Bluetooth Controlled Vehicle

Rijvi Rajib – Richard Lau – Tomasz Glazewski

Overview

- Introduction
- Background
- Analysis
- Tasks
- Deliverables
- Demonstration

Introduction

- Objective
  - Multipurpose vehicle framework
  - Bluetooth powered device
  - Simple API
- Purpose
  - Military
  - Commercial

Background

- Bluetooth
  - Implement a universal standard
  - Frequency-hopping spread spectrum
    - Gaussian frequency-shift keying (GFSK)
  - 2.4 GHz short-range radio frequency
  - Class 1, Class 2, Class 3
  - 1 Mbit/s on version 1.2
  - 3 Mbit/s on version 2.0 + EDR

Background

- Android
  - Mobile Operating System
    - Linux Kernel
  - Kernel 2.6.29 (2.1)
  - Robust SDK
  - Bluetooth 2.0
  - Java programming
  - ARM Native Code

Analysis

- Hardware
  - Set of DC Servo motors
  - Optimal power supply
  - Expandable frame
  - Integrated microcontroller
  - Wireless capabilities

Item	Cost
RC Test Vehicle	$189.95
Bluetooth modem	$64.95
Total Cost	$254.90

Analysis

- Software (Vehicle)
  - Bluetooth integration
    - Keyboard profile
    - Interpret remote signals
  - Digital control of analog DC motors
  - Speed sensor and steering
  - Traction control

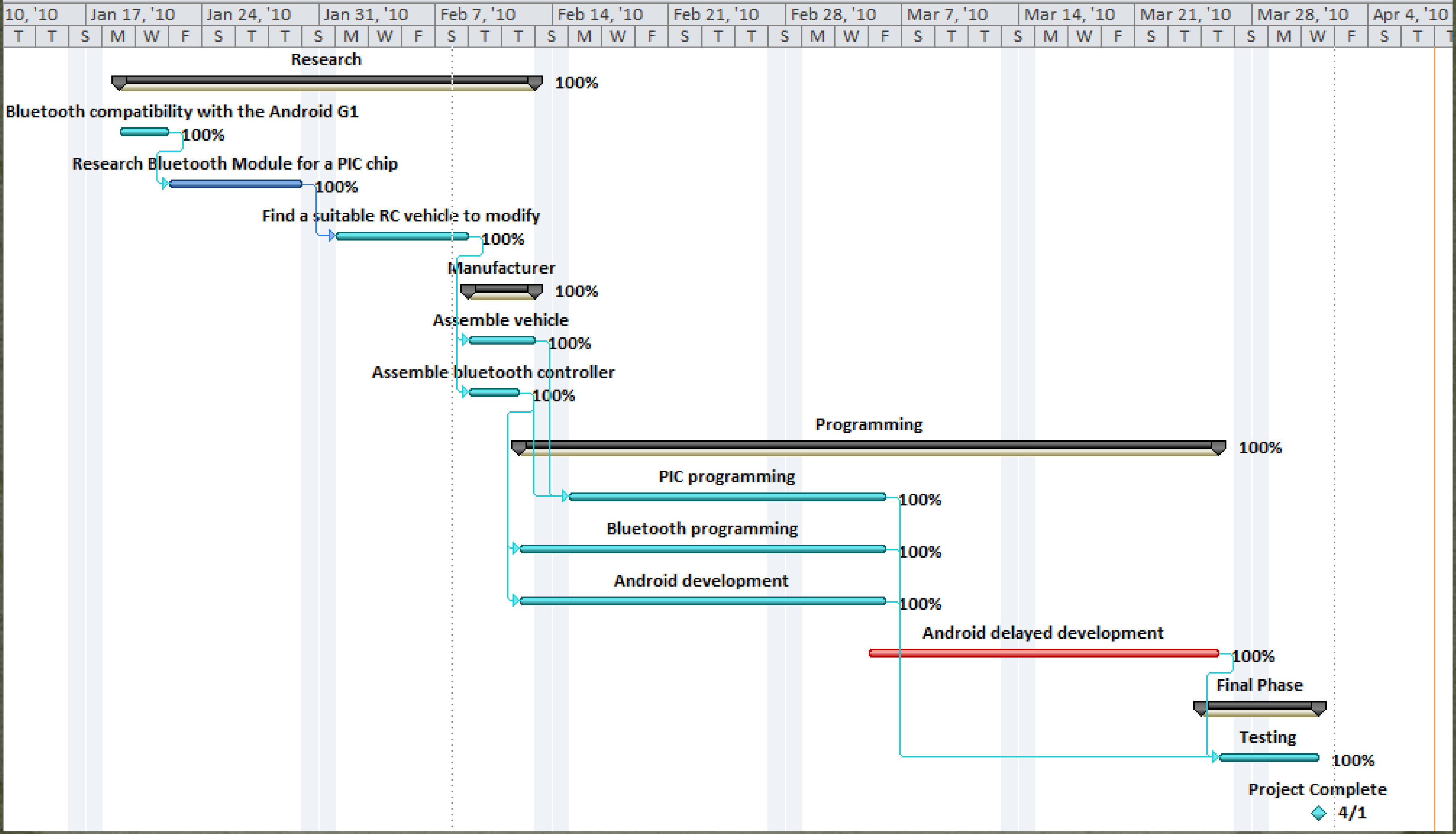
Analysis

- Software (Remote)
  - Android application
  - Windows integration
  - Bi-directional communication
    - Report speed
    - Error conditions
    - Display speeds
  - Speed control

Tasks

- Research
  - Bluetooth module
  - Operating systems
- Programming
  - Bluetooth connection
  - Android application
  - PIC C programming
- Testing
  - Multiple controllers

Tasks


10, '10
Jan 17, '10
Jan 24, '10
Jan 31, '10
Feb 7, '10
Feb 14, '10
Feb 21, '10
Feb 28, '10
Mar 7, '10
Mar 14, '10
Mar 21, '10
Mar 28, '10
Apr 4, '10
Research 100%
Bluetooth compatibility with the Android G1 100%
Research Bluetooth Module for a PIC chip 100%
Find a suitable RC vehicle to modify 100%
Manufacturer 100%
Assemble vehicle 100%
Assemble bluetooth controller 100%
Programming 100%
PIC programming 100%
Bluetooth programming 100%
Android development 100%
Android delayed development 100%
Final Phase
Testing 100%
Project Complete 4/1

Tasks

- Microcontroller Integration
  - Wire control and signals
  - Process signals from vehicle (motor speed)
  - Framework for control, feedback, and expansion

- Bluetooth Integration & Remote testing
  - Allow seamless unattended pairing
  - Process BT keyboard signals
  - Test with multiple devices
  - Simple application for cellular phone

Deliverables

- Fully functional RC vehicle supporting 4 directions of travel
  - Forward, Reverse, Left, Right
- Bluetooth remote control
  - Regular keyboard over Desktop
  - Android application
    - T-Mobile G1
    - Verizon Droid

Deliverables

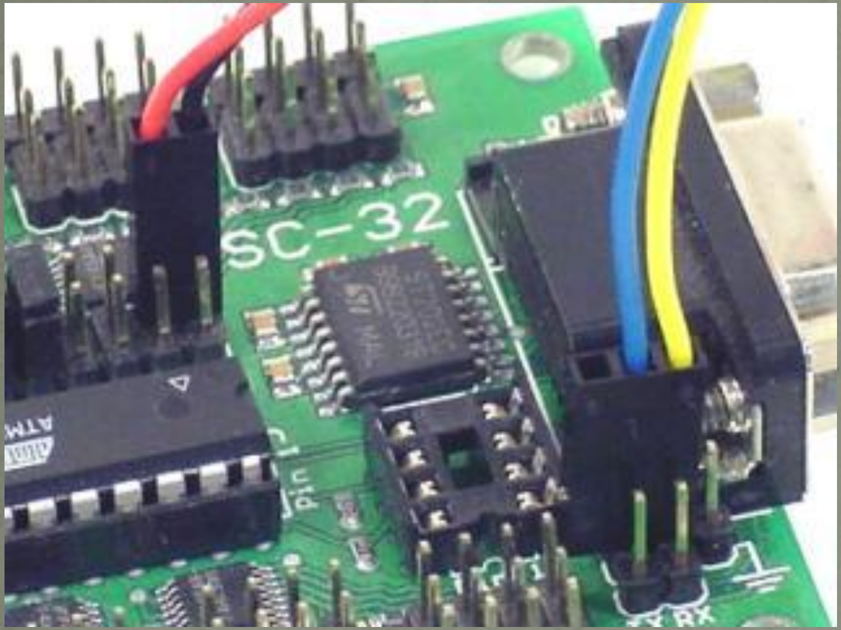
- Optimization
  - Microcontroller code optimization
  - Error handling
    - Restricted movement
    - Low battery
- Expansion
  - Vehicle options
    - Lights
    - Traction control
    - Speed reporting
  - Software add-ons

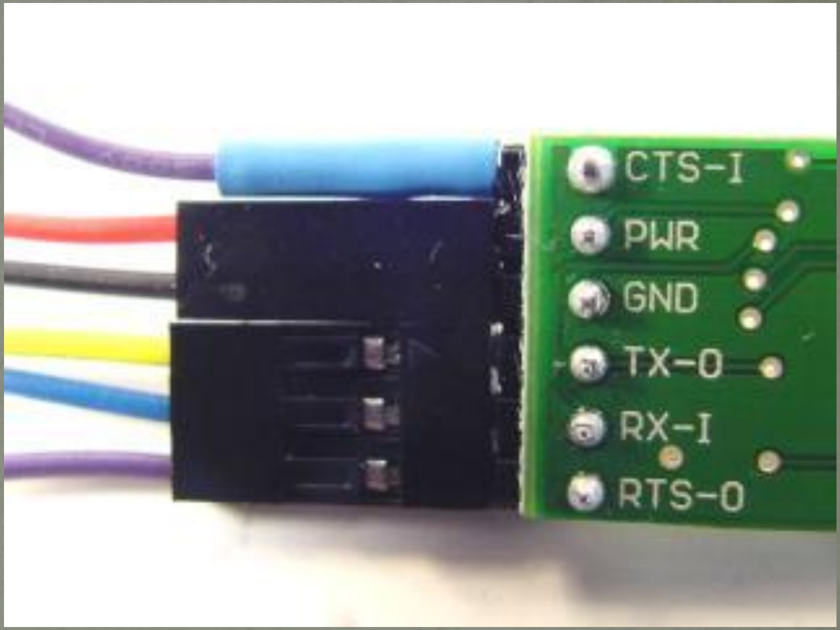
Conclusion

- The Verizon Droid Bluetooth stack is limited, by the kernel, on what it can connect to and how.
- The T-Mobile G1, with kernel modification, can successfully connect to the Bluetooth modem for controls.
- The vehicle has additional functionality, on top of the everyday control method.

Updates

- Hardware changes





Microcontroller connected

- Rx connection for receiving
- Tx connection for transmitting

Bluetooth modem connected

- Custom hardware soldered

Updates

- Code Snippets

```
void Change_Duty(char speed)
 {
        if (speed != motor_duty_)       // Check Same old speed
        {
               motor_duty_=speed;             // Save for old speed
               Pwm1_Change_Duty(speed);      // Motor A
               Pwm2_Change_Duty(speed);      // Motor B
        }
 }
void Motor_A_Forward()
{
        Pwm1_Start();
        PORTD.F0 =0;
        PORTD.F1 =1;
}
```

BluetoothService.java

Writing BYTE data to the Bluetooth mode,

```
PUBLIC VOID WRITE(BYTE[] OUT) {
        // CREATE TEMPORARY OBJECT
                CONNECTEDTHREAD R;
                // SYNCHRONIZE A COPY OF THE CONNECTEDTHREAD
                SYNCHRONIZED (THIS) {
                        IF (MSTATE != STATE_CONNECTED) RETURN;
                        R = MCONNECTEDTHREAD;
                }
                // PERFORM THE WRITE UNSYNCHRONIZED
                R.WRITE(OUT);
}


PUBLIC VOID WRITE(BYTE[] BUFFER) {
                TRY {
                        MMOUTSTREAM.WRITE(BUFFER);
                        // SHARE THE SENT MESSAGE BACK TO THE UI ACTIVITY
                        MHANDLER.OBTAINMESSAGE(BLUECAR.MESSAGE_WRITE, -1, -1, BUFFER)
                        .SENDTOTARGET();
                } CATCH (IOEXCEPTION E) {
                        LOG.E(TAG, "EXCEPTION DURING WRITE", E);
                }
}
```