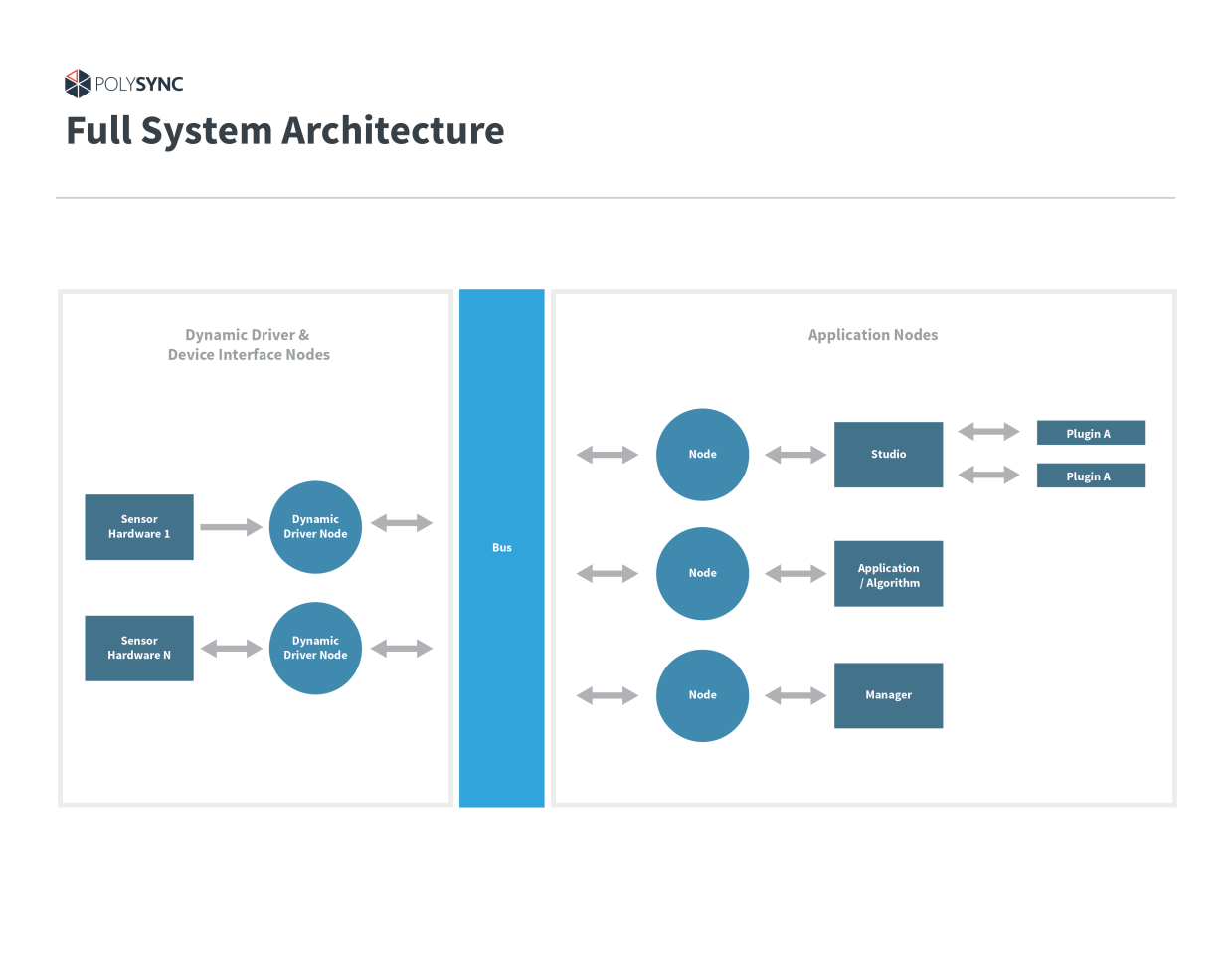
מגישים :

* ולינטין בונדר - 314515156
* דור אידיבו - 205566433
* רון בנטולילה - 205521206

PolySync – זו פלטפורמה (סוג של מערכת הפעלה אוטונומית) מנגישה שירותים נפוצים בקרב אפליקציות של רכבים אוטונומיים, פלטפורמה זו מאפשרת למפתחים להתעסק בדברים החשובים כמו האלגוריתמים כאשר הם מפתחים אפליקציות לרכבים אוטונומיים.

ארכיטקטורת המערכת:

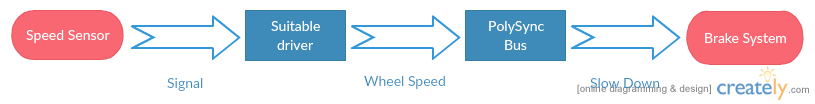
הרכב האוטונומי מורכב מהרבה מאוד מערכות שצריכות לעבוד בתיאום מושלם, התמונה הבאה מראה את ארכיטקטורת המערכת:

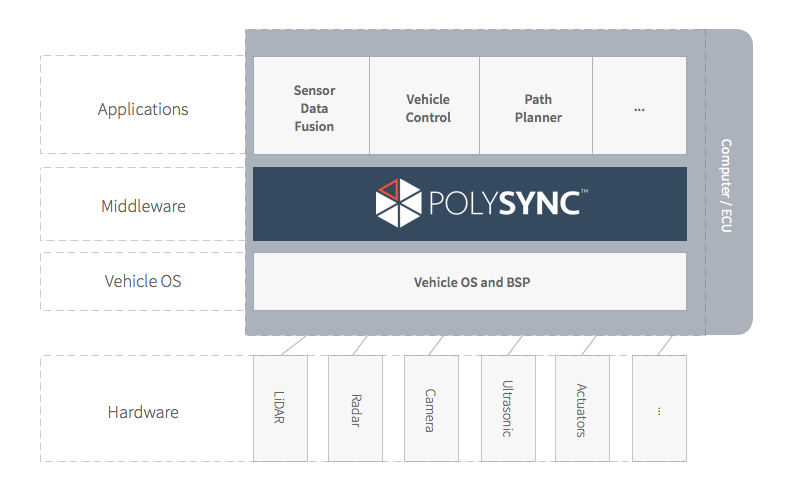
**Application Nodes:** אפליקציות שרצות על מערכת מסוימת ומתקשרות דרך שכבת ה- PolySync.

**Dynamic Driver & Device Interface Nodes: שכבה זו מכילה את הסנסורים (חיישנים) והדרייברים המתאימים לכל סנסור. בעצם זה הinput ל PolySync Bus.**

1. **חיישנים פיזיים – חיישנים שמדווחים על הרכב (לדוגמה: לחץ הבלם, זווית ההגה, מהירות גלגלים וכו..**(
2. Dynamic Drivers**- אחראים על התממשקות עם חומרה נתמכת ולעתים קרובות כוללות תמיכה לאבחון, פרמטרים, וסוגי נתונים מיוחדים.**

**PolySync Bus: ערוץ משותף (מבוסס** Ethernet wire**) לכל הצמתים המוגדרים בזמן ריצה, זה המקום בו הודעות עוברות בין ה**nodes **(הסבר בהמשך).**

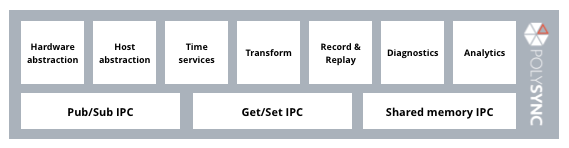
**Flow** לדוגמא:

אפשר להסתכל על PolySync כסוג של שכבה המפרידה בין אפליקציות הרכב מול מערכת ההפעלה שלו והחומרה הפיזית כמו שניתן לראות בדיאגרמה.

לפני שנסביר את ארכיטקטורת המערכת נגדיר את המושג הבא:

* צומת – אפליקציות שרצות על מערכת מסוימת ומתקשרות דרך שכבת ה- PolySync.

ארכיטקטורת PolySync – Middleware:



השכבה התחתונה (המסומנת בצבע ירוק) מכילה מספר סוגים של שירותים של תקשורת פנים תהליך (Inter-Process communications):

1. Publish/Subscriber – ארכיטקטורת תקשורת בה צומת יכול לשדר הודעות להרבה צמתים אחרים שמאזינים במקביל וכל צומת יכול להאזין להרבה שידורים במקביל.
2. Get/Set – סוג של ארכיטקטורת תקשורת כאשר האפליקציות יכולות להישאל עבור פרמטרים מסוימים (לדוגמה: יכול להעיד על סטטוס של האפליקציה) וגם יכולים להשתנות מרחוק.
3. Shared Memory – כאשר הצמתים משתמשים ברוחב פס גדול מאוד (כמו פריימים של וידאו), הזיכרון המשותף מספק תור המאפשר גישה קלה עבור כל הצמתים על מחשב אחד.

השכבה העליונה (המסומנת במסגרת כחולה) מספקת שירותים שונים כמו:

1. Hardware abstraction
2. Host abstraction
3. Time services
4. Coordinate frame transformation
5. Record and replay
6. Diagnostics
7. Analytics

טכנולוגיות בעת יצור הפלטפורמה:

תוכנה: בין תוכנות שהשתמשו לייצור הפלטפורמה polySync היו eclipse ואובונטו.  
 שפות תכנות: c, c++, matlab\simulink ועוד.   
סביבת פיתוח:

ARM 64 bit on:

* NVIDIA DrivePX
  + Ubuntu 15.04
* NVIDIA Jetston TX1
  + Ubuntu 14.04

ARM 32 bit on:

* NVIDIA Jetston TK1
  + Ubuntu 14.04
* Renesas R-Car H2 Lager Board
  + Yocto Daisy 1.6
* Renesas R-Car H3 Lager Board
  + Yocto Linux 2.12
* Raspberry Pi 2/3
  + Ubuntu 14.04

x86\_64 on:

* Ubuntu 14.04.4

Virtual Machines

* VMware Player 12
  + Host: Ubuntu 14.04
  + Client: Ubuntu 14.04
* VirtualBox
  + Host: Ubuntu 14.04
  + Client: Ubuntu 14.04
* Parallels
  + Host: Mac OS X
  + Client: Ubuntu 14.04

כלים:

OSSC השתמשו בכמה רכיבים שונים לשליטה על הרכב, כל אחד מהם השתמש בטכנולוגיות שונות לצורך כך:

Steering

1. Arduino Uno
2. PolySync Steering/Throttle Shield R0

Throttle

1. Arduino Uno
2. PolySync Steering/Throttle Shield R0

Brake

1. 04-09 Prius Brake actuator with wiring pigtail
2. Brake Fluid Reservoir
3. Arduino Uno
4. PolySync Brake Shield R0

CAN-Gateway & Control Can

1. Arduino Uno
2. Seeed Studio CAN-BUS Shield, v1.2
3. ScanTool OBD Interface Cable
4. 22 AWG Stranded Twisted Pair Wire
5. Solder Cup DB-9 Connector

ספריות:

ספריות חיצוניות שמשתמשים בתוכנה:

SPI.h

PID\_v1.h

FiniteStateMachine.h

Stdint.h

avr/wdt.h

רכיבים:

* API
  1. Set of APIs to enable the development of level 4/5 autonomous vehicle applications
* Dynamic Driver
  1. Dynamically loads the .so interfaces written to communicate directly with supported sensors and actuators
  2. Interfaces are provided by PolySync for each supported sensor, and they can also be developed in-house
* Manager
  1. Entry point to the PolySync runtime
  2. Spawns nodes that are defined in the SDF Configurator
  3. Manages the runtime wall-clock and replay clock
* SDF Configurator
  1. Defines the PolySync runtime, all hosts, and nodes in the system
     1. What sensors are connected to the ECU, or
     2. What sensors will be replaying data
* Studio
  1. Plugin based visualization node
  2. Subscribes to all PolySync message types

API:

הממשק כתוב בשתי שפות, C++ ו C.

ממשק בשפת C

PolySync provides a robust set of C APIs to access and process data from the

PolySync bus. This data comes from connected sensors and hardware in the form of

PolySync messages. Messages are published and subscribed to the bus by nodes. The

C APIs allow you to create, publish, subscribe, and process sensor data, as well as

control the vehicle through the environment.

ממשק בשפת C++

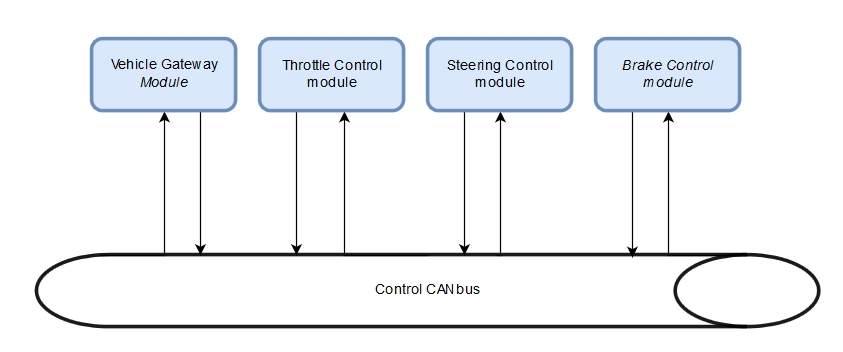
PolySync provides a set of convenient C++ APIs to develop autonomous vehicle

applications. The C++ API wraps the PolySync C API, and provides access to the

PolySync bus that contains data stored within PolySync messages. The message data

can be input from sensors around the vehicle or output to actuators for controlling

steering and speed.



**Vehicle Gateway Module:**מודול האחראי לקלוט ולשלוח מידע דרך ה Control CAN bus ומתממשק עם מערכות הרכב הקיימות

**Brake Control module :**מודול זה אחראי להעניק פונקציונליות מלאה למערכת הבלמים של הרכב - מתממשק דרך ה bus ומסוגל "לדמות" לחץ על דוושת הבלם שהיא תגרום לעצירה על הרכב

**Steering Control module:**מידול זה אחראי להעניק פונקציונליות למלאה על המערכת ההגאים של הרכב - ההגה - מתממשק דרך bus ,מסוגל להטות את הרכב לפי זווית שקיבל דרך ה Bus מרכיב האחראי לניהול הדרך

**Throttle Control module:**מודול זה אחראי להעניק פונקציונליות מלאה על מערכת ההאצה של הרכב - דוושת ההאצה , מסוגל לדמות לחץ על הדוושה ההאצה היא תגרום להאצת הרכב

* בכול לרגע נתון עוברים מסרים ב bus ת לכול מסר ID וכל רכיב יודע לזהות מסרים עבורו   
  מסר יכול להיות פקודה ( command ) או פלט (report) של הרכיב . פלטים נשלחים בעבור רכיב ניהול הדרך - מידע זה חיוני לניהול המסלול
* ניתן לראות פירוט מלא של המסרים ואת המבנה שלהם בקישור מטה   
  <https://github.com/PolySync/OSCC/wiki/CAN-Specification>