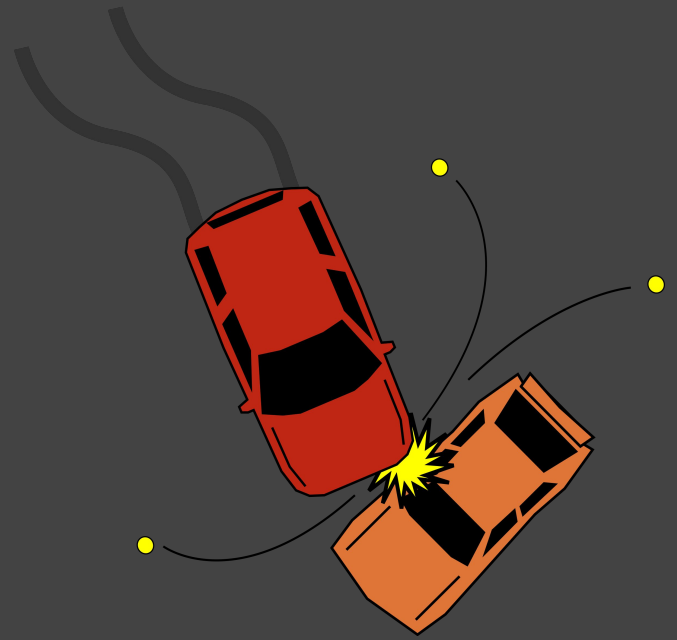
Heterogeneous **A**utomotive **R**esponse **A**pparatus **M**ade for **B**road **E**mergencies

Group 33:
Jacob Wurm
Ismael Rivera
Jihang Li
Tommy Goris

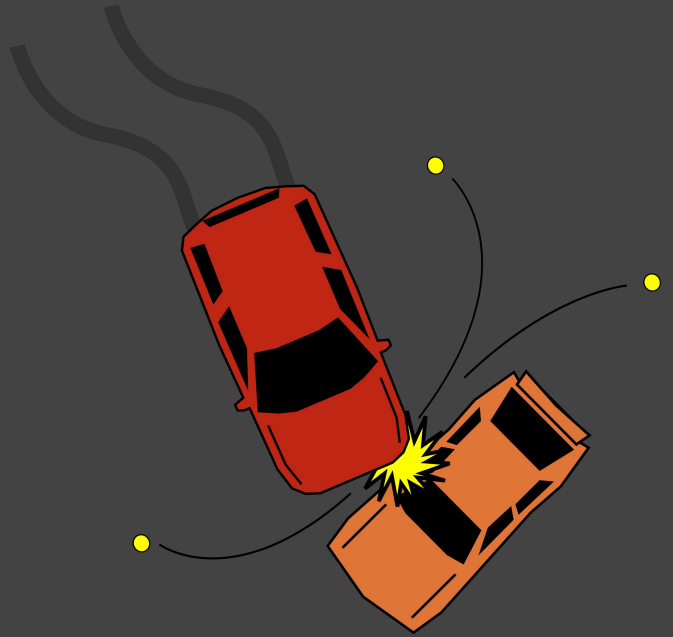
Motivation

- Motor vehicle traffic deaths
  - 35,398 deaths
  - 11.1 deaths per 100,000
- Eliminating the time between when an accident occurs and when first responders are dispatched to the scene decreases mortality rates by 6%



Motivation

- Wanted a device that would automatically contact emergency services in the event of an accident
- Wanted a system that could appeal to as broad an audience as possible
- Wanted something cheaper and more functional than existing solutions
  - OnStar
  - Automatic



Objectives

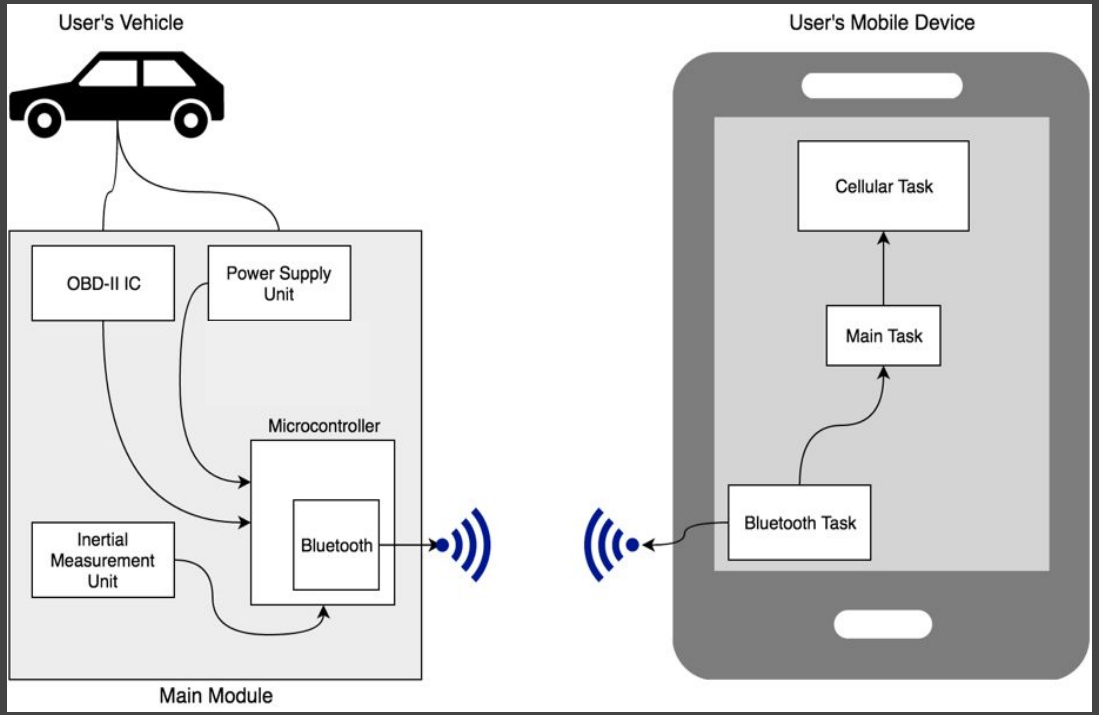
- Needs to continuously detect whether or not the the user has been in an accident
- Device needs to communicate with a mobile phone
  - To trigger response in mobile application
  - To contact emergency services
- User interaction must be minimal (plug ‘n play)
- Needs to be able to work in as many vehicles as possible
- Needs to comfortably fit under the dashboard of the vehicle
- Needs to communicate with the mobile phone in the event of power loss

Specifications and Requirements

- Extract data available from OBD-II and accelerometer
  - Velocity
  - Throttle Position
  - Engine RPM
  - Orientation
- Detect if the user has been in an accident based on sensor data
- Needs to remain powered on for at least 10s after power loss
- Needs to communicate with mobile device using Bluetooth protocol
  - Android
- Mobile application needs to be user-friendly and have a low impact on overall system performance
- Device footprint less than 2x5 inches
  - Needs to fit comfortably under the vehicle

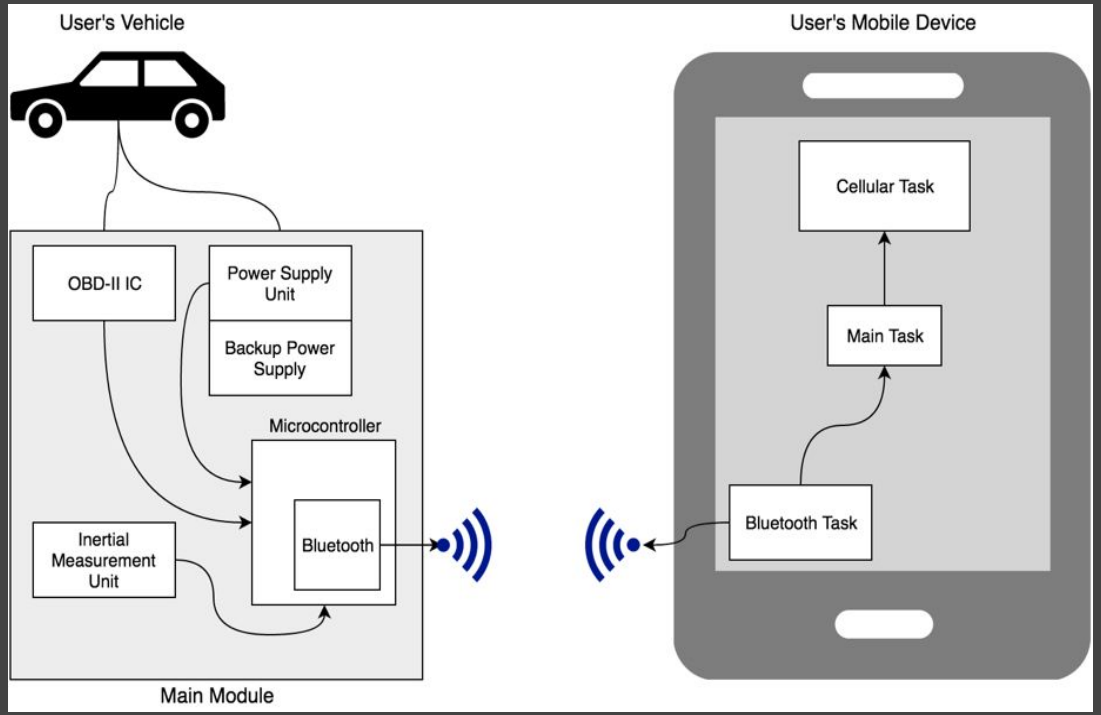
System Overview

- User plugs the device into the OBD-II port under their steering wheel
- Device is waiting to be paired with user's phone
- User opens phone app and pairs with the device
- Microcontroller begins gathering information from OBD-II and accelerometer



System Overview cont.

- Microcontroller is continuously determining whether the user has been in an accident
- If an accident occurs the Bluetooth module inside the microcontroller sends a signal to the phone
- Phone application calls emergency services



How can we tell the user has been in an accident?

Two sources of information:

- Vehicle diagnostic data (OBD-II)
- Vehicle acceleration, orientation (Accelerometer, Gyroscope)

Parameters that determine whether the user has been in an accident

- Sudden, large changes in acceleration (i.e. Slamming on the brakes)
- Abrupt changes in orientation (i.e. Car flips over)

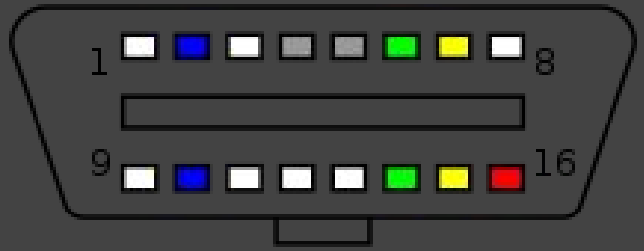
What is OBD-II? (On-board diagnostics)

A vehicle’s self diagnostic and reporting tool is known as the On-board diagnostic (OBD).

We can get valuable information from the OBD that can be used to detect a car crash.

Mode of operation determines the type of data the OBD will send back

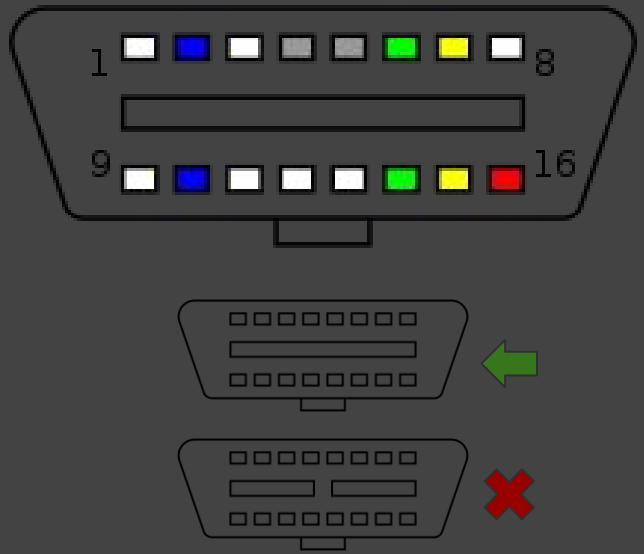
PID (Parameter IDs) codes are sent to the OBD in return for specific car data (i.e. RPM, velocity…).



Female OBD-II connector pinout – front view

OBD-II Pinout

1	Manufacturer Discretion	9	Manufacturer Discretion
2	Manufacturer Discretion	10	Manufacturer Discretion
3	Bus Positive SAE J1850	11	Bus Negative SAE J1850
4	Chassis Ground	12	Manufacturer Discretion
5	Signal Ground	13	Manufacturer Discretion
6	CAN-High	14	CAN-Low
7	K-Line	15	L-Line
8	Manufacturer Discretion	16	Battery Voltage



Some PID codes

Based on mode of operation $01, which is used for basic powertrain information.

PID code (HEX)	Data bytes returned	Description
0C	2	Engine RPM
0D	1	Vehicle Speed
0E	1	Timing advance
0F	1	Intake air temperature
10	2	MAF air flow rate
11	1	Throttle position
2F	1	Fuel Tank Level Input

Design Architecture: Power Source

Primary power supply options

- OBD-II port (12V)
- USB adapter (5V)

Ended up choosing OBD-II for the sake of convenience and single point of contact with the vehicle

Voltage Regulator

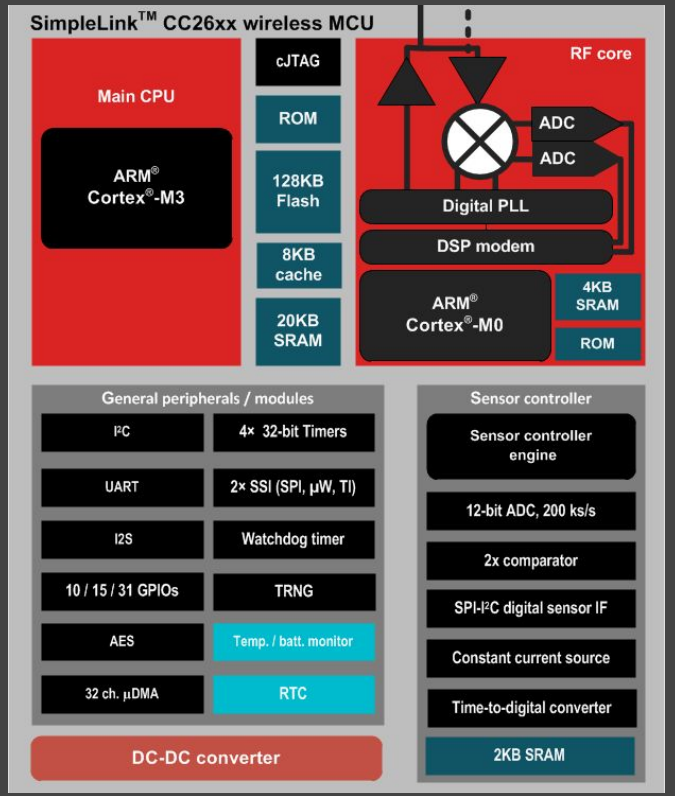
Variables	Linear Regulator	Switching Regulator
Family Name	LE33CZ-TR	LM2574HV
Vin min.	4.3V	4V
Vin max.	18V	40V
Efficiency	27.5%	72%
Cost	.66	2.76
Complexity	low	high

OBD-II Interpreters (ELM327 vs STN1110)

	ELM327 v1.4	STN1110
Architecture	8-bit	16-bit
Processing speed	4 MIPS	40 MIPS
FLASH (ROM)	32 KB	128 KB
RAM	1.5 KB	8 KB
Supports all OBD-II protocols	yes	yes
Supported UART baud rates	9600 bps to 500 kbps	38 bps to 10 Mbps
Price each, for 1000 units	$24	$10
Large OBD message memory buffer	no	Yes

Design Architecture: Microcontroller & BT

- Originally wanted to use external BT IC
  - Not secure
  - More effort for implementation
  - Not enough control over profiles
- Decided to use SoC that contains both
  - More security
  - More control over profiles
  - Simpler implementation
  - Unified development environment



Design Considerations: Bluetooth

- Classic
  - Higher Bandwidth
  - Higher Power Consumption
  - Proper Security
- Low Energy
  - Lower Bandwidth
  - Low power consumption
  - Poor to great security depending on implementation
- Decided to use BLE

Security Considerations: Bluetooth Low Energy

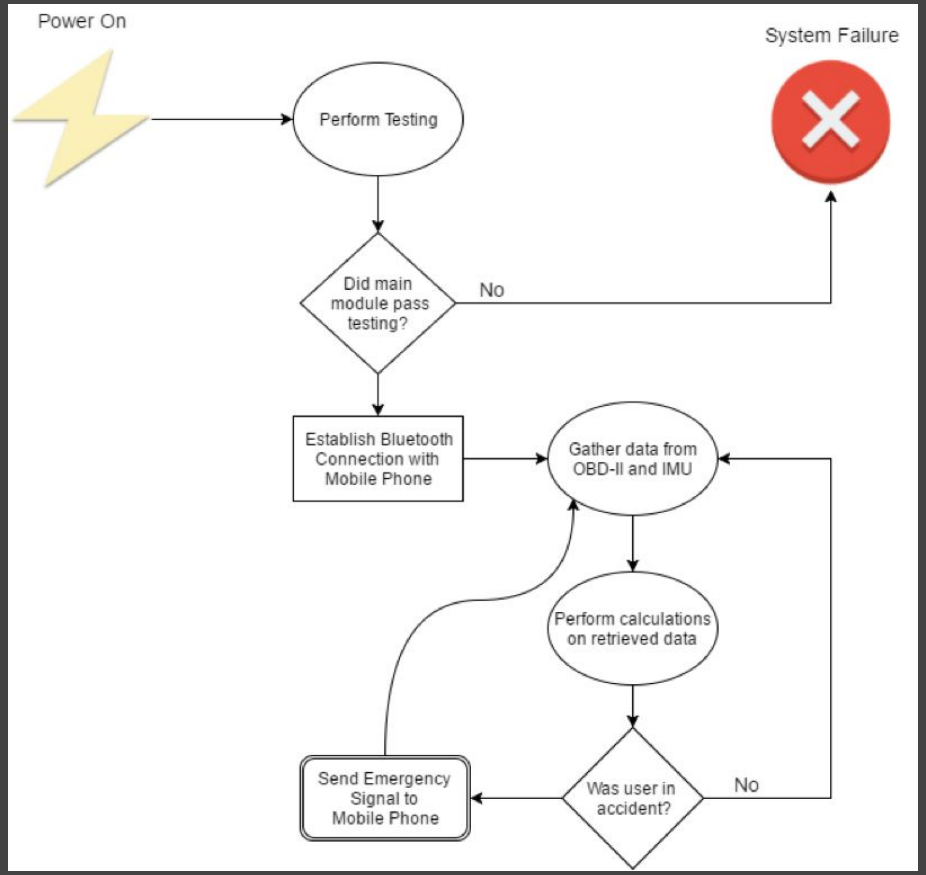
- LE Legacy (4.0, 4.1)
  - TK, STK
- LE Legacy Pairing Methods
  - Just Works™
  - OOB Pairing
  - Passkey
- LE Secure (4.2+)
  - LTK, ECDH
- LE Secure Pairing Methods
  - Just Works™
  - OOB Pairing
  - Passkey

Considerations: Operating System

- No OS a.k.a. *Super Loop*
  - Rely entirely on interrupts to guide execution
    - UART, I2C
  - No Multitasking
  - Manual Resource Management
  - Difficult data consistency
  - Unpredictable execution time
    - Nested ISRs

- Real-time Operating System (TI-RTOS)
  - Allows for multitasking
  - Overarching Scheduler
  - Set priorities for different tasks
    - Bluetooth Highest
  - Deterministic Behavior
  - Inter-task communication
  - Easier resource management
  - Predictable Memory Usage

Embedded High-Level Software Flow



Design Considerations: Mobile Application

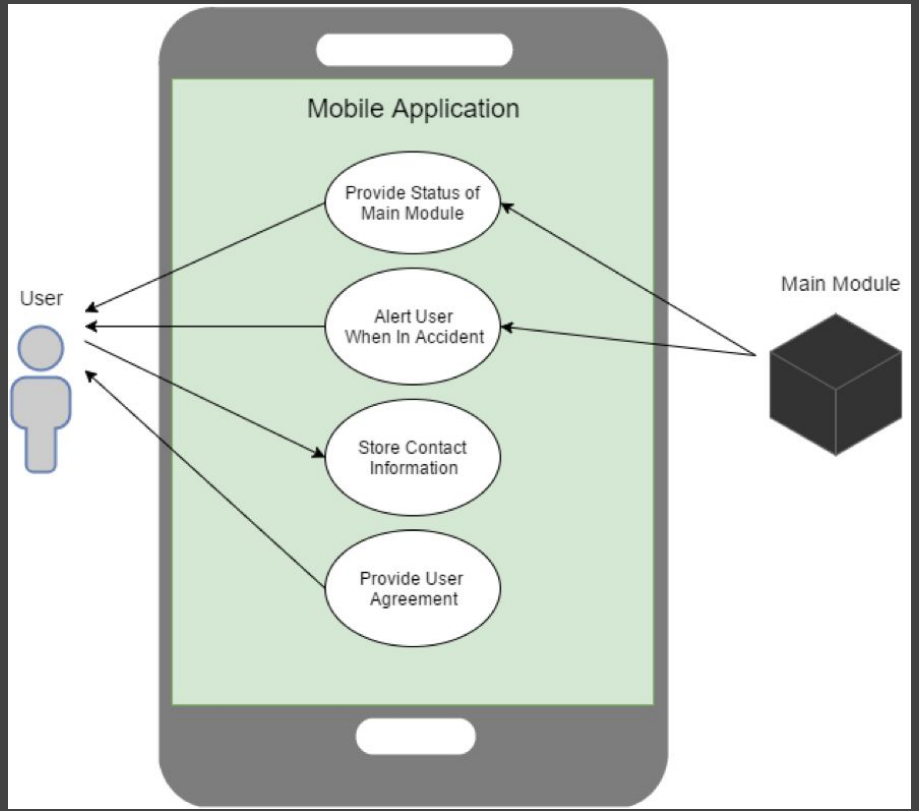
- Wanted to appeal to the largest user base
  - Android
- Also want to support the largest amount of devices possible
  - For BLE 4.2 the lowest API level is 18

Period	Android	iOS	Windows Phone	Others
2015Q4	79.6%	18.7%	1.2%	0.5%
2016Q1	83.5%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%
2016Q3	86.8%	12.5%	0.3%	0.4%

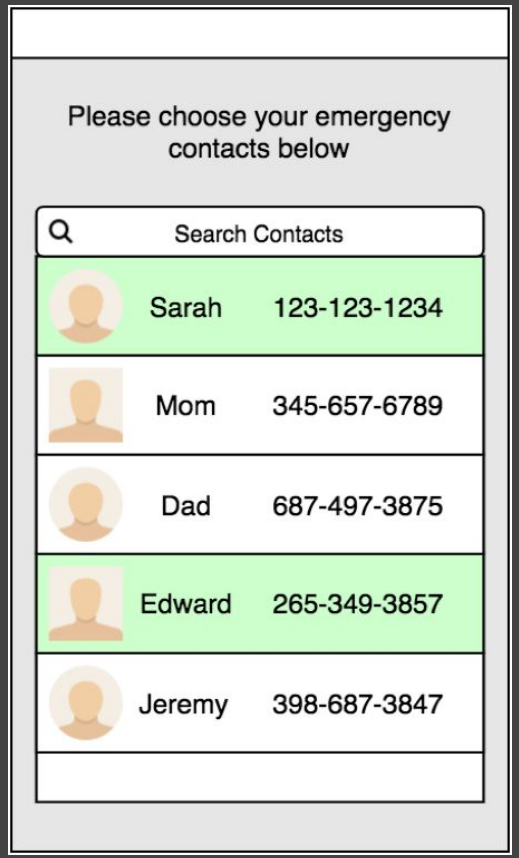
Source: IDC, Nov 2016

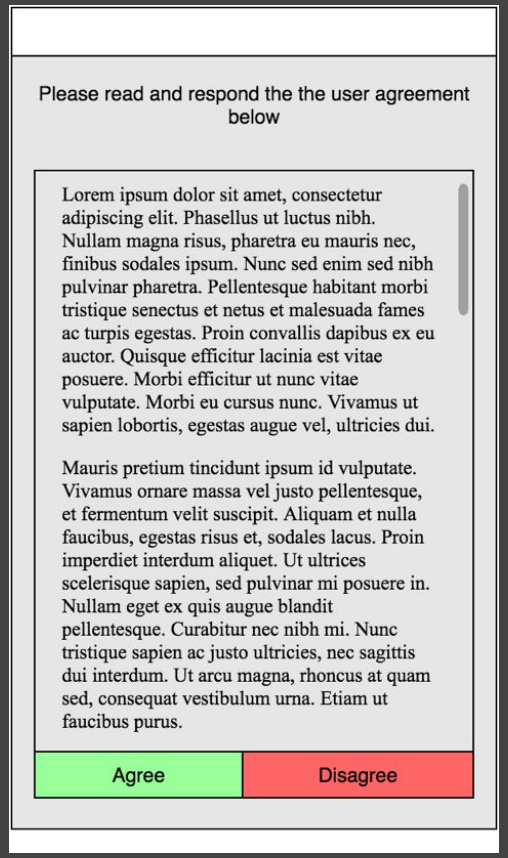
Mobile Application Software Design

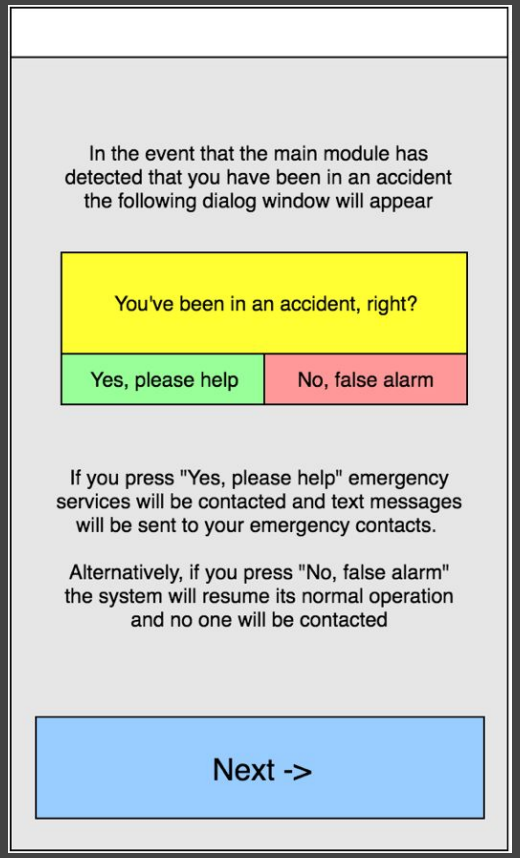
- Android Application
  1. Contact emergency services
  2. Contact friends/families



Mobile Application Interface Mockups

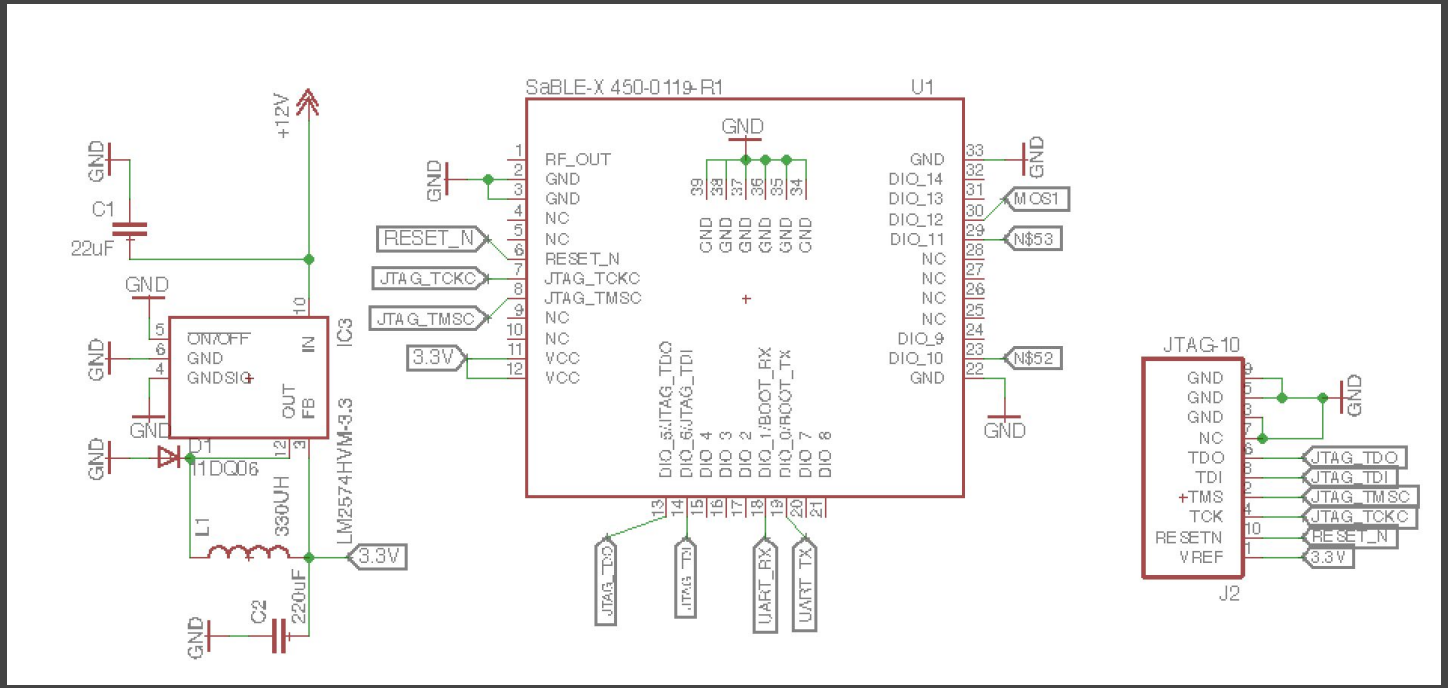






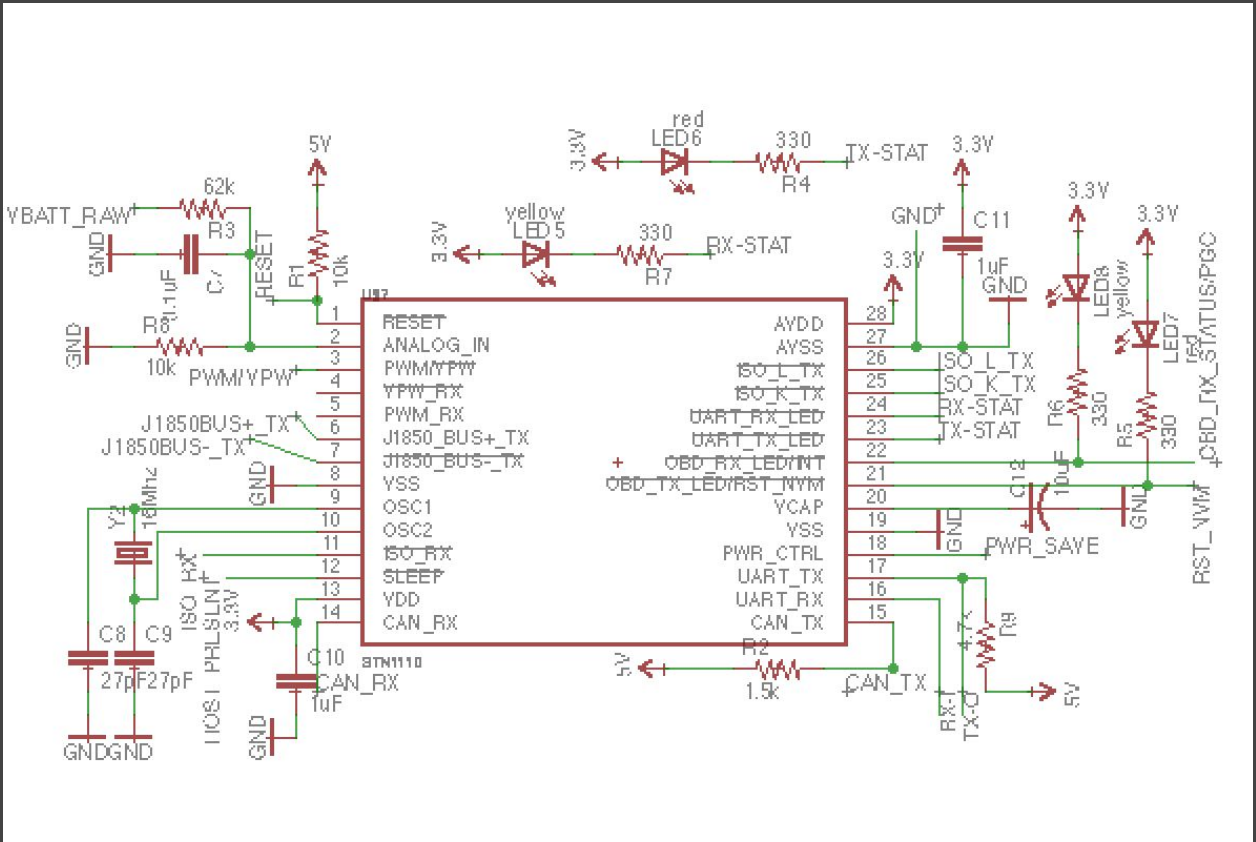
Specifications and Requirements

MCU / Bluetooth Module



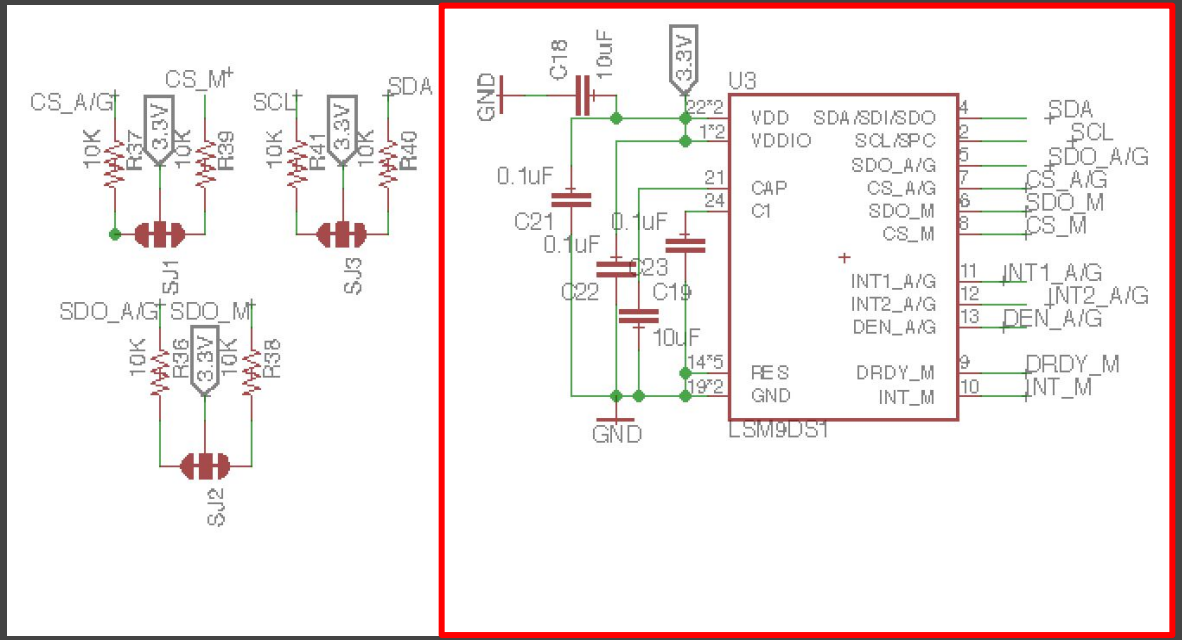
Specifications and Requirements

OBD-II



Specifications and Requirements

Accelerometer (LSM9DS1):



Administrative Details

Bill of Materials

Category	Name	Cost
Switching Regulator	LM2574N-3.3	$2.76
Switching Regulator	LM2574N-5.0	$2.76
CAN Transceiver	MCP2551	$2.10
MIcrocontroller / BLE	SABLE-X	$16.52
OBD to UART IC	STN1110	$10.99
PCB Fabrications	---	$30.00
Circuitry	Res, Cap, Ind, etc.	$18.00
Total	---	$83.13

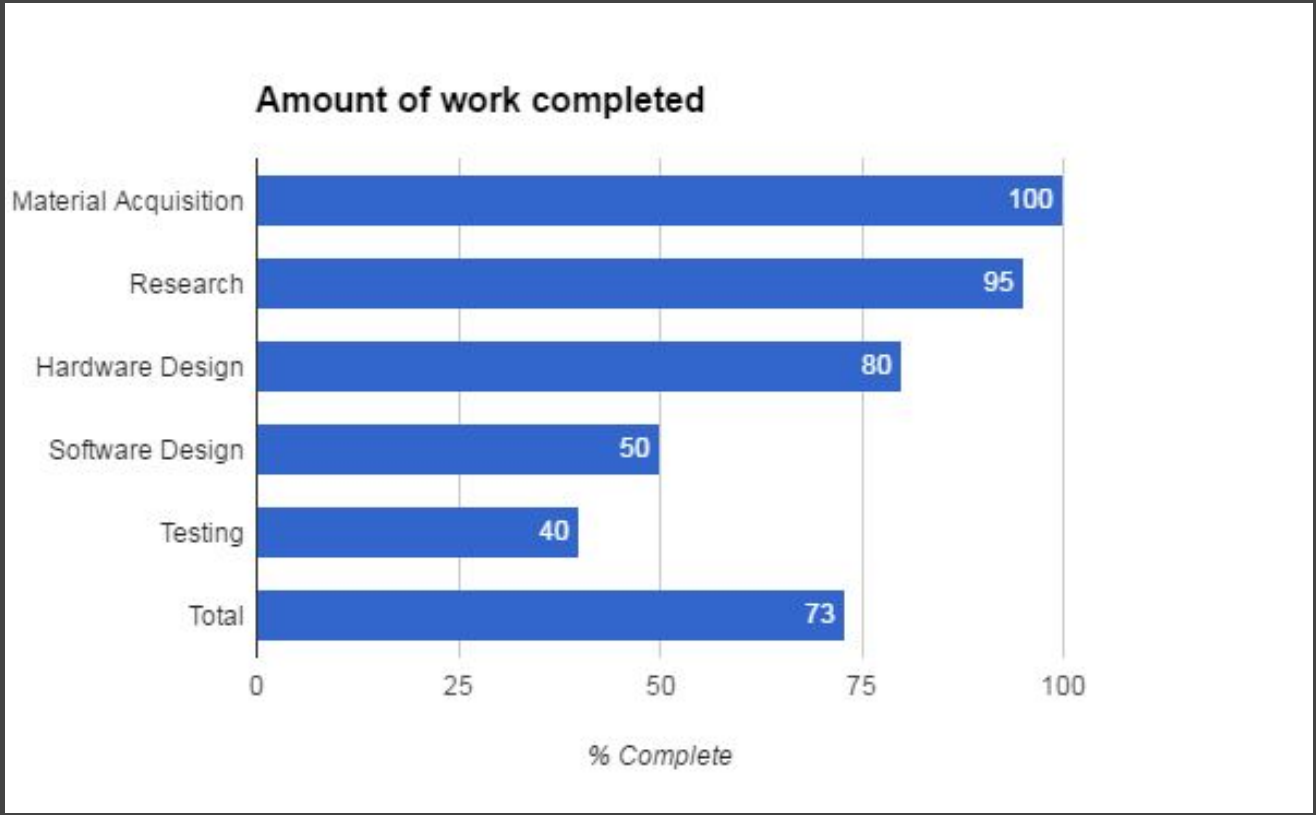
Overall Budget

Type	Cost
Scraped Components	$30.00
Breakout / Development	$200.00
Prototyping	$100.00
Spare Components	$50.00
Total	$380.00

Work Distribution

Name	Major	Tasking
Jacob Wurm	Computer Engineering	Microcontroller Programming
Ismael Rivera	Electrical Engineering	Power Systems Development / PCB Design
Tommy Goris	Computer Engineering	Android Application Development
Jihang Li	Electrical Engineering	Power Systems Development / PCB Design

Progress


Amount of work completed
Material Acquisition
Research
Hardware Design
Software Design
Testing
Total
100
95
80
50
40
73
0
25
50
75
100
% Complete

Questions?