionally go offline. And the database server would deny any php requests if the records exceeded a certain limit.
3. Lack of basic components such as 3.3V voltage regulators, and 3.3V to 5V logic level converters caused significant delay in our project.
4. Bonus part of our project was creating an autonomous robot that could navigate to specified location with the help of Google Maps and computer vision. In order to navigate, the robot needed to detect the roads so that it would stay in the center of the road and to detect turns. Roads in LUMS had no identifying features like lanes, so it was particularly difficult to keep track of the road. Our initial algorithm was based on sensing the elevated footpath; we spend a lot of time on that algorithm but although it was quite good, it wasn't enough for navigation (because the footpath was of the same color as the road, and there was no room for errors in a project like this). We solved this problem by adding Road surface marking ourselves with yellow colored packing tape. The problem of detecting the road was solved but it turned out that the height of our robot chassis was too low and the cameras mounted on the surface of the robot couldn't see the markings. By this it was already too late, so we abandoned this part of the project.

Block Diagram(s):