



# Progress Review II

Group 8:

<b>Choong Jin Ng</b>	<i>Project Lead</i>
<b>Takehiro Tanaka</b>	<i>App Design Engineer</i>
<b>Enes Yazici</b>	<i>App Design Engineer</i>
<b>Win Aung</b>	<i>Hardware Engineer</i>
<b>Nicholas Lau</b>	<i>Software Engineer</i>
<b>Ranjoat Chana</b>	<i>Systems Engineer, Communication Officer</i>



# Agenda

## Items for Discussion:

- Approval of the agenda and minutes of first progress review meeting held on June 10th, 2020 (1 min)
- Business arising from the minutes of the previous meeting (1 min)
- Clarification on OBD-II History (1 min)
- Introduction outlining basic review and any changes in scope (1 min)
- Progress, outlining completed research and planning (4 mins)
- Risk and remediations (1.5 mins)
- Summary of current state of project (0.5 min)
- Question period and follow-up discussion (10 mins)

# Approval of Minutes Progress Review Meeting I (June 10th, 2020)

Revisions as per instructor feedback.

Access:

<https://docs.google.com/document/d/11FljHZPkfOvQZikw2MDsO0bJIYVXZhjqDZQs1nxVHB0/edit#>





# Changes to Previous Meeting Minutes

- Action items are highlighted
  - New summary of action items
- Assigned action items to individuals to be held responsible
- The following comments made by participants corrected
  - Creative commons licensing
  - “Some internal companies”
  - “Odometer” feature



# Business Arising

**Action:** Ranjoat will share a powerpoint with the team to collectively add slides for the next progress review meeting by July 6th (complete)

**Action:** Jin will verify that Evo car sharing does not block access to OBD-II port for purposes of testing by week of July 6th (complete)

**\*Action:** Jin will research the licensing options for hardware and software by June 14th for requirements document

**\*Action:** Nicholas will research and investigate options for protocols and supported extensions by week of July 6th for design document preparation

**\*Action:** Ranjoat will research adding maintenance features and service updates based on odometer readings in software application by week of July 6th for design document preparation

*\*research based actions will be further discussed in Progress & Research section*



# Clarification on OBD-II Maintenance

- OBD-II does not access actual vehicle's odometer reading
  - Privacy concerns
  - Liability issues with mishandling
  - Inaccurate
  - Most cars that prompt their users to service their cars are not common
  - Target audience usually ignores OEM's scheduled maintenance
    - Too conservative or too generous
- We will not develop a feature that suggests a maintenance to the user



# Clarification on OBD-II History

"CAN bus [...] was officially released in 1986 at the Society of Automotive Engineers (SAE) conference in Detroit, Michigan. [...] CAN bus is one of five protocols used in the on-board diagnostics (OBD)-II vehicle diagnostics standard. The OBD-II standard has been mandatory for all cars and light trucks sold in the United States since 1996." - Wikipedia

This paragraph is misleading. Wikipedia's 'On-board diagnostics' History implies a different history:

- 1996: The OBD-II specification is made mandatory for all cars sold in the United States.
- 2008: All cars sold in the United States are required to use the **ISO 15765-4**<sup>[5]</sup> signaling standard (a variant of the **Controller Area Network (CAN) bus**).

Reference: [https://en.wikipedia.org/wiki/On-board\\_diagnostics#History](https://en.wikipedia.org/wiki/On-board_diagnostics#History)



# Clarification on OBD-II History

- Although CAN was originally developed for passengers cars, CAN was first applied to other systems first (elevators, textile machines, X-ray machine, etc.)
  - Most first applications, including cars, were done in Europe
  - Although CAN was introduced by Bosch in Feb 1996, there were many problems (errors, incompleteness, political disputes, etc)
- ISO 15765-4 (i.e CAN BUS) was formally specified in 2005.
- All cars manufactured after 2008 in North America should only use CAN (ISO 15765-4)
  - When we refer to 'CAN BUS Reader', we are talking about an OBD-II reader that works on cars after 2008

Reference:

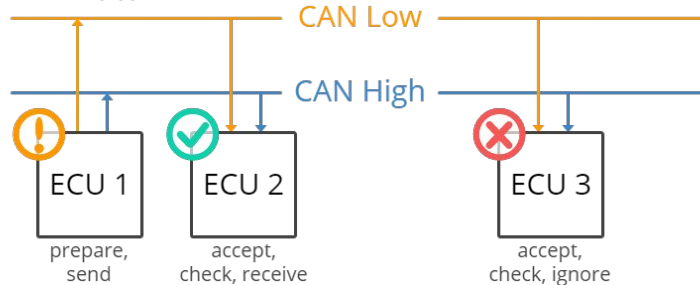
<https://www.can-cia.org/can-knowledge/can/can-history/>

<https://www.iso.org/standard/33619.html>



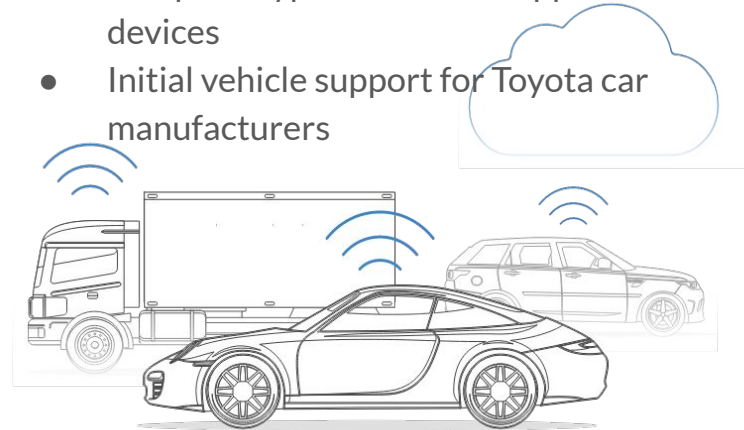
# Introduction & Background

- The CAN bus system enables communication between the ECUs in a vehicle system
- CAN provides the user with full access to car data whereas OBD2 provides limited data

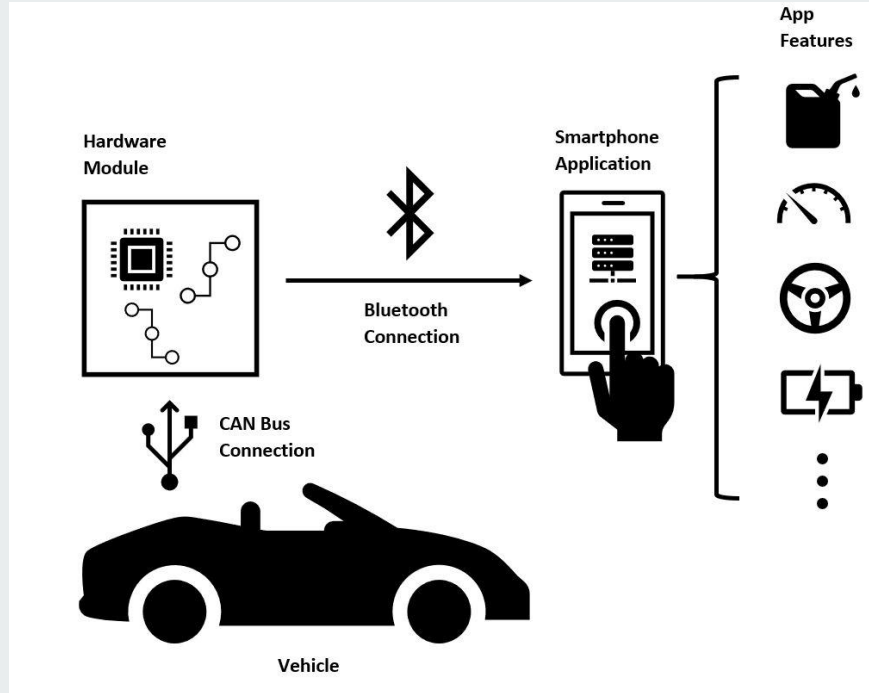


## Changes in scope:

- Our prototype device will support android devices
- Initial vehicle support for Toyota car manufacturers



# System Overview



-Our CANBus reader is designed to connect to the OBD-II port in a vehicle

-The hardware module is comprised of a microcontroller and Bluetooth adapter to process and transmit data from the vehicle CAN bus system (CAN low, CAN high ) to the user's android device

-The smart phone app will allow the user to read DTCs and live sensor data such as oil temperature, engine speed, GPS, fuel rate and speed, etc.

## Progress & Research - Hardware



- Arduino Uno with SparkFun CAN BUS Shield
- Using SparkFun's internal libraries
  - Example code is a useful jumping ground for a Proof-of-Concept
- Connect the CAN BUS Shield to a OBD-II port using a DB9 to OBD-II connector



## Progress & Research - Data Sample

```
1 18:02:25.892 -> CAN Read - Testing reception of CAN Bu
2 18:02:26.876 -> CAN Init ok
3 18:02:27.861 -> ID: 7D8, Data: 84 61 62 3 0 0 0 0
4 18:02:27.908 -> ID: 7C8, Data: 83 61 29 16 0 0 0 0
5 18:02:27.955 -> ID: 7D8, Data: 84 61 62 3 0 0 0 0
6 18:02:28.002 -> ID: 750, Data: 840 2 21 A3 0 0 0 0
7 18:02:28.049 -> ID: 7C0, Data: 82 21 21 0 0 0 0 0
8 18:02:28.049 -> ID: 7D0, Data: 82 21 62 0 0 0 0 0
9 18:02:28.096 -> ID: 7C8, Data: 83 61 29 16 0 0 0 0
10 18:02:28.142 -> ID: 7D8, Data: 84 61 62 3 0 0 0 0
11 18:02:28.189 -> ID: 758, Data: 840 4 61 A2 20 0 0 0
12 18:02:28.236 -> ID: 7C0, Data: 82 21 29 0 0 0 0 0
13 18:02:28.236 -> ID: 7C0, Data: 82 21 21 0 0 0 0 0
14 18:02:28.283 -> ID: 7E0, Data: 82 21 28 0 0 0 0 0
15 18:02:28.330 -> ID: 7D0, Data: 82 21 62 0 0 0 0 0
16 18:02:28.377 -> ID: 750, Data: 840 2 21 A4 0 0 0 0
17 18:02:28.424 -> ID: 7C0, Data: 82 21 21 0 0 0 0 0
18 18:02:28.471 -> ID: 750, Data: 840 2 21 A5 0 0 0 0
19 18:02:28.471 -> ID: 7C8, Data: 83 61 29 16 0 0 0 0
20 18:02:28.517 -> ID: 4E0, Data: 824 0 44 1 0 0 0 0
```

- Data sample from Toyota Prius (Evo car sharing) using demo code
- Able to read data frames that are sent to the Arduino
- Suspect that not all possible messages are not being read because example code does not request for them



# Progress & Research - Data Sample

```
17:59:34.860 -> CAN-Bus Demo
17:59:34.860 -> CAN Init ok
17:59:35.844 -> Vehicle Speed:
17:59:36.828 -> Engine RPM:
17:59:37.860 -> Throttle:
17:59:38.844 -> Engine Coolant Temp: 15 %
17:59:39.829 -> O2 Voltage: 15 %
17:59:40.848 -> MAF Sensor: 15 %
17:59:41.832 -> Vehicle Speed: 0 g/s
17:59:42.864 -> Engine RPM: 0 km
17:59:43.848 -> Throttle: 0 km
17:59:44.832 -> Engine Coolant Temp: 15 %
17:59:45.863 -> O2 Voltage: 60 degC
17:59:46.848 -> MAF Sensor: 60 degC
17:59:47.866 -> Vehicle Speed: 0 g/s
17:59:48.851 -> Engine RPM: 0 km
17:59:49.869 -> Throttle: 0 rpm
17:59:50.853 -> Engine Coolant Temp: 15 %
17:59:51.838 -> O2 Voltage: 60 degC
17:59:52.856 -> MAF Sensor: 60 degC
17:59:53.864 -> Vehicle Speed: 0 g/s
17:59:54.829 -> Engine RPM: 0 km
17:59:55.861 -> Throttle: 0 rpm
```

- Another data sample from Toyota Prius (Evo car sharing) using demo code
- Code requests for
  - Vehicle Speed
  - Engine RPM
  - Throttle
  - Engine Coolant Temp
  - O2 Voltage
  - MAF Sensor
- Promising though highlights the needs to check the data's validity before displaying



# Progress & Research - Evo Car Sharing

## How do I sign up?

[Signing up](#) is quick and easy. Just register online, then call ICBC or your licensing authority and have them send us your driving record for the past 2 years. Once approved you'll get your Evo card in the mail within 5-7 days. But you don't need to wait for your card, since you can [download the App](#) as soon as you're approved and hit the road right away.

## What is the Driving Record Scoring Criteria?

We want to make sure Evo drivers are safe on the road. So we need to check your driving record before we approve you. To meet the basic requirements drivers must:

- Be at least 18 years of age
- Have a Minimum N licence (class 7) with 2 years driving experience (including Learners)
- Not have more than 2 traffic violations in the past 24 months
- Not have any major driving violations (6+ points) or any offences that go under the Criminal Code of Canada
- Not have any current suspensions

If you are unsure if you have any traffic violations or points against your licence, check them through ICBC [here](#).

## • Advantages

- Able to test on a standardised car
  - Toyota Prius
- Available through the Lower Mainland
- Low operating rates

## • Disadvantages

- Need a valid driving license (Class 7 or N at minimum)
- Need to have 2 years of driving experience
- Not convenient for bursts of development and testing
- Alternative solution (CAN BUS Simulator) is more convenient



# Progress & Research - Protocols

## Research - Protocols (CANBUS)

The listed ISO standard is the intended focus protocol for CANBUS

- **ISO 15765-4 - Emissions and external data access (Diagnostic Communication over CAN)**
  - ISO 11898 series- Physical and Data Link Layers (frame format)
    - Part 1 - Physical and Data Link Layer
    - Part 2 - High Speed Transmission
  - ISO 15765-2 - Network and Transport Layer
  - Also implemented in ISO 27145-4 for WWH-OBd

Table 1 — Enhanced and legislated OBD diagnostic specifications applicable to the OSI layers

OSI 7 layers <sup>a</sup>	Vehicle-manufacturer-enhanced diagnostics	Legislated OBD (on-board diagnostics)		Legislated WWH-OBd (on-board diagnostics)	
Application (layer 7)	ISO 14229-1, ISO 14229-3	ISO 15031-5		ISO 27145-3, ISO 14229-1	
Presentation (layer 6)	Vehicle manufacturer specific	ISO 15031-2, ISO 15031-5, ISO 15031-6, SAE J1930-DA, SAE J1979-DA, SAE J2012-DA		ISO 27145-2, SAE 1930-DA, SAE J1979-DA, SAE J2012-DA, SAE J1939-DA (SPNs), SAE J1939-73 Appendix A (FMIs)	
Session (layer 5)		ISO 14229-2			
Transport protocol (layer 4)	ISO 15765-2	ISO 15765-2	ISO 15765-4	ISO 15765-4, ISO 15765-2	ISO 27145-4
Network (layer 3)				ISO 15765-4, ISO 11898-1	
Data link (layer 2)	ISO 11898-1	ISO 11898-1		ISO 11898-1, ISO 11898-2	
Physical (layer 1)	ISO 11898-1, ISO 11898-2, ISO 11898-3, or vehicle manufacturer specific	ISO 11898-1, ISO 11898-2			

<sup>a</sup> 7 layers according to ISO/IEC 7498-1 and ISO/IEC 10731

<sup>a</sup> 7 layers according to ISO/IEC 7498-1 and ISO/IEC 10731





# Progress & Research - CAN Data

## Research - CAN IDs and Messages

Depending on car, messages are either read automatically in intervals or must be requested manually

Identified different ways to decode CAN message format since it proprietary to Original Equipment manufacturer (OEM):

- Reverse Engineer
  - Perform different actions on car to see which IDs correlate to a specific function
  - Tedious and time consuming
- Look-up CAN IDs
  - Identified a subset of IDs for Toyota cars (eg. vehicle speed, RPM)
  - Fast and convenient



# Progress & Research - Data

## CAN - Raw Data Sample and Translation

ID: 610, Data: 820 0 0 64 C2 0 0 0

HEX 610 - Vehicle Speed

HEX 0 (Byte 3) - 0 km/h

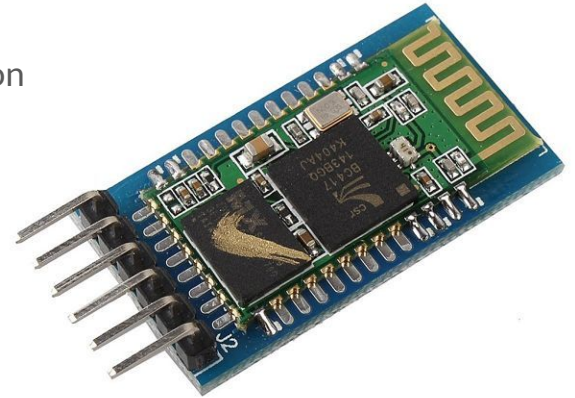
ID: 63B, Data: 816 0 43 4E 0 0 2 7F

Unknown, Need further research or reverse engineering

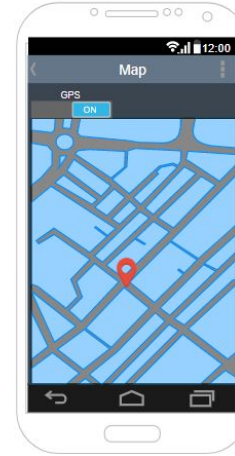
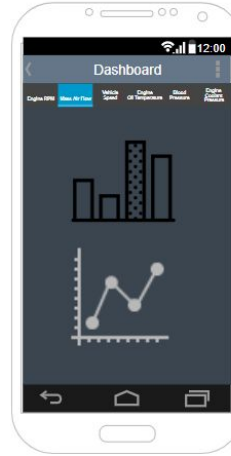
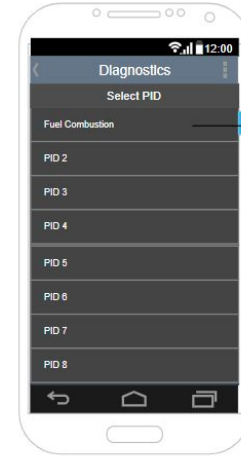
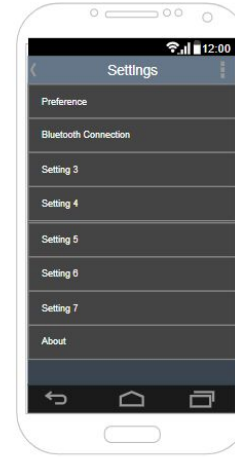
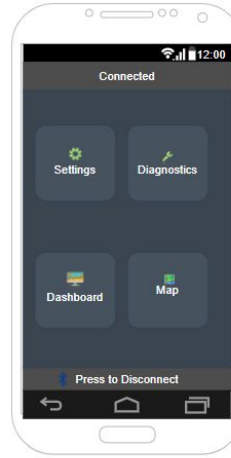
# Progress & Research - Bluetooth

## Bluetooth Modules

- **HC-05** -based on Bluetooth 2.0 and offers master/slave configuration
- **HC-06** -based on Bluetooth 2.0 and offers only slave configuration
  - Both HC-05 and HC-06 are compatible with android devices only
- **HM-10/HM-11** -based on Bluetooth 4.0 BLE
  - compatible with iphone and android 4.3+
  - intended for low power consumption applications



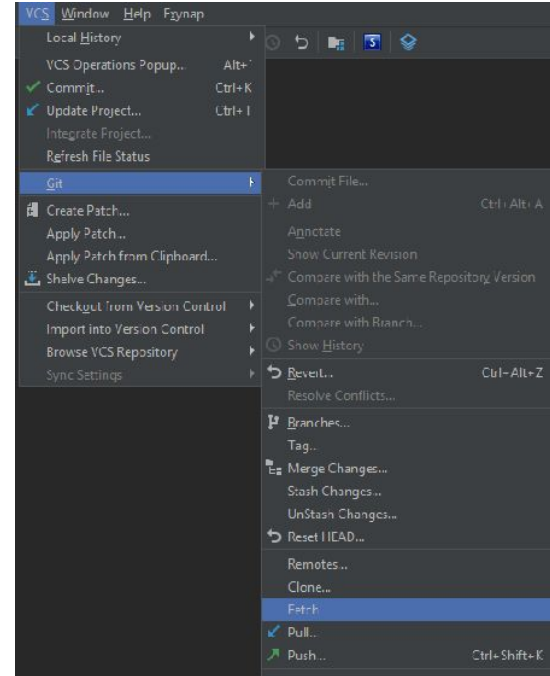
# Progress & Research - App



# Progress & Research - App

## App Planning

- Android Studio
  - o XML, Java, Git
- Test on different devices (e.g. Screen)
  - o Physical device
  - o Virtual device (Emulator)





# Progress & Research - App

## Design Choices

- Easy to use (no difficult learning curve for users)
- Separate tabs makes development easier
  - o Separate XML activities and Java classes(no dependency between different functionalities)
  - o More organized

## Colour & Image Choices

- Do not use a colour that is too bright
  - o May hurt eyes & Annoying
- Use common color and icon associations (green for connect, red for disconnect etc.)



# Progress & Research - App

## Software App Progress

- Currently
  - o Converting sketches to app ui
    - Built the app basic display (main screen only)
  - o Working on the logical components
    - Enabling bluetooth, searching devices, pairing devices etc.
- Next steps
  - o Attach to arduino to receive data
  - o Parse the data
  - o Implement Individual UI components ( dashboard, chart, map etc)

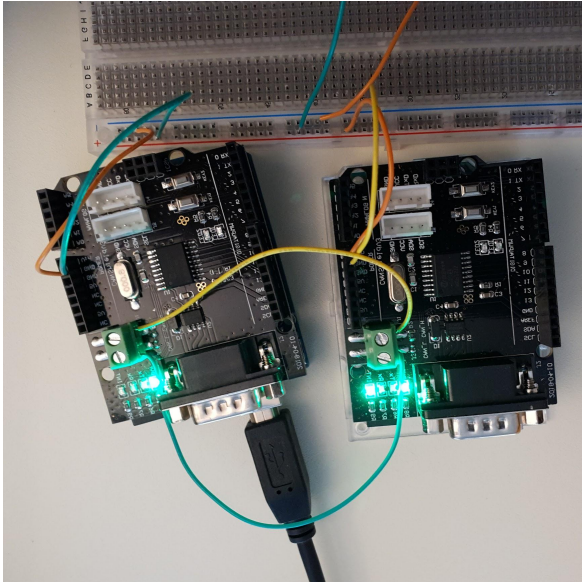


# Risk and Remediation - Scheduling

- To aid scheduling
  - Planning to meet up in person for hardware setup of prototype design
  - Regular meetings and discussion over online
  - Reallocate resources as required for hardware and software component

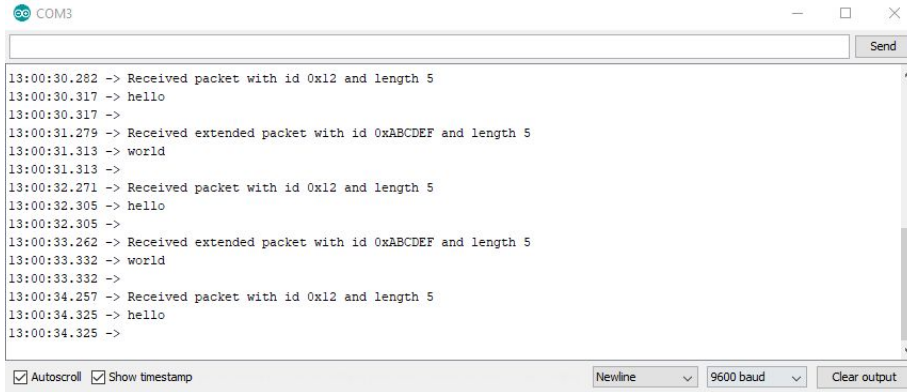


# Risk and Remediation - CAN BUS Simulator



- Use another Arduino and CAN BUS Shield to act as a “car simulator”
- CAN-High connects to CAN-High, and CAN-Low connects to CAN-low
- Run a pre-defined script that simulates how the car’s internal CAN BUS system works
- Sends the appropriate messages to the reader when requested

# Risk and Remediation - CAN BUS Simulator



```
13:00:30.282 -> Received packet with id 0x12 and length 5
13:00:30.317 -> hello
13:00:30.317 ->
13:00:31.279 -> Received extended packet with id 0xABCDEF and length 5
13:00:31.313 -> world
13:00:31.313 ->
13:00:32.271 -> Received packet with id 0x12 and length 5
13:00:32.305 -> hello
13:00:32.305 ->
13:00:33.262 -> Received extended packet with id 0xABCDEF and length 5
13:00:33.332 -> world
13:00:33.332 ->
13:00:34.257 -> Received packet with id 0x12 and length 5
13:00:34.325 -> hello
13:00:34.325 ->
```

- Sender sends the string 'hello' and 'world' to the reader
- Reader simply prints the packet's contents to serial
- Future work will enable those who do not have cars to work at home
- Also able to have repeatable verification tests



# Risk and Remediation - Licensing

- Hardware and related documents falls under CERN Open Hardware License Version 2 - Permissive
  - Referred to as CERN-OHL-P
  - Only applies to hardware and its associated documentation
- Software and related documents falls under MIT License
- Goal of this product is to make working on your car easier, not necessarily to turn a profit
- Customers prefer to stick with official vendors and active customer support



# Summary

## Current State:

- Arduino CAN bus shield and demo code has allowed for successful connection to test vehicles to extract data frame samples
- Basic app components and main screen options determined
- Familiarized with the protocols and standards (incl. licensing) for our prototype design
- Determined best practice to verify our product using a simulator
  - Evo car sharing is an option



# Summary

## Next Steps:

- Working towards finalizing the UI design and functionality
- Establish a connection with the android phone via selected bluetooth module
- Ensure app is correctly mapping data samples to selected features for prototype design stage



**Questions?**



**Thank You**



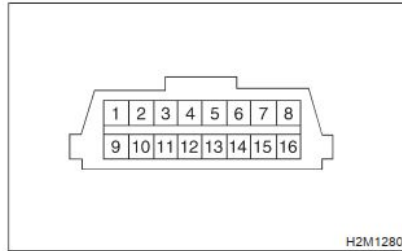
## Appendix: Clarification on OBD-II History

- A (modern) OBD-II vehicle uses the five communication protocols:
  - SAE J1850 PWM
  - SAE J1850 VPW
  - ISO 9141-2
  - ISO 14230-4
  - ISO 15765-4/SAE J2480 (since 2003) - i.e CAN

Reference: [http://www.obdtester.com/obd2\\_protocols](http://www.obdtester.com/obd2_protocols)



## Appendix: Clarification on OBD-II History



Terminal No.	Contents	Terminal No.	Contents
1	Power supply	9	Blank
2	Blank	10	K line of ISO 9141 CARB
3	Blank	11	Blank
4	Subaru Select Monitor signal (ECM to Subaru Select Monitor)*	12	Ground
5	Subaru Select Monitor signal (Subaru Select Monitor to ECM)*	13	Ground
6	Line end check signal 1	14	Blank
7	Blank	15	Blank
8	Line end check signal 2	16	Blank

\*: Circuit only for Subaru Select Monitor

- Even then, not every car uses all five communication protocols
- This OBD-II diagram is taken from a service manual for a Subaru Impreza 1996-2001 sold in North America
  - Note that only OBD- II's ISO 9141 is used
- Data collected by an OBD-II reader is slower than CAN BUS



# Appendix: Progress & Research - Protocols

## Research - Protocols (CANBUS)

Additional listed ISO standards are of focus for OBD2 PID-related data

- ISO 14229-2 - Session Layer (Unified Diagnostic Services)
- ISO 15031-2, -5, -6 - Application and Presentation Layer
  - Part 2 - Definitions and Terms
  - **Part 5 - Emissions Diagnostic Services (OBD2 PIDs)**
  - Part 6 - Diagnostic Trouble Codes
  - Based on SAE J1930-DA, **J1979-DA** and J2012-DA OBD

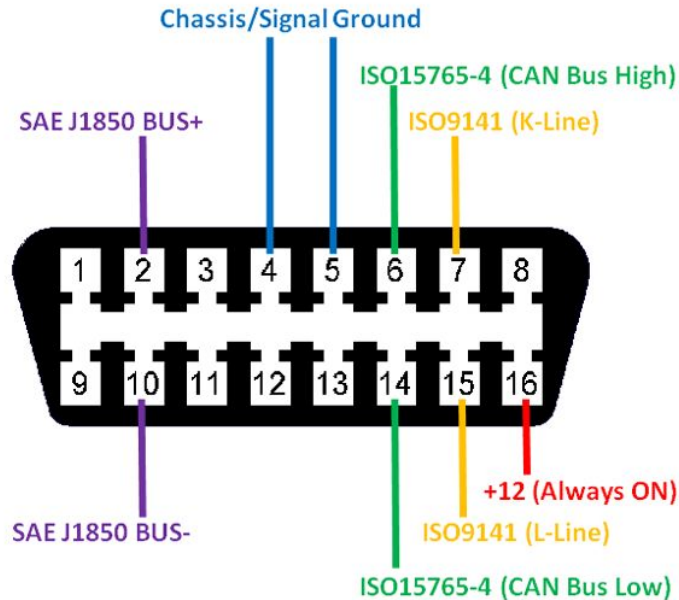


## Appendix: Progress & Research - Protocols

- Intended network stack is the legislated OBD
- Additional manufacturer specific codes will need to be identified

The International Organization for Standardization, "ISO 15765-4:2016(en), Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 4: Requirements for emissions-related systems," 2016. [Online]. Available: <https://www.iso.org/obp/ui/#iso:std:iso:15765:-4:ed-3:v1:en>. [Accessed 2 July 2020].

## Appendix: Progress & Research - OBD-II



- The important data lines we will use are pins 6 and 14 (CAN High and CAN Low respectively)
- We will also use Pins 12 and 4 to power the reader
  - Pin 4 and 5 is Chassis and Signal Ground respectively

# Appendix: Progress & Research

## Current version of the app

- Status bar on the top
  - o Logic needs to be added
- Connection bar on the bottom
  - o Logic needs to be added
- Card view layout implemented
  - o Needs to be adjusted (padding etc.)
  - o Logic needs to be added

