8 Appendices

## Appendix A: Installation Guide

BusTracker

Department of Electrical Engineering - Senior Design

University of Notre Dame

***Installation Guide***

Table of Contents

[Overview of Parts and Connections 1](#_Toc450075059)

[Installation of BusTracker 3](#_Toc450075060)

[Installation on Gillig Bus (Older Bus) 3](#_Toc450075061)

[Installation on New Flyer Bus (Newer Bus) 4](#_Toc450075062)

[Setup of BusTracker 6](#_Toc450075063)

[Verify Operation of BusTracker 6](#_Toc450075064)

[Troubleshooting 7](#_Toc450075065)

[Troubleshooting BusTracker Box 7](#_Toc450075066)

[Troubleshooting BusTracker App 7](#_Toc450075067)

[Customer Support 7](#_Toc450075068)

Overview of Parts and Connections

The major components of the BusTracker system is shown in Figure 1 below.



Power Connection

GPS Antenna

Cellular Antenna

Figure 1. Major BusTracker Components.

The inside of the BusTracker box is shown in Figure 2 below with the Indicator LED highlighted. The driver should never need to open the BusTracker box except for troubleshooting. The BusTracker box should not be opened while the bus is in operation by the driver.



Figure 2. Inside of BusTracker Box, LED circled

Installation of BusTracker

The BusTracker is currently developed specifically for the South Bend Transpo buses (expansion beyond Transpo is possible but not currently implemented). There are currently two different bus models driving Transpo bus routes - the 2004 Gillig and the 2014 New Flyer.. Installation and placement of the BusTracker differs slightly depending on the model. Installation on a Transpo bus requires a slightly different power connector since 12V cigarette lighter car jacks are not available (utilized during personal vehicle demonstration). One end of the power cable for the bus should have a 2.1mm barrel jack connector. The other end of the cable will be stripped so that it can be tapped to a 12V line connection on the bus. The bus, regardless of model, should not be on during installation.

# Installation on Gillig Bus (Older Bus)

On the Gillig model, the BusTracker will be mounted on top of the front dashboard next to the Motorola radio box. Figure **2** below shows the placement location of the BusTracker with a red circle.



Figure **2**. BusTracker location on Gillig

The power cable for the BusTracker should follow the same path as the Motorola radio box as shown in Figure **3** below.



Figure **3.** Power line connection to follow

The other end of the power cable should tap into one of the 12V lines behind left side panel of the driver's seat. The GPS antenna of the BusTracker should be affixed nearby on the dashboard without any objects blocking the antenna to the window.

# Installation on New Flyer Bus (Newer Bus)

On the New Flyer model, the BusTracker will be mounted in the top right corner of the bus as seen from the driver's point of view. The BusTracker cannot be placed on the dashboard as on the Gillig bus because there are no 12V line connections nearby to tap power from. The GPS antenna will be affixed nearby along the top panel so that it will not interfere with the driver's line of sight. The mounting location for the BusTracker is shown in Figure **4** below.



Figure **4.** Mounting location on New Flyer

The top panel of the bus also contains 12V power lines as seen in Figure **5** below. This is where the power cable of the BusTracker will tap into for power. The connections should be similar to the Motorola radio box connections above the driver's seat on the left side of the bus.



Figure **5.** Inside of top panel of New Flyer bus with power lines

Setup of BusTracker

Before testing the BusTracker, an account must be setup with Telit. The chip that contains the GPS receiver and cellular connector is produced by Telit. Telit also maintains the deviceWise online server where information about connected BusTrackers and GPS data is stored. Transpo operators should contact Telit to register for a deviceWise account and AT&T connectivity data plan.

The BusTracker is programmed to automatically set itself up once it receives power. When a bus is turned on, power from the bus's 12V line will start up the BusTracker. The onboard LED of the BusTracker should start blinking once it is fully operating. The BusTracker will connect to the cellular network as start transmitting current GPS positions to the deviceWise server. The process will take, at most, two minutes. There is no setup required on the driver's side.

To download the Bus Tracker app, XCode must be installed on a Mac computer. Download the code from the Box folder and run the app on a connected iPhone.

# Verify Operation of BusTracker

Operation of the BusTracker can be verified by two different methods. Transpo operators can login to the deviceWISE server and observe the locations of entire bus fleet that is currently running. Bus riders can also download the BusTracker app and see the current location of the bus in question.

Troubleshooting

# Troubleshooting BusTracker Box

* If the BusTracker system no longer updates the location of the bus, check the BusTracker box to verify that the LED is blinking, which signifies the tracking module being correctly turned on.
* Check all antenna connectors to verify that neither the cellular nor GPS antenna has come unconnected. If the location of the bus does not seem to be changing, check the mounting location of the GPS antenna as it may have lost its satellite fix.
* If the board does not seem to be powering on at all verify that the Telit miniPCIe adapter on the board has not come loose from its connector.
* Verify that the BusTracker has not accidently been connected to the 24V power line.

# Troubleshooting BusTracker App

* If the current bus location does not seem to be changing, try switching back and forth between your current stop in order to refresh the bus location.
* Close and restart the app in case of crashes.

# Customer Support

For BusTracker box hardware issues, contact Christine Joseph at cjoseph1@nd.edu or Grant Weber at gweber2@nd.edu

For Telit account and connectivity questions, contact Grant Weber at gweber2@nd.edu

For questions regarding the BusTracker app download and code, contact SeungGoo Kang at skang2@nd.edu or Racine Hansen at rhansen3@nd.edu

## Appendix B: Microcontroller Code to Setup UARTs and Send and Parse GPS AT Commands

/\*

\* File: main.c

\*/

#include <xc.h>

#include "configbits-16ex8.h"

#include "string.h"

#define u2\_buff\_full U2STAbits.UTXBF

#define u2\_buff\_fullRX U2STAbits.URXDA

#define u1\_buff\_fullRX U1STAbits.URXDA

#define GetSystemClock() (40000000ul)

#define GetPeripheralClock() (GetSystemClock())

#define BUFMAX 128

char gps\_string[BUFMAX];

float lat, lon, lat1, lon1;

int fix, qual, dd;

char \*start\_ptr, \*end\_ptr, \*latitude, \*longitude, \*quality;

//initialize UART1

void serial\_init1(unsigned long rate) //rate = desired baud rate

{

U2MODEbits.ON = 1; //enable UART1

U2MODEbits.BRGH = 1; //baud rate

U2MODEbits.UEN = 0b00; //enable bits

U2MODEbits.PDSEL = 0b00; //8 bit no parity

U2MODEbits.STSEL = 0; //1 stop bit

U2STAbits.URXEN = 1; //enable receiver

U2STAbits.UTXEN = 1; //enable transmitter

U2BRG = ((GetSystemClock())/(4\*rate))-1;

return;

}

//initialize UART2

void serial\_init2(unsigned long rate) //rate = desired baud rate

{

U1MODEbits.ON = 1; //enable UART1

U1MODEbits.BRGH = 1; //baud rate

U1MODEbits.UEN = 0b00; //enable bits

U1MODEbits.PDSEL = 0b00; //8 bit no parity

U1MODEbits.STSEL = 0; //1 stop bit

U1STAbits.URXEN = 1; //enable receiver

U1STAbits.UTXEN = 1; //enable transmitter

U1BRG = ((GetSystemClock())/(4\*rate))-1;

return;

}

void setup\_pps(void) {

SYSKEY = 0x0;

SYSKEY = 0xAA996655;

SYSKEY = 0x556699AA;

CFGCONbits.IOLOCK = 0;

U2RXRbits.U2RXR = 0; //pin 3 rpa1

RPA3Rbits.RPA3R = 2; //pin 10 rpa3

U1RXRbits.U1RXR = 2; //pin12 rpa4

RPB4Rbits.RPB4R = 1; //pin 11 rpb4

CFGCONbits.IOLOCK = 1;

SYSKEY = 0x0;

}

//transmit function for UART1

int SerialTransmitU1(const char \*buffer)

{

unsigned int size = strlen(buffer);

while( size)

{

while( U1STAbits.UTXBF); // wait while TX buffer full

U1TXREG = \*buffer; // send single character to transmit buffer

buffer++; // transmit next character on following loop

size--; // loop until all characters sent (when size = 0)

}

while( !U1STAbits.TRMT); // wait for last transmission to finish

return 0;

}

//transmit function for UART2

int SerialTransmitU2(const char \*buffer)

{

unsigned int size = strlen(buffer);

while( size)

{

while( U2STAbits.UTXBF); // wait while TX buffer full

U2TXREG = \*buffer; // send single character to transmit buffer

buffer++; // transmit next character on following loop

size--; // loop until all characters sent (when size = 0)

}

while( !U2STAbits.TRMT); // wait for last transmission to finish

return 0;

}

//receive function for UART2

unsigned int SerialReceiveU2(char \*buffer, unsigned int max\_size)

{

unsigned int num\_char = 0;

/\* Wait for and store incoming data until either a carriage return is received

\* or the number of received characters (num\_chars) exceeds max\_size \*/

while(num\_char < max\_size)

{

while( !U2STAbits.URXDA); // wait until data available in RX buffer

\*buffer = U2RXREG; // empty contents of RX buffer into \*buffer pointer

// insert nul character to indicate end of string

if( \*buffer == '\r'){

\*buffer = '\0';

break;

}

buffer++;

num\_char++;

}

return num\_char;

} // END SerialReceive()

//receive function for UART1

unsigned int SerialReceiveU1(char \*buffer, unsigned int max\_size)

{

unsigned int num\_char = 0;

/\* Wait for and store incoming data until either a carriage return is received

\* or the number of received characters (num\_chars) exceeds max\_size \*/

while(num\_char < max\_size)

{

while( !U1STAbits.URXDA); // wait until data available in RX buffer

\*buffer = U1RXREG; // empty contents of RX buffer into \*buffer pointer

// insert nul character to indicate end of string

if( \*buffer == '\r'){

\*buffer = '\0';

break;

}

buffer++;

num\_char++;

}

return num\_char;

} // END SerialReceive()

//custom delay function

void myDelay(int delay){

//delay of 1 second = 3200000

unsigned int count;

for( count = 0; count < delay; count++) {}

}

void putu1(unsigned char val) // put char

{

\_mon\_putc(val);

return;

}

void myprintf(char S[]){

int length, i;

length = strlen(S);

for(i=0;i<length;i++){

putu1(S[i]);

}

}

//AT commands to start getting GPS data

void initalizeGPS(){

myprintf("AT\t");

myDelay(32000);

myprintf("AT$GPSRST\t");

myDelay(32000);

myprintf("AT$GPSNVRAM=15,0\t");

myDelay(32000);

myprintf("AT$GPSACP\t");

myDelay(32000);

}

void getGPSstr(){

char letter = SerialReceiveU1();

int i = 1;

while(letter != '$'){

letter = SerialReceiveU1();

}

gps\_string[0] = letter;

while(letter != '\*'){

letter = SerialReceiveU1();

gps\_string[i] = letter;

i++;

}

}

//function to parse NMEA sentence

void parse(volatile unsigned char \*string)

{

printf("%s\n", gps\_string);

// LATITUDE

start\_ptr=strchr(gps\_string,','); //find start of time field

start\_ptr=strchr(start\_ptr+1,','); //find start of latitude field

latitude = start\_ptr+1; //find first character in latitude field

printf ("Latitude starts at gps\_string[%d]\n",latitude-gps\_string);

end\_ptr=strchr(latitude,','); //find end of latitude field

printf ("Latitude ends at gps\_string[%d]\n",end\_ptr-gps\_string-1);

if (latitude-gps\_string>=end\_ptr-gps\_string) //determine if latitude exists

{printf("Latitude not received\n");}

else

{

lat=atof(latitude); //convert to float

printf("Latitude = %f\n",lat);

}

// LATITUDE

// LONGITUDE

start\_ptr=strchr(start\_ptr+1,',');

start\_ptr=strchr(start\_ptr+1,',');

longitude = start\_ptr+1;

printf ("Longitude starts at gps\_string[%d]\n",longitude-gps\_string);

end\_ptr=strchr(longitude,',');

printf ("Longitude ends at gps\_string[%d]\n",end\_ptr-gps\_string-1);

if (longitude-gps\_string>=end\_ptr-gps\_string)

{printf("Longitude not received\n");}

else

{

lon=atof(longitude);

printf("Longitude = %f\n",lon);

}

// LONGITUDE

// QUALITY

start\_ptr=strchr(start\_ptr+1,',');

start\_ptr=strchr(start\_ptr+1,',');

quality = start\_ptr+1;

printf ("Quality starts at gps\_string[%d]\n",quality-gps\_string);

end\_ptr=strchr(quality,',');

printf ("Quality ends at gps\_string[%d]\n",end\_ptr-gps\_string-1);

if (quality-gps\_string>=end\_ptr-gps\_string)

{

printf("Quality not received\n");

fix = 0;

//RB1=0;

}

else

{

qual=atoi(quality);

printf("Quality = %d\n",qual);

if (qual==1|qual==2){

printf("Fix acquired\n");

fix = 1; //if fix acquired, turn GPS LED on

//RB1=1;

}

else{

printf("Fix not aquired\n");

fix=0; //if fix not acquired, turn GPS LED off

//RB1=0;

}

}

}

/\*

\* Program to flash the LED's

\*/

int main(int argc, char\*\* argv)

{

//DEVCFG0bits.ICESEL = 1;

//OSCCONbits.FRCDIV = 5;

//OSCCONbits.COSC = 0b101;

//CFGCONbits.JTAGEN = 0;

TRISACLR = 0xFFFF;

TRISBCLR = 0xFFFF;

ANSELACLR = 0xFFFF;

ANSELBCLR = 0xFFFF;

TRISAbits.TRISA3 = 0; //tx output

TRISAbits.TRISA1 = 1; //rx input

TRISAbits.TRISA4 = 1; //rx1 input

TRISBbits.TRISB4 = 0; //tx1 output

TRISAbits.TRISA2 = 0;

setup\_pps();

serial\_init1(9600);

serial\_init2(9600);

initalizeGPS();

//parse();

return (EXIT\_SUCCESS);

}

## Appendix C: Mobile App View Controller Code

import UIKit

import CoreLocation

import PXGoogleDirections

import GoogleMaps

import CocoaMQTT

var originField = CLLocationCoordinate2D(latitude:41.669419,longitude:-86.250977)

var destinationField = CLLocationCoordinate2D(latitude: 41.659131, longitude: -86.180606)

var destinationScroll = 0

var date = NSDate()

var formatter = NSDateFormatter()

var fromLatitude:Double = 0

var fromLongitude:Double = 0

var toLatitude:Double = 0

var toLongitude:Double = 0

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MAIN VIEW CONTROLLER

- not used

- Seung Goo disabled in storyboard

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class MainViewController: UIViewController, CLLocationManagerDelegate, UIPickerViewDelegate{

var startArriveDate: NSDate?

var waypoints: [PXLocation] = [PXLocation]()

var mqtt: CocoaMQTT?

// var originField = CLLocationCoordinate2D(latitude:41.669419,longitude:-86.250977)

override func viewDidAppear(animated: Bool) {

super.viewDidAppear(animated)

directionsAPI.delegate = self

directionsAPI.from = PXLocation.CoordinateLocation(originField)

directionsAPI.to = PXLocation.CoordinateLocation(destinationField)

directionsAPI.mode = modeFromField()

directionsAPI.departureTime = .Now

// WaypointViewController()

calculateRoute()

}

private var directionsAPI: PXGoogleDirections {

return (UIApplication.sharedApplication().delegate as! AppDelegate).directionsAPI

}

private func modeFromField() -> PXGoogleDirectionsMode {

return PXGoogleDirectionsMode(rawValue: 0)!

}

private func WaypointViewController(){

let lat = 41.679211

let long = -86.235158

var waypoint: PXLocation?

waypoint = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat, long))

waypoints.append(waypoint!)

let lat2 = 41.680218

let long2 = -86.216060

var waypoint2: PXLocation?

waypoint2 = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat2, long2))

waypoints.append(waypoint2!)

let lat3 = 41.672990

let long3 = -86.196518

var waypoint3: PXLocation?

waypoint3 = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat3, long3))

waypoints.append(waypoint3!)

let lat4 = 41.659131

let long4 = -86.180606

var waypoint4: PXLocation?

waypoint4 = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat4, long4))

waypoints.append(waypoint4!)

directionsAPI.waypoints = waypoints

}

private func calculateRoute(){

directionsAPI.calculateDirections { (response) -> Void in

dispatch\_async(dispatch\_get\_main\_queue(), { () -> Void in

switch response {

case let .Error(\_, error):

let alert = UIAlertController(title: "PXGoogleDirectionsSample", message: "Error: \(error.localizedDescription)", preferredStyle: UIAlertControllerStyle.Alert)

alert.addAction(UIAlertAction(title: "OK", style: .Default, handler: nil))

self.presentViewController(alert, animated: true, completion: nil)

case let .Success(request, routes):

if let rvc = self.storyboard?.instantiateViewControllerWithIdentifier("Results") as? ResultsViewController {

rvc.request = request

rvc.results = routes

self.presentViewController(rvc, animated: true, completion: nil)

}

}

})

}

}

}

var cnt:Double = 0

var cnt2:Int = 0

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

RESULTS VIEW CONTROLLER

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class ResultsViewController: UIViewController, CLLocationManagerDelegate, GMSMapViewDelegate, UIPickerViewDelegate{

@IBOutlet weak var mqttTestLabel: UILabel!

@IBOutlet weak var pickerStop: UIPickerView!

@IBOutlet weak var routesLabel: UILabel!

@IBOutlet weak var mapView: GMSMapView!

@IBOutlet weak var directions: UITableView!

var request: PXGoogleDirections!

var results: [PXGoogleDirectionsRoute]!

var routeIndex: Int = 0

var waypoints: [PXLocation] = [PXLocation]()

var mqtt: CocoaMQTT?

//var originField = CLLocationCoordinate2D(latitude:41.669419,longitude:-86.250977)

@IBOutlet weak var arriveTime: UILabel!

private var directionsAPI: PXGoogleDirections {

return (UIApplication.sharedApplication().delegate as! AppDelegate).directionsAPI

}

private func modeFromField() -> PXGoogleDirectionsMode {

return PXGoogleDirectionsMode(rawValue: 0)!

}

//PICKER VIEW

var bstop = ["LaSelle & Eddy (Outbound)","McKinley & Ironwood (Outbound)","Jefferson & Logan (Outbound)","Mishawaka Transfer Center(Outbound)", "Jefferson & Logan (Inbound)","McKinley & Ironwood (Inbound)","LaSelle & Eddy (Inbound)"]

public func numberOfComponentsInPickerView(pickerView: UIPickerView) -> Int

{

return 1

}

func pickerView(pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int

{

return bstop.count

}

func pickerView(pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String?

{

return bstop[row]

}

func pickerView(pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int)

{

let calendar = NSCalendar.currentCalendar()

let components = calendar.components([.Hour, .Minute], fromDate: date)

let cHour = components.hour

let cMin = components.minute

var leftRight = 0

// Right = 1 , Left = 0

waypoints.removeAll()

if cHour >= 5 && cHour <= 21

{

if cMin >= 20 && cMin <= 44

{

leftRight = 1

}

if cMin > 44

{

leftRight = 0

}

}

if cHour >= 6 && cHour <= 22

{

if cMin <= 12

{

leftRight = 0

}

}

leftRight = 1

var zone = 0

if originField.longitude < -86.235250

{

zone = 1

}

if originField.longitude >= -86.235250 && originField.longitude < -86.216060

{

zone = 2

}

if originField.longitude >= -86.216060 && originField.longitude < -86.196518

{

zone = 3

}

if originField.longitude >= -86.196518

{

zone = 4

}

let lat = 41.679211

let long = -86.235158

var waypoint: PXLocation?

waypoint = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat, long))

// waypoints.append(waypoint!)

let lat2 = 41.680218

let long2 = -86.216060

var waypoint2: PXLocation?

waypoint2 = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat2, long2))

// waypoints.append(waypoint2!)

let lat3 = 41.672990

let long3 = -86.196518

var waypoint3: PXLocation?

waypoint3 = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat3, long3))

// waypoints.append(waypoint3!)

let lat4 = 41.659131

let long4 = -86.180606

var waypoint4: PXLocation?

waypoint4 = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat4, long4))

// waypoints.append(waypoint4!)

mapView.clear()

var bstopSelected = bstop[row]

if bstopSelected == "LaSelle & Eddy (Outbound)"{

destinationField = CLLocationCoordinate2DMake(41.679211, -86.235158)

destinationScroll = 0

}

if bstopSelected == "McKinley & Ironwood (Outbound)"{

destinationField = CLLocationCoordinate2DMake(41.680218,-86.216060)

destinationScroll = 1

if zone == 1 && leftRight == 1

{

waypoints.append(waypoint!)

}

}

if bstopSelected == "Jefferson & Logan (Outbound)"{

destinationField = CLLocationCoordinate2DMake(41.672990,-86.196518)

if zone == 1 && leftRight == 1

{

waypoints.append(waypoint!)

waypoints.append(waypoint2!)

}

if zone == 2 && leftRight == 1

{

waypoints.append(waypoint2!)

}

destinationScroll = 2

}

if bstopSelected == "Mishawaka Transfer Center(Outbound)"{

destinationField = CLLocationCoordinate2DMake(41.659131,-86.180606)

if zone == 1 && leftRight == 1

{

waypoints.append(waypoint!)

waypoints.append(waypoint2!)

waypoints.append(waypoint3!)

}

if zone == 2 && leftRight == 1

{

waypoints.append(waypoint2!)

waypoints.append(waypoint3!)

}

if zone == 3 && leftRight == 1

{

waypoints.append(waypoint3!)

}

destinationScroll = 3

}

if bstopSelected == "Jefferson & Logan (Inbound)"{

destinationField = CLLocationCoordinate2DMake(41.672990,-86.196518)

if zone == 1 && leftRight == 1

{

waypoints.append(waypoint!)

waypoints.append(waypoint2!)

waypoints.append(waypoint3!)

waypoints.append(waypoint4!)

}

if zone == 2 && leftRight == 1

{

waypoints.append(waypoint2!)

waypoints.append(waypoint3!)

waypoints.append(waypoint4!)

}

if zone == 3 && leftRight == 1

{

waypoints.append(waypoint3!)

waypoints.append(waypoint4!)

}

if zone == 4 && leftRight == 1

{

waypoints.append(waypoint4!)

}

destinationScroll = 4

}

if bstopSelected == "McKinley & Ironwood (Inbound)"{

destinationField = CLLocationCoordinate2DMake(41.680218,-86.216060)

if zone == 1 && leftRight == 1

{

waypoints.append(waypoint!)

waypoints.append(waypoint2!)

waypoints.append(waypoint3!)

waypoints.append(waypoint4!)

waypoints.append(waypoint3!)

}

if zone == 2 && leftRight == 1

{

waypoints.append(waypoint2!)

waypoints.append(waypoint3!)

waypoints.append(waypoint4!)

waypoints.append(waypoint3!)

}

if zone == 3 && leftRight == 1

{

waypoints.append(waypoint3!)

waypoints.append(waypoint4!)

waypoints.append(waypoint3!)

}

if zone == 4 && leftRight == 1

{

waypoints.append(waypoint4!)

waypoints.append(waypoint3!)

}

if zone == 4 && leftRight == 0

{

waypoints.append(waypoint3!)

}

destinationScroll = 5

}

if bstopSelected == "LaSelle & Eddy (Inbound)"{

destinationField = CLLocationCoordinate2DMake(41.679211, -86.235158)

if zone == 1 && leftRight == 1

{

waypoints.append(waypoint!)

waypoints.append(waypoint2!)

waypoints.append(waypoint3!)

waypoints.append(waypoint4!)

waypoints.append(waypoint3!)

waypoints.append(waypoint2!)

}

if zone == 2 && leftRight == 1

{

waypoints.append(waypoint2!)

waypoints.append(waypoint3!)

waypoints.append(waypoint4!)

waypoints.append(waypoint3!)

waypoints.append(waypoint2!)

}

if zone == 3 && leftRight == 1

{

waypoints.append(waypoint3!)

waypoints.append(waypoint4!)

waypoints.append(waypoint3!)

waypoints.append(waypoint2!)

}

if zone == 4 && leftRight == 1

{

waypoints.append(waypoint4!)

waypoints.append(waypoint3!)

waypoints.append(waypoint2!)

}

if zone == 4 && leftRight == 0

{

waypoints.append(waypoint3!)

waypoints.append(waypoint2!)

}

if zone == 3 && leftRight == 0

{

waypoints.append(waypoint2!)

}

destinationScroll = 6

}

mqttSetting()

directionsAPI.from = PXLocation.CoordinateLocation(originField)

directionsAPI.to = PXLocation.CoordinateLocation(destinationField)

directionsAPI.mode = modeFromField()

directionsAPI.departureTime = .Now

directionsAPI.waypoints = waypoints

// mqttTestLabel.text = String(originField.latitude)

date = NSDate()

let marker = GMSMarker()

marker.position = originField

marker.title = "Bus"

marker.map = mapView

marker.icon = UIImage(named:"bus.png")

stopMark()

//

// viewDidLoad()

calculateRoute()

updateRoute()

}

override func viewDidLoad() {

super.viewDidLoad()

mqttSetting()

mapView.delegate = self

// directionsAPI.delegate = self

directionsAPI.from = PXLocation.CoordinateLocation(originField)

directionsAPI.to = PXLocation.CoordinateLocation(destinationField)

directionsAPI.mode = modeFromField()

directionsAPI.departureTime = .Now

// directionsAPI.waypoints = waypoints

pickerStop.selectRow(destinationScroll, inComponent: 0, animated: true)

let marker = GMSMarker()

marker.position = originField

marker.title = "Bus"

marker.map = mapView

marker.icon = UIImage(named:"bus.png")

stopMark()

}

func stopMark(){

//Stop1

let marker1 = GMSMarker()

marker1.position = CLLocationCoordinate2DMake(41.669419, -86.250977)

marker1.title = "South Bend Station"

marker1.map = mapView

//Stop2

let marker2 = GMSMarker()

marker2.position = CLLocationCoordinate2DMake(41.679211, -86.235158)

marker2.title = "LaSelle & Eddy"

marker2.map = mapView

//Stop3

let marker3 = GMSMarker()

marker3.position = CLLocationCoordinate2DMake(41.680218,-86.216060)

marker3.title = "McKinley & Ironwood"

marker3.map = mapView

//Stop4

let marker4 = GMSMarker()

marker4.position = CLLocationCoordinate2DMake(41.672990,-86.196518)

marker4.title = "Jefferson & Logan"

marker4.map = mapView

//Stop5

let marker5 = GMSMarker()

marker5.position = CLLocationCoordinate2DMake(41.659131,-86.180606)

marker5.title = "Mishawaka Transfer Center"

marker5.map = mapView

}

private func WaypointViewController(){

let lat = 41.679211

let long = -86.235158

var waypoint: PXLocation?

waypoint = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat, long))

waypoints.append(waypoint!)

let lat2 = 41.687114

let long2 = -86.238023

var waypoint2: PXLocation?

waypoint2 = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat2, long2))

waypoints.append(waypoint2!)

let lat3 = 41.694414

let long3 = -86.226490

var waypoint3: PXLocation?

waypoint3 = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat3, long3))

waypoints.append(waypoint3!)

let lat4 = 41.702451

let long4 = -86.232815

var waypoint4: PXLocation?

waypoint4 = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat4, long4))

waypoints.append(waypoint4!)

let lat5 = 41.699957

let long5 = -86.216555

var waypoint5: PXLocation?

waypoint5 = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(lat5, long5))

var waypointSSS: PXLocation?

waypointSSS = PXLocation.CoordinateLocation(CLLocationCoordinate2DMake(41.669419, -86.250977))

waypoints.append(waypoint5!)

waypoints.append(waypoint4!)

waypoints.append(waypoint3!)

waypoints.append(waypoint2!)

waypoints.append(waypoint!)

waypoints.append(waypointSSS!)

directionsAPI.waypoints = waypoints

}

private func calculateRoute(){

directionsAPI.calculateDirections { (response) -> Void in

dispatch\_async(dispatch\_get\_main\_queue(), { () -> Void in

switch response {

case let .Error(\_, error):

let alert = UIAlertController(title: "PXGoogleDirectionsSample", message: "Error: \(error.localizedDescription)", preferredStyle: UIAlertControllerStyle.Alert)

alert.addAction(UIAlertAction(title: "OK", style: .Default, handler: nil))

self.presentViewController(alert, animated: true, completion: nil)

case let .Success(request, routes):

if let rvc = self.storyboard?.instantiateViewControllerWithIdentifier("Results") as? ResultsViewController {

rvc.request = request

rvc.results = routes

self.presentViewController(rvc, animated: true, completion: nil)

}

}

})

}

}

override func viewWillAppear(animated: Bool) {

super.viewWillAppear(animated)

updateRoute()

}

func updateRoute() {

for i in 0 ..< (results).count {

if i != routeIndex {

results[i].drawOnMap(mapView, strokeColor: UIColor.lightGrayColor(), strokeWidth: 3.0)

}

}

mapView.animateWithCameraUpdate(GMSCameraUpdate.fitBounds(results[routeIndex].bounds, withPadding: -10.0))

results[routeIndex].drawOnMap(mapView, strokeColor: UIColor.purpleColor(), strokeWidth: 4.0)

results[routeIndex].drawOriginMarkerOnMap(mapView, title: "Origin", color: UIColor.greenColor(), opacity: 1.0, flat: true)

results[routeIndex].drawDestinationMarkerOnMap(mapView, title: "Destination", color: UIColor.redColor(), opacity: 1.0, flat: true)

directions.reloadData()

}

func mqttSetting() {

let clientIdPid = "CocoaMQTT-" + String(NSProcessInfo().processIdentifier)

mqtt = CocoaMQTT(clientId: clientIdPid, host: "api.devicewise.com", port: 1883)

//mqtts

//let mqtt = CocoaMQTT(clientId: clientIdPid, host: "localhost", port: 8883)

//mqtt.secureMQTT = true

if let mqtt = mqtt {

mqtt.username = "gweber2@nd.edu"

mqtt.password = "BusTracker2016!"

//mqtt.willMessage = CocoaMQTTWill(topic: "/will", message: "dieout")

mqtt.keepAlive = 60

mqtt.delegate = self

mqtt.connect()

//mqtt.subscribe("thing/357164040554068/property/latitude")

//dispatch\_main()

//originField = CLLocationCoordinate2D(latitude:41.679419,longitude:-86.260977)

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EXTENSIONS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extension ResultsViewController:UITableViewDataSource{

func numberOfSectionsInTableView(tableView: UITableView) -> Int {

return (results[routeIndex].legs).count

}

func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

return (results[routeIndex].legs[section].steps).count

}

func tableView(tableView: UITableView, titleForHeaderInSection section: Int) -> String? {

let leg = results[routeIndex].legs[section]

formatter.dateFormat = "hh:mm a"

if let dist = leg.distance?.description, dur = leg.duration?.description, durr = leg.duration?.duration {

let abs = Double(durr)

// print(section)

let defaultTime = formatter.stringFromDate(date);

formatter.dateFormat = "hh:mm a"

var arrtime = date

var addDuration = [Double](count:15, repeatedValue: 0.0)

var totalDuration:Double

totalDuration = 0

// print(results[routeIndex].legs[0].endLocation)

// fromLatitude = (results[routeIndex].legs[0].endLocation?.latitude)!

// fromLongitude = (results[routeIndex].legs[0].endLocation?.longitude)!

// var tmp = results[routeIndex].legs.count-1

//

// toLatitude = (results[routeIndex].legs[tmp].endLocation?.latitude)!

// toLongitude = (results[routeIndex].legs[tmp].endLocation?.longitude)!

//

for var n=0;n<=results[routeIndex].legs.count-1;++n

{

addDuration[n+1] = Double((results[routeIndex].legs[n].duration?.duration)!)

//arrtime = date.dateByAddingTimeInterval(Double((results[routeIndex].legs[n].duration?.duration)!))

}

for var n=0;n<=results[routeIndex].legs.count-1;++n

{

totalDuration = totalDuration + addDuration[n+1]

//arrtime = date.dateByAddingTimeInterval(Double((results[routeIndex].legs[n].duration?.duration)!))

}

arrtime = date.dateByAddingTimeInterval(totalDuration)

let aTime = formatter.stringFromDate(arrtime);

arriveTime.text = aTime

return "Stop \(section + 1) (\(dist), \(dur))"

} else {

return "Stop \(section + 1)"

}

}

func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {

var cell = tableView.dequeueReusableCellWithIdentifier("RouteStep")

if (cell == nil) {

cell = UITableViewCell(style: .Subtitle, reuseIdentifier: "RouteStep")

}

let step = results[routeIndex].legs[indexPath.section].steps[indexPath.row]

print(results[routeIndex].legs[0].steps[0].endLocation)

if let instr = step.rawInstructions {

cell!.textLabel!.text = instr

}

if let dist = step.distance?.description, dur = step.duration?.description {

cell!.detailTextLabel?.text = "\(dist), \(dur)"

}

return cell!

}

}

extension ResultsViewController: CocoaMQTTDelegate {

func mqtt(mqtt: CocoaMQTT, didConnect host: String, port: Int) {

print("didConnect \(host):\(port)")

}

func mqtt(mqtt: CocoaMQTT, didConnectAck ack: CocoaMQTTConnAck) {

//print("didConnectAck \(ack.rawValue)")

if ack == .ACCEPT {

mqtt.subscribe("thing/357164040554068/property/latitude", qos: CocoaMQTTQOS.QOS0)

mqtt.subscribe("thing/357164040554068/property/longitude", qos: CocoaMQTTQOS.QOS0)

mqtt.ping()

}

}

func mqtt(mqtt: CocoaMQTT, didPublishMessage message: CocoaMQTTMessage, id: UInt16) {

print("didPublishMessage with message: \(message.string)")

}

func mqtt(mqtt: CocoaMQTT, didPublishAck id: UInt16) {

print("didPublishAck with id: \(id)")

}

func mqtt(mqtt: CocoaMQTT, didReceiveMessage message: CocoaMQTTMessage, id: UInt16 ) {

print("didReceivedMessage: \(message.string) with id \(message.topic)")

if (message.topic == "thing/357164040554068/property/latitude"){

// mqttTestLabel.text = message.string

let originDoubleLat:Double = Double(message.string!)!

originField.latitude = originDoubleLat

}

if (message.topic == "thing/357164040554068/property/longitude"){

let originDoubleLong:Double = Double(message.string!)!

originField.longitude = originDoubleLong

}

}

func mqtt(mqtt: CocoaMQTT, didSubscribeTopic topic: String) {

print("didSubscribeTopic to \(topic)")

}

func mqtt(mqtt: CocoaMQTT, didUnsubscribeTopic topic: String) {

print("didUnsubscribeTopic to \(topic)")

}

func mqttDidPing(mqtt: CocoaMQTT) {

print("didPing")

}

func mqttDidReceivePong(mqtt: CocoaMQTT) {

\_console("didReceivePong")

}

func mqttDidDisconnect(mqtt: CocoaMQTT, withError err: NSError?) {

\_console("mqttDidDisconnect")

}

func \_console(info: String) {

print("Delegate: \(info)")

}

}

extension ResultsViewController: UITableViewDelegate {

}

extension MainViewController: PXGoogleDirectionsDelegate {

func googleDirectionsWillSendRequestToAPI(googleDirections: PXGoogleDirections, withURL requestURL: NSURL) -> Bool {

NSLog("googleDirectionsWillSendRequestToAPI:withURL:")

return true

}

func googleDirectionsDidSendRequestToAPI(googleDirections: PXGoogleDirections, withURL requestURL: NSURL) {

NSLog("googleDirectionsDidSendRequestToAPI:withURL:")

NSLog("\(requestURL.absoluteString.stringByAddingPercentEncodingWithAllowedCharacters(NSCharacterSet.URLQueryAllowedCharacterSet())!)")

}

func googleDirections(googleDirections: PXGoogleDirections, didReceiveRawDataFromAPI data: NSData) {

NSLog("googleDirections:didReceiveRawDataFromAPI:")

NSLog(NSString(data: data, encoding: NSUTF8StringEncoding) as! String)

}

func googleDirectionsRequestDidFail(googleDirections: PXGoogleDirections, withError error: NSError) {

NSLog("googleDirectionsRequestDidFail:withError:")

NSLog("\(error)")

}

func googleDirections(googleDirections: PXGoogleDirections, didReceiveResponseFromAPI apiResponse: [PXGoogleDirectionsRoute]) {

NSLog("googleDirections:didReceiveResponseFromAPI:")

NSLog("Got \(apiResponse.count) routes")

for i in 0 ..< apiResponse.count {

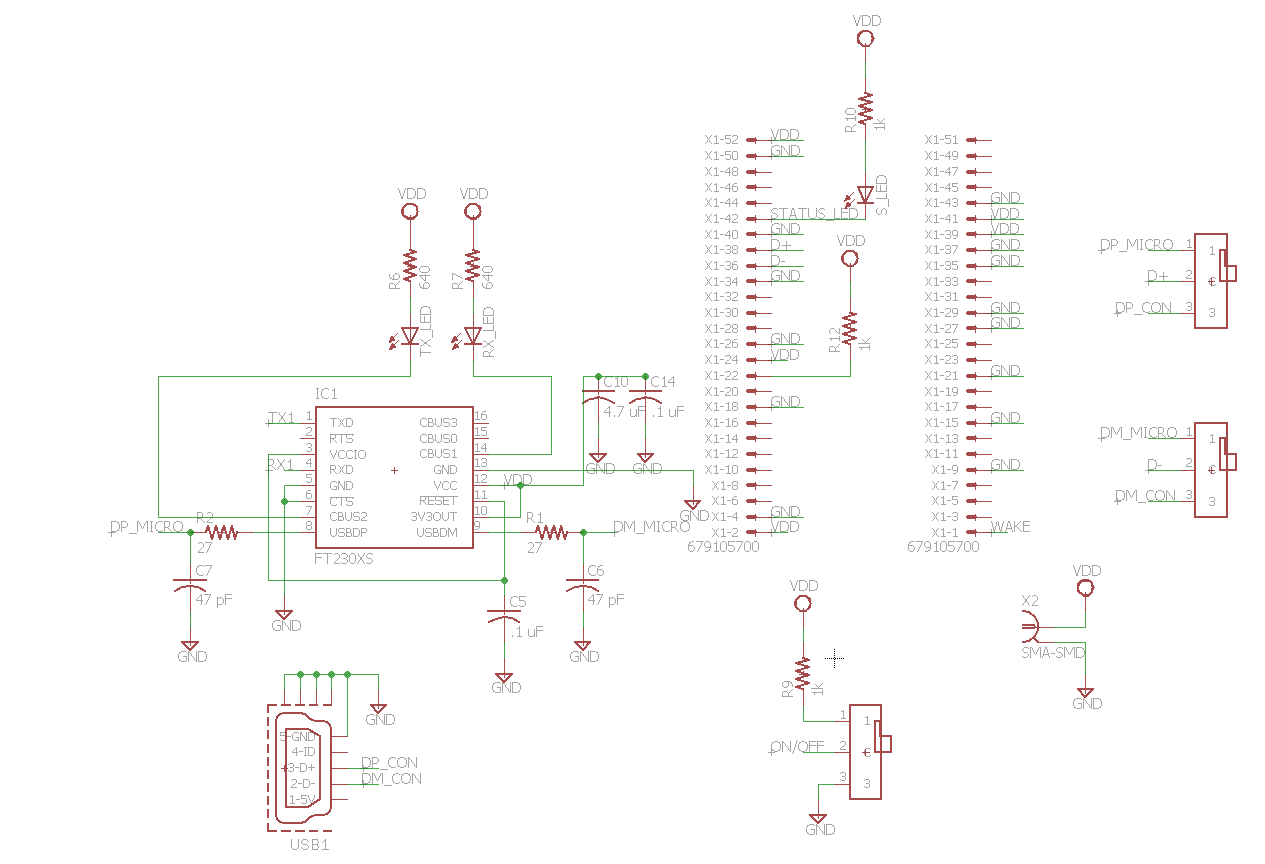
NSLog("Route \(i) has \(apiResponse[i].legs.count) legs")

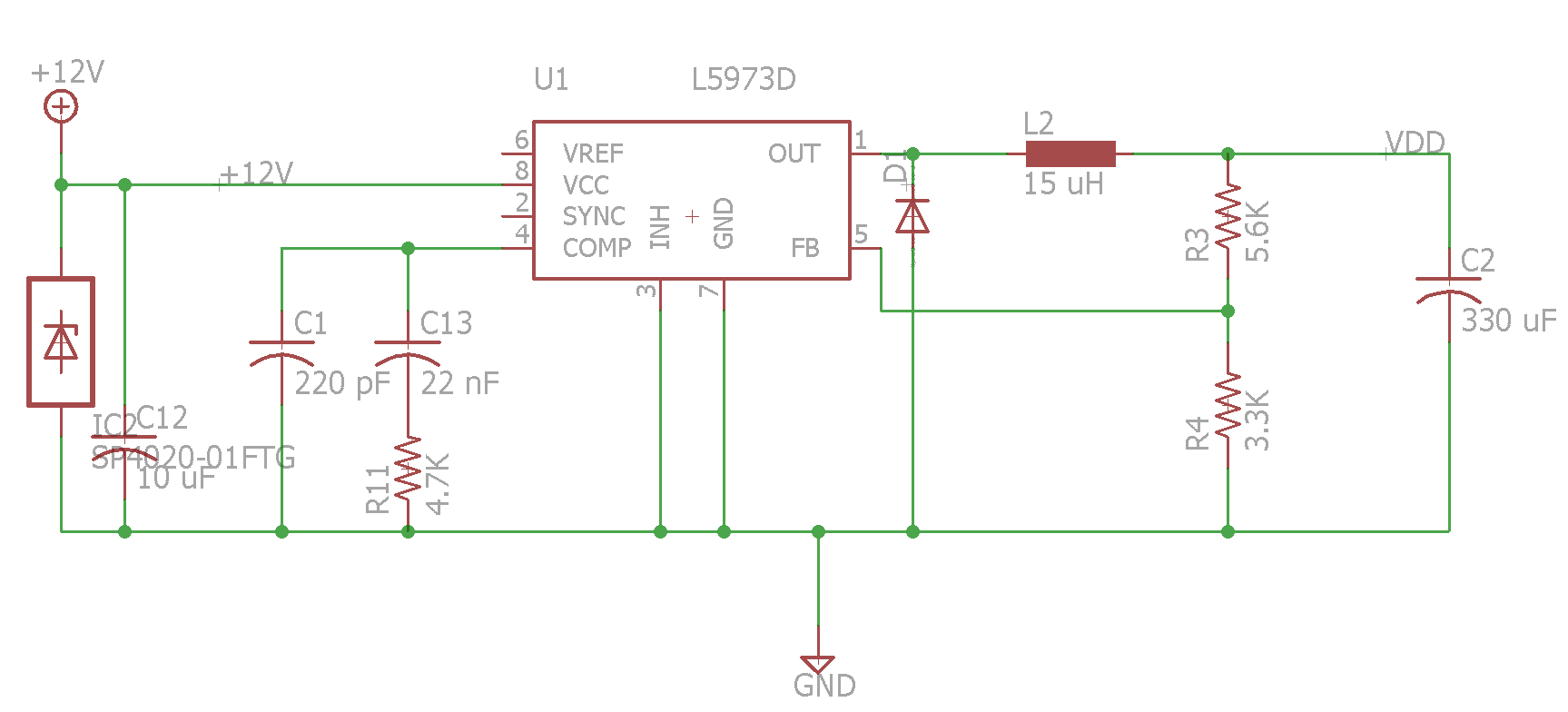
}

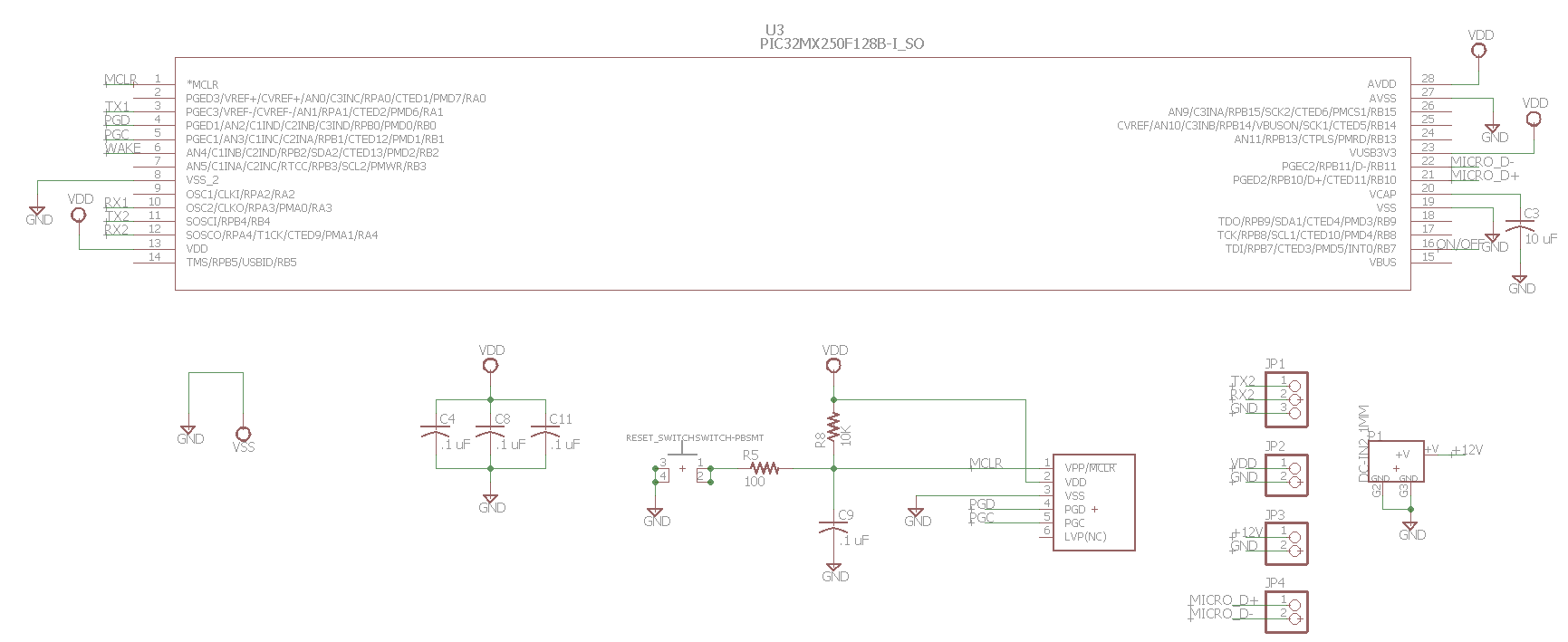
}

}

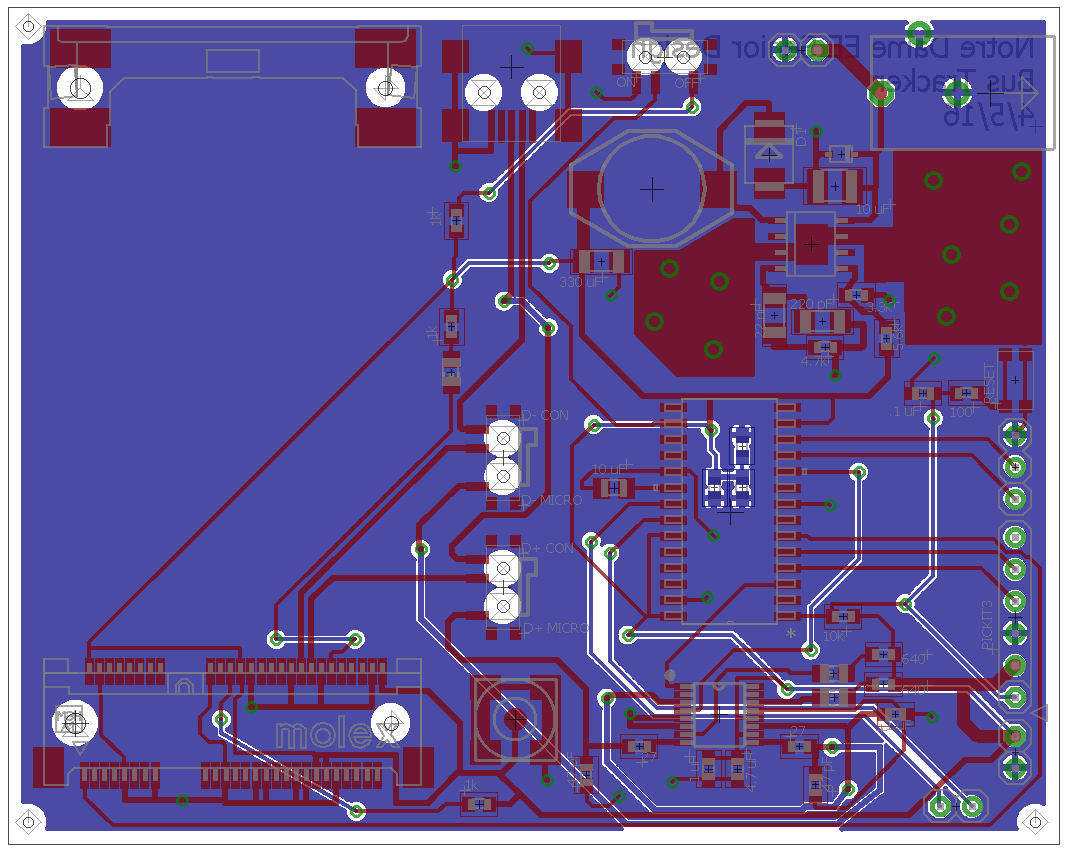
## Appendix D: Final Board Schematics







## Appendix E: Final Board



## Appendix F: Relevant Data Sheets and Hardware Guides:

PIC32MX250F128B Datasheet:

<http://ww1.microchip.com/downloads/en/DeviceDoc/60001168J.pdf>

FT230XS Datasheet:

<http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT230X.pdf>

Telit HE910 Mini PCIE Hardware Guide:

<http://www.telit.com/fileadmin/user_upload/products/Downloads/Telit_xE910_Mini_PCIe_Adapter_Hardware_User_Guide_r10.pdf>

Telit AT Commands Reference Guide:

<http://www.janus-rc.com/Documentation/Telit_HE910_AT_Commands_Reference_Guide_r5.pdf>

Telit HE910 Hardware Guide:

<http://www.telit.com/fileadmin/user_upload/products/Downloads/3G/xe910/Telit_HE910_Hardware_User_Guide_r27.pdf>

Mini-Circuits Bias Tee Datasheet:

<http://www.minicircuits.com/pdfs/ZFBT-4R2G+.pdf>

## Appendix G: Build of Materials for PCB

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Qty | Value | Device | Package | Parts | Description |
| 3 |  | LEDCHIP-LED0805 | CHIP-LED0805 | RX\_LED, S\_LED, TX\_LED | LED |
| 3 |  | PINHD-1X2S | 1X02N | JP2, JP3, JP4 | PIN HEADER |
| 1 |  | PINHD-1X3 | 1X03N | JP1 | PIN HEADER |
| 1 |  | USB-MINI-MICRO-BMINI | USB-MINI-FCI10033527 | USB1 |  |
| 6 | .1 uF | C-USC0603 | C0603 | C4, C5, C8, C9, C11, C14 | CAPACITOR, American symbol |
| 1 | 10 uF | C-USC0805K | C0805K | C3 | CAPACITOR, American symbol |
| 1 | 10 uF | GRM32DR61E106KA12L | C1210 | C12 | CAPACITOR, American symbol, Voltage regulator part |
| 1 | 100 | R-US\_R0603 | R0603 | R5 | RESISTOR, American symbol |
| 1 | 10K | R-US\_R0603 | R0603 | R8 | RESISTOR, American symbol |
| 1 | 15 uH | DO3316P | DO3316P | L2 | COILCRAFT |
| 3 | 1k | R-US\_R0603 | R0603 | R9, R10, R12 | RESISTOR, American symbol |
| 1 | 22 nF | C-USC1206C223K5RAC | C1206 | C13 | CAPACITOR, American symbol, Voltage regulator part |
| 1 | 220 pF | C-USC1206C221J5GAC | C1206 | C1 | CAPACITOR, American symbol |
| 2 | 27 | R-US\_R0603 | R0603 | R1, R2 | RESISTOR, American symbol |
| 1 | 3.3K | R-US\_R0603 | R0603 | R4 | RESISTOR, American symbol |
| 1 | 330 uF | POSCAP 6TPB330M | C3216 | C2 | CAPACITOR, American symbol |
| 1 | 4.7 uF | C-USC0603 | C0603 | C10 | CAPACITOR, American symbol |
| 1 | 4.7K | R-US\_R0603 | R0603 | R11 | RESISTOR, American symbol |
| 2 | 47 pF | C-USC0603 | C0603 | C6, C7 | CAPACITOR, American symbol |
| 1 | 5.6K | R-US\_R0603 | R0603 | R3 | RESISTOR, American symbol |
| 2 | 270 | R-US\_R0603 | R0603 | R6, R7 | RESISTOR, American symbol |
| 1 | 679105700 | 679105700 | 679105700 | X1 | MINI PCI EXPRESS CONNECTOR 0.8MM PITCH, 52 CIRCUIT |
| 1 | DC-IN2.1MM | DC-IN2.1MM | DC-POWERJACK | P1 |  |
| 1 | ES2D | STPS2L25U | SMB | D1 | DIODE, Need 2A protection |
| 1 | FT230XS | FT230XS | SSOP16 | IC1 | USB to BASIC UART IC |
| 1 | L5973D | L5973D | HSOP8 | U1 | 2.5A switch step down switching regulator |
| 1 | PIC32MX250F128B-I\_SO | PIC32MX250F128B-I\_SO | SOIC28-W\_MC | U3 |  |
| 1 | PICKIT3 | PICKIT3 | PICKIT3 | U$1 |  |
| 3 | SLIDE-SPDTCUS12B | SLIDE-SPDTCUS12B | CUS12B | U$5, U$6, U$7 |  |
| 1 | SMA-SMD | SMA-SMD | SMA-SMD | X2 | SMA 50 Ohm Straight Jack Receptacle - Surface Mount |
| 1 | SP4020-01FTG | SP4020-01FTG | SOD323 | IC2 | SP4020 Series 2.5pF, 30A Discrete TVS Diode |
| 1 | SWITCH-PBSMT | SWITCH-PBSMT | SMTPBSW | RESET\_SWITCH |  |