UNIVERSITY OF ALBERTA

iOS Device Controlled RC Car

Critical Design Presentation
CMPE 450/490
Group 1

Presentation by: Max Marcus, Barry Peyton, Robert Hood

FACULTY OF ENGINEERING
Department of Electrical and Computer Engineering

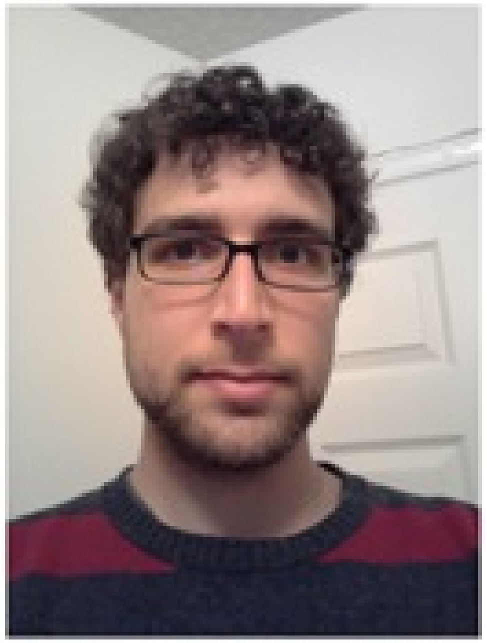
Group 1 Members

Barry Peyton

- Leading the iOS Application Development component of the project

Robert Hood

- Leading the Altera to RC Car Interfacing component of the project

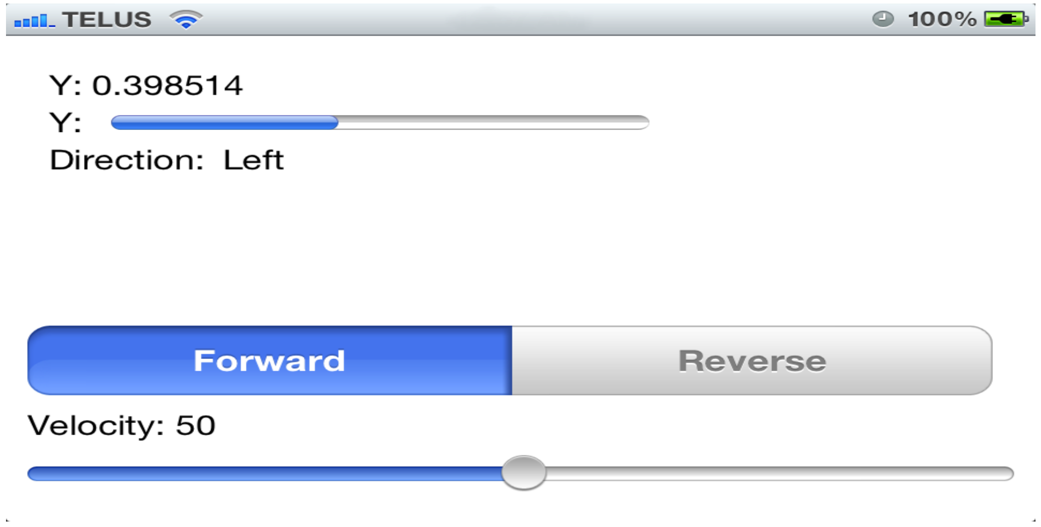

Max Marcus

- Leading the FPGA/Wi-Fi Module Development component of the project

Functionality

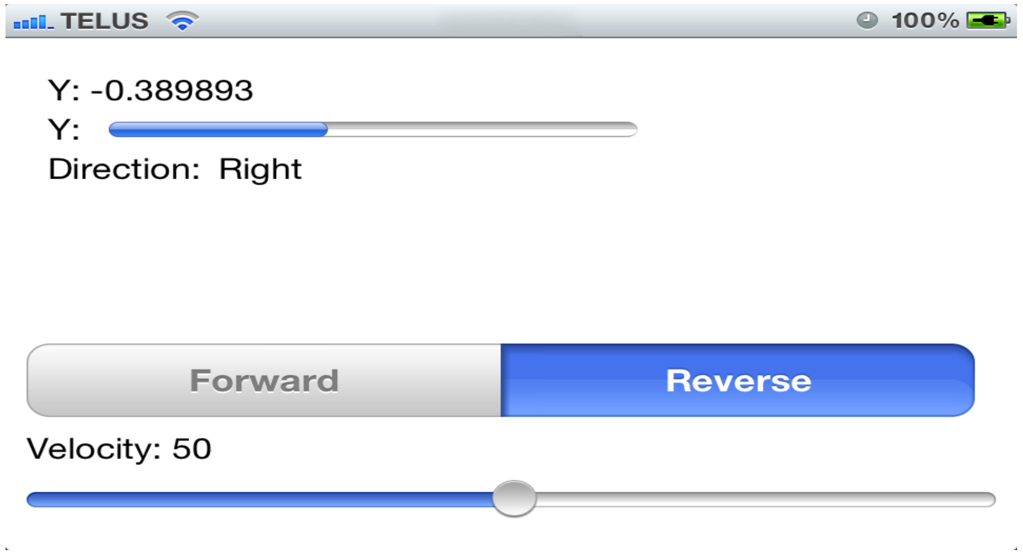
- The movement of the RC car will be fully controlled by the iOS device application
- The RC car will have the ability to be driven either forward or reverse as well as turn left or right
- The acceleration of the car either in the forward or reverse direction will be based on user input on the iOS device
- The RC car will turn left & right according to the degree of tilt of the built-in Accelerometer on the iOS Device

User Interface (Snap Shot)



Y: 0.398514
Y:
Direction: Left

Velocity: 50



Y: -0.389893
Y:
Direction: Right

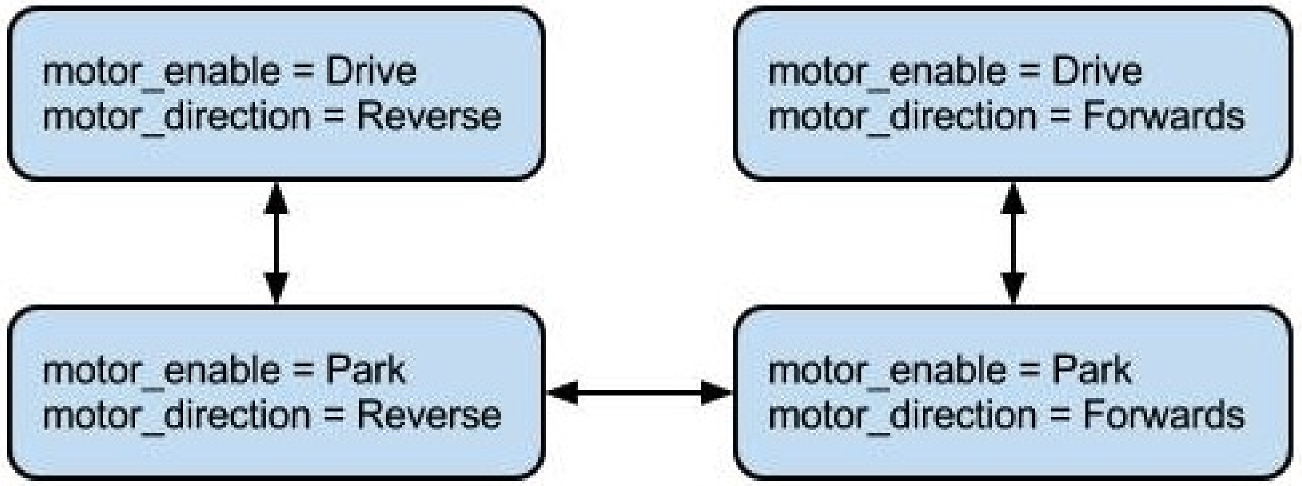
Velocity: 50

Motivation for the Project

- Practical Applications:
  - Search and Rescue
  - Suspicious package identification
  - Remote Control Surveillance
- Personal Interests
  - Application development
  - Digital Control of Mechanical Systems

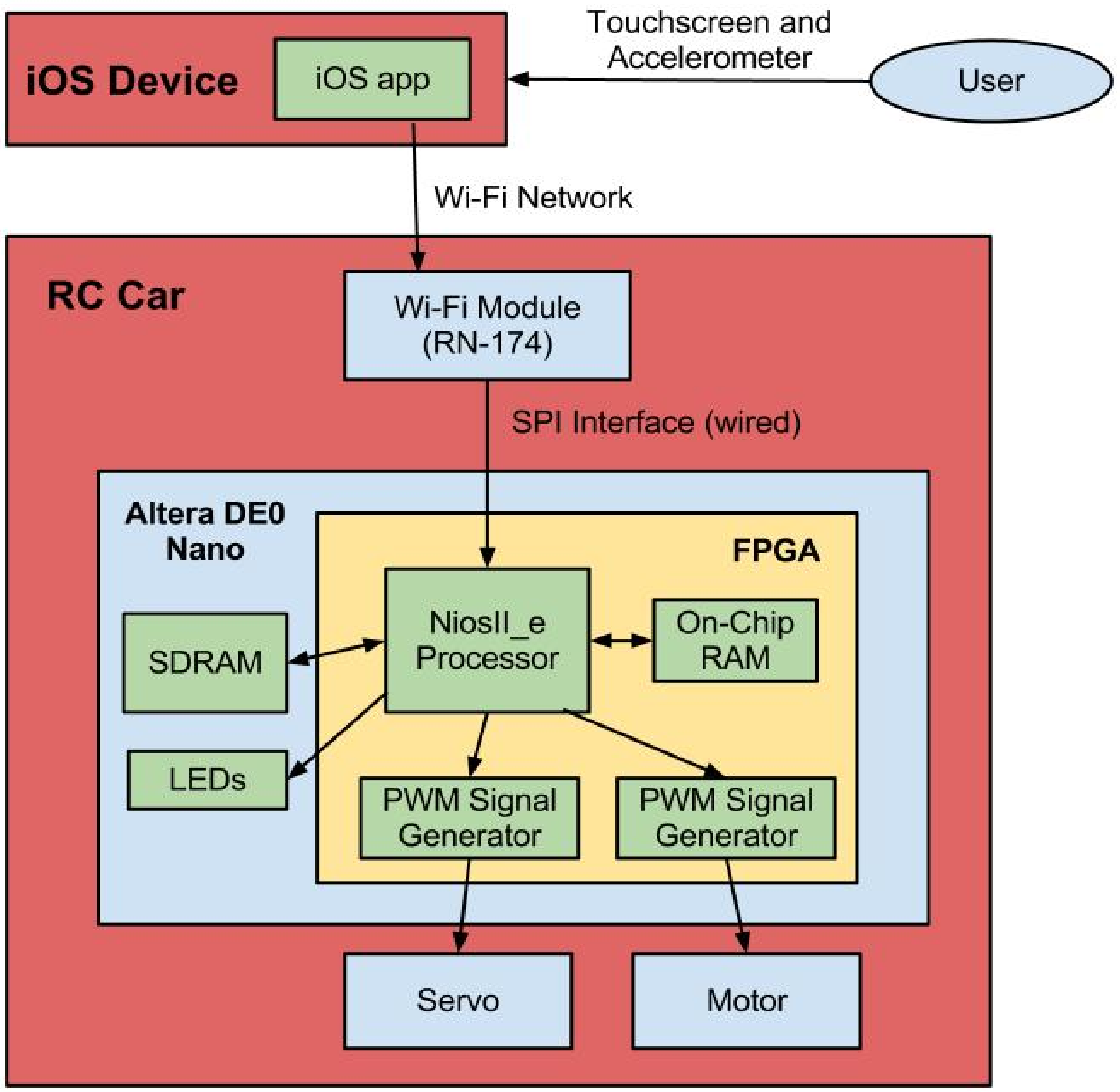
Design

- The user interacts with the iOS app via the touchscreen and accelerometer.
- The iPhone app interacts with the RN-174 Wi-Fi module wirelessly via the iPhone's Wi-Fi radio over a UDP Broadcast (i.e. data pipeline).
- The RN-174 Wi-Fi module interacts with the FPGA-implemented CPU via a four pin SPI interface.
- The CPU reads the movement commands and then sends the appropriate commands to the Pulse Width Modulation (PWM) Signal Generators.
- One PWM Generator sends signals to the servo on the RC car to control the position of the servo arm for steering.
- The other PWM Generator sends signals to the motor to control its speed.


motor_enable = Drive
motor_direction = Reverse
motor_enable = Drive
motor_direction = Forwards
motor_enable = Park
motor_direction = Reverse
motor_enable = Park
motor_direction = Forwards

System Modelled as a Finite State Machine

Hardware Block Diagram



Bumps In The Road Thus Far

Bluetooth Communication

Pros:

- iOS devices have built in Bluetooth functionality
- Its never been implemented in CMPE 450 before
- low power consumption
- Security

Cons:

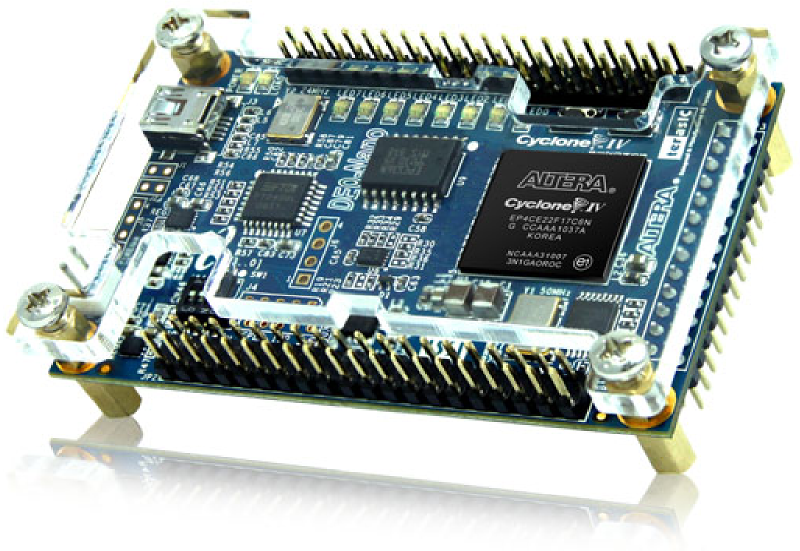
- MFI program

Solution:

WiFi-Module

Altera DE0 Nano FPGA Board

- Equipped with a Cyclone IV FPGA
- Two 40-pin Headers (GPIOs) providing 72 I/O pins
- Small size (4.9cm x 7.25cm) makes it ideal for our application
- Requires (3.6~5.7V) to the 2-pin external power header

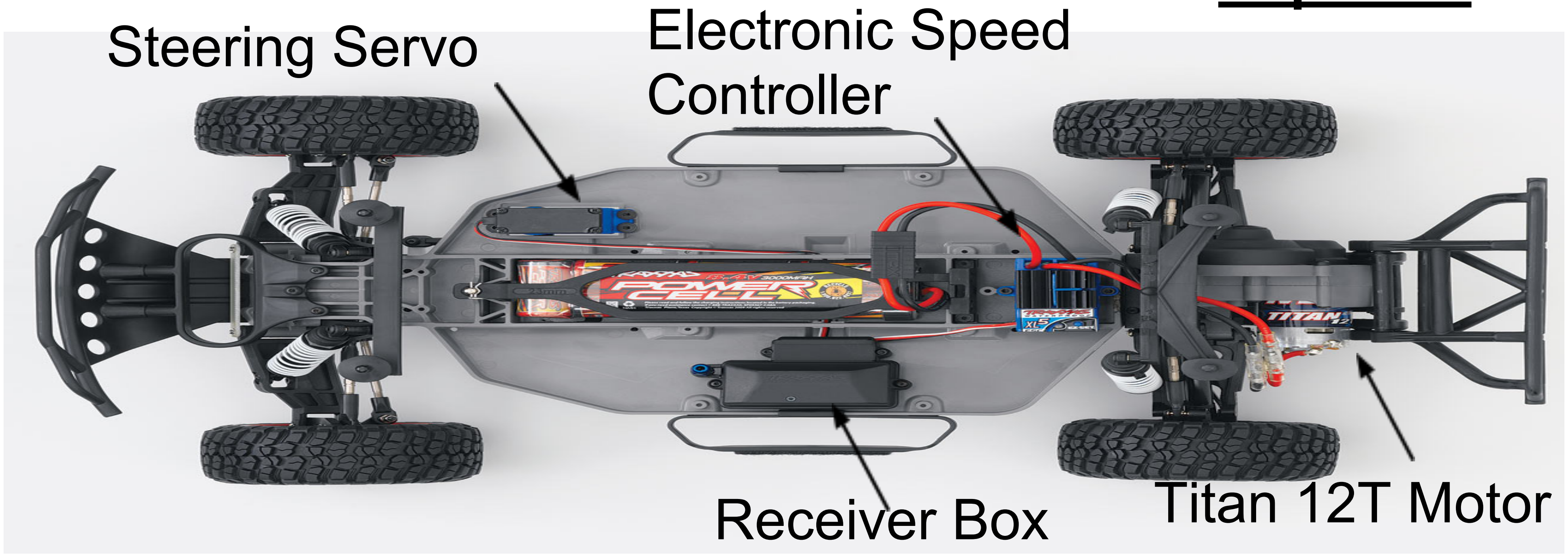


Roving Networks (RN-174) Wi-Fi Module

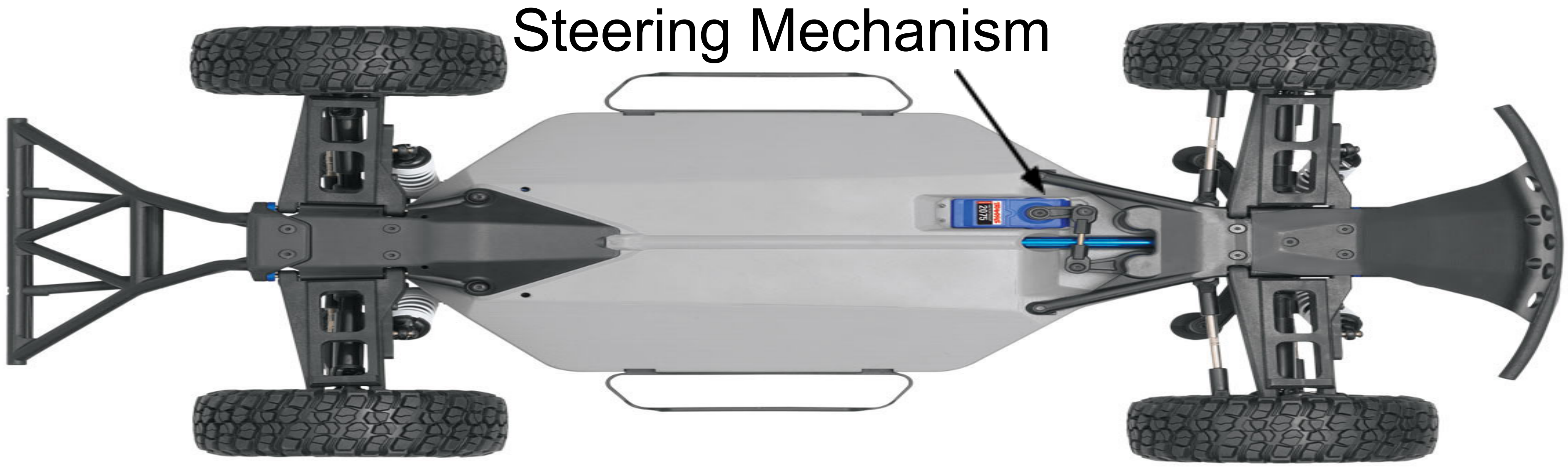
- Module is manufactured by Roving Networks
- RN-174 is a through-hole development platform built to be used with the RN-171Wi-Fi Module (surface mount)
- The module will be connected to the DE0 Nano Board using an SPI Slave Interface
- It will be connected to the iOS device via an Adhoc network
- Requires 3.3V to be powered

Traxxas RC Car (SLASH 5805)

Top View

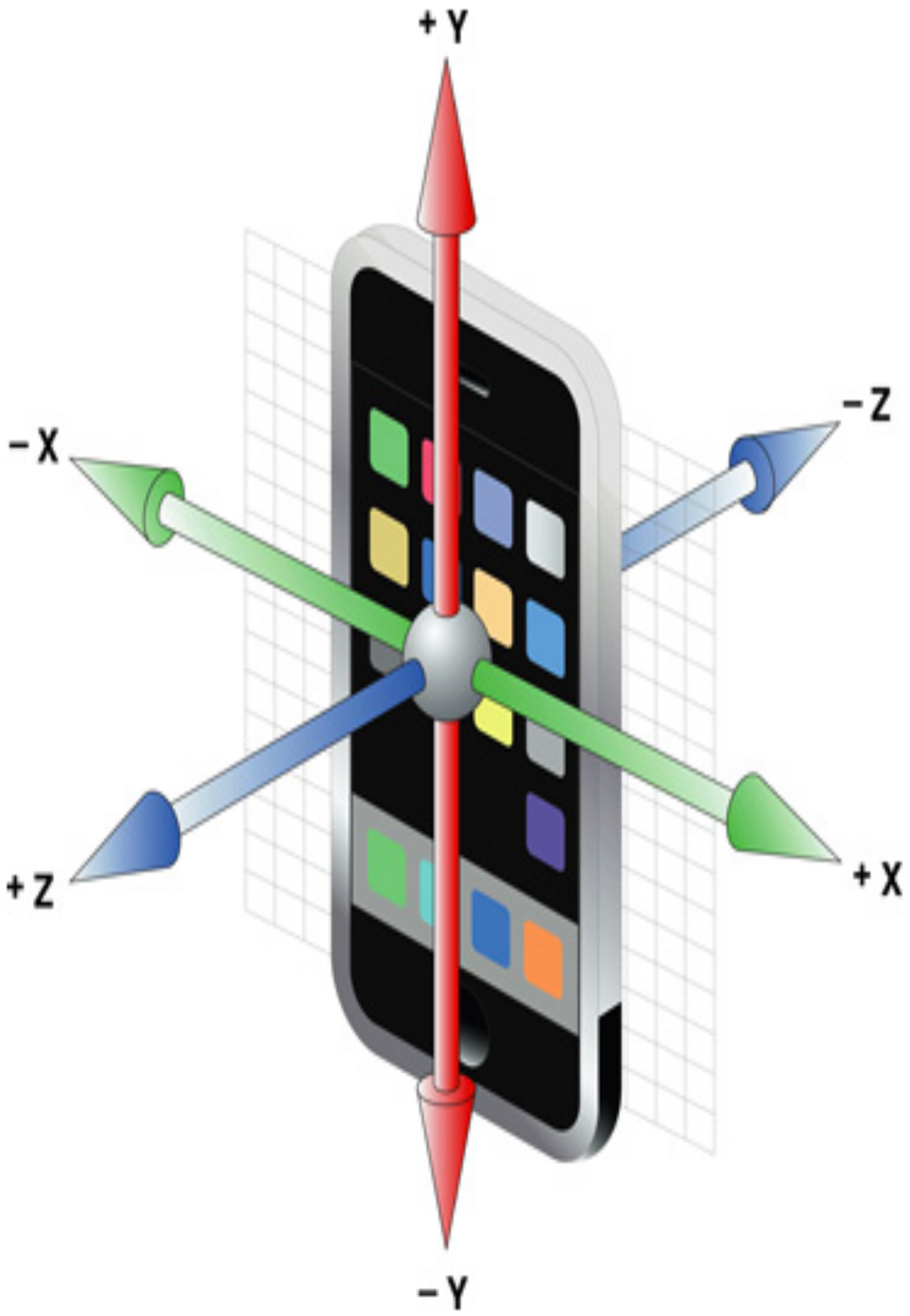


Bottom View



Code Example (.h File)

```
//  MainViewController.h
//  AccelerometerTest
//  Created by Barry Peyton on 12-02-05.
//  Copyright 2012 University of Alberta. All rights reserved.
#import <UIKit/UIKit.h>

@interface MainViewController : UIViewController
<UIAccelerometerDelegate> {
  IBOutlet UILabel *labelX;
  IBOutlet UILabel *labelY;
  IBOutlet UILabel *labelZ;
  IBOutlet UIProgressView *progressY;
  IBOutlet UISlider *sliderCtl;
  UISegmentedControl *segControl;
  UIAccelerometer *accelerometer;
  NSNumber *velocity;
}

//Components used to implement the .XIB file for the interface
@property (nonatomic, retain) IBOutlet UILabel *labelX;
@property (nonatomic, retain) IBOutlet UILabel *labelY;
@property (nonatomic, retain) IBOutlet UILabel *labelZ;
@property (nonatomic, retain) IBOutlet UIProgressView *progressY;
@property (nonatomic, retain) UISlider *sliderCtl;
@property (nonatomic, retain) UISegmentedControl *segControl;
@property (nonatomic, retain) UIAccelerometer *accelerometer;
@property (nonatomic, retain) NSNumber *velocity;

//Methods used to detect user inputs
-(IBAction) sliderChanged:(id) sender;

-(IBAction) segmentedControlIndexChanged:(id) sender;

@end
```



Code Example (.m File)

```
//  MainViewController.m
//  AccelerometerTest
//  Created by Barry Peyton on 12-02-05.
//  Copyright 2012 University of Alberta. All rights reserved.
#import "MainViewController.h"
@implementation MainViewController
@synthesize labelX,labelY,labelZ,progressY;
@synthesize sliderCtl,segControl,accelerometer;
@synthesize velocity;

- (void)viewDidLoad {
   [super viewDidLoad];
   //create accelerometer object
   self.accelerometer = [UIAccelerometer sharedAccelerometer];
   self.accelerometer.updateInterval = .1;
   self.accelerometer.delegate = self;
   //create segmented control object
   self.segControl = [[UISegmentedControl alloc] init];
   [segControl addTarget:self action:@selector
(segmentedControlIndexChanged::) forControlEvents:
UIControlEventValueChanged];
   self.segControl.selectedSegmentIndex = 0;
}
- (void)accelerometer:(UIAccelerometer *)accelerometer
didAccelerate:(UIAcceleration *)acceleration {

   labelY.text = [NSString stringWithFormat:@"%@%f", @"Y: ",
acceleration.y];
   //determine direction (Left or Right)
 if (acceleration.y > 0) {
      labelZ.text = [NSString stringWithFormat:@"%@", @"Left"];
   } else if (acceleration.y < 0){
      labelZ.text = [NSString stringWithFormat:@"%@", @"
Right"];
   } else {
      labelZ.text = [NSString stringWithFormat:@"%@", @"
Stopped"];    }
   //Update progress bar
   self.progressY.progress = ABS(acceleration.y);
   //create slider acceleration controller
   if (sliderCtl == nil) {
      CGRect frame = CGRectMake(174.0, 12.0, 120.0, 7.0);
      sliderCtl = [[UISlider alloc] initWithFrame:frame];
      [sliderCtl addTarget:self action:@selector(sliderAction:)
forControlEvents:UIControlEventValueChanged];
      sliderCtl.backgroundColor = [UIColor clearColor];
      sliderCtl.minimumValue = 0.0;
      sliderCtl.maximumValue = 100.0;
      sliderCtl.continuous = YES;
   }
}
- (void)sliderChanged:(id)sender{
   //if slider value changes update label
   self.velocity = [NSNumber numberWithDouble:(sliderCtl.value
+ 0.5f)];
   labelX.text = [NSString stringWithFormat:@"%@%d", @"
Velocity: ", self.velocity.intValue];
   NSLog(@"%@%d", @"Velocity: ", self.velocity.intValue);
   NSLog(@"%@%d", @"Segment Value: ", self.segControl.
selectedSegmentIndex);
}
```

Code Example (.m File)

```
-(void) segmentedControlIndexChanged:(id)sender{
    //if segment control changes negate acceleration to invoke reverse
    int value1 = -1;
    int value2 = self.velocity.intValue;
    switch (self.segControl.selectedSegmentIndex) {
        case 0:
            break;
        case 1:
            value2 = value1 * value2;
            NSLog(@"%@%d", @"Value2: ", value2);
            self.velocity = [NSNumber numberWithDouble:(value2 + 0.5
f)];
            break;
        default:
            break;
    }
}
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
interfaceOrientation {
    //allow application to rotate
    if (interfaceOrientation == UIInterfaceOrientationPortrait ||
        interfaceOrientation == UIInterfaceOrientationLandscapeLeft ||
        interfaceOrientation == UIInterfaceOrientationLandscapeRight)
        return YES;
    else {
        return NO;
    }
```

```
- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
}
- (void)viewDidUnload {
    [sliderCtl release];
    sliderCtl = nil;
}

- (void)dealloc {
    [super dealloc];
}
@end
```

Unit Test Plan

iOS:

- Can iOS application properly generate useful data from the built-in Accelerometer?
- Does iOS control logic pass all verification tests based on FSM?
- Can iOS application properly convert this accelerometer data into commands that can be read by the Wi-Fi Module? (test using a laptop)

Wi-Fi:

- Download a simple program to the module which makes its LEDs blink.
- Can Wi-Fi module be accessed over an Adhoc network and receive broadcasted UDP messages? (test using a laptop)
- Can it pass wirelessly received data to a computer over RS-232?

FPGA Board:

- Download a simple program to the module which makes its LEDs blink.

RC Car:

- Drive car around using remote control from box. Does it behave correctly?

Integrated Test Plan

- Once the separate units are tested individually, we will start to interface various components and test them together.
- Does the iOS device correctly pass commands to the Wi-Fi module?
  - Have certain LEDS blink for certain commands.
  - Have the module echo the commands to a PC over RS-232.
- Does the Wi-Fi module correctly pass commands to the CPU?
  - Download a simple program to the module which makes the LEDs on the DE0 Nano board blink.
- Can FPGA Board correctly operate the motor and steering servo based on a list of commands downloaded from a computer?
- Incrementally add one more component to each level of integration tests after that.

Possible Extensions:

- The iPhone screen will display a live video feed that is sent from a front-facing digital camera, mounted on the car.
- Use accelerometers on the DE0 Nano to detect collisions and stop the motor if one is detected.
- Proximity sensors, placed on four sides of the car, prevent the car from getting into collisions. When nearby objects get too close, the CPU temporarily blocks the user from controlling the car and runs an obstacle-avoidance subroutine, causing the car to stop and then drive a few centimeters away from the obstacle.
- Mount the digital camera on a platform that can rotate through a 180-degree horizontal viewing range. The user can control this rotation using a button on the iPhone screen.
- Add a second rear-facing camera to the car and allow the user to switch between the front-facing and rear-facing video feeds.

Possible Simplifications:

- Instead of controlling the movement of the RC car over a Wi-Fi network using an iPhone, the user would would control the car using a homemade remote control
- Only have one maximum speed that the car accelerates to and from. In this case, the Set Speed slider would become a switch (Go and Stop).

UNIVERSITY OF ALBERTA

Questions?

Critical Design Presentation
CMPE 450/490
Group 1

Presentation by: Max Marcus, Barry Peyton, Robert Hood