Vehicle-to-Vehicle (V2V) Communication Implementation

Gunnar Fandrich
Group: Autogators
Electrical and Computer Engineering
University of Florida
Gainesville, Florida
gunnarfandrich@ufl.edu

Mark Lai
Group: Autogators
Electrical and Computer Engineering
University of Florida
Gainesville, Florida
marklai@ufl.edu

Rafael Hernandez-Lopez
Group: Autogators
Electrical and Computer Engineering
University of Florida
Gainesville, Florida
rhernandezlopez1@ufl.edu

Rohan Malik
Group: Autogators
Electrical and Computer Engineering
University of Florida
Gainesville, Florida
rmalik@ufl.edu

***Abstract*—Vehicle-to-vehicle (V2V) allows communication between vehicles, promoting driver awareness and potentially reducing the number of collisions. Existing V2V implementations (cellular vehicle-to-everything, C-V2X) rely on cloud data, whereas the implementation shown in this paper does true V2V between vehicles. A mockup utilizing two ESP32 with a time-of-flight (ToF) sensor and accelerometer communicate using UDP to deliver a proof-of-concept for V2V that can be used as a base for V2V.**

***Index Terms*—V2V, vehicles, driver, awareness, C, ESP32, WiFi, UDP, UDP Broadcast**

I. Introduction

A. Problem Statement

As society moves towards utilizing more autonomous driving systems, vehicles can act as a network to promote efficient and safe driving. This network is referred to as vehicle-to-vehicle (V2V). Automotive vehicles should communicate with each other through V2V to increase driver awareness and reduce the number of collisions [1]. A simple mockup of a V2V system should illustrate the safety increase provided by V2V communication.

B. Background and Related Works

The current state of V2V is nonexistent. The closest thing on the market to V2V is the 2023 Safety Cloud for Chrysler Vehicles, which is implemented through a cellular network to create C-V2X (cellular vehicle-to-everything) [2], [3]. While this is fairly close to V2V, passing through a cloud does not exhibit true V2V. Ideally the cars would communicate directly to each other, which is what is covered by the implementation discussed in this paper. Additionally, Stellantis announced vehicle-to-grid (V2G) testing in 2019 [4].

In 1999, the FCC designated a 75 MHz spectrum in the 5.9 GHz band for dedicated short-range communications (DSRC) [5]. Unfortunately, this band was reallocated in 2020 for unlicensed WiFi [5] use due to the failure of automobile makers to release V2X production cars. Due to the removal of this band, we chose UDP over WiFi as an acceptable simplified protocol for V2V. Ideally, UDP would be replaced with another protocol built on top of UDP – or similar protocol – specifically for V2V and operate on an FCC-designated band for V2V.

V2V will be illustrated using an ESP32 paired with time-of-flight (ToF) sensor and accelerometer. Each ESP32 represents a simplified car, and communicate with each other using WiFi. UDP Broadcast was chosen as the communication protocol. UDP Broadcast sends a message to every vehicle listening within the network.

In recent years, newer vehicles have been produced with various driver warning systems. One such driver warning system is lane departure warning. Lane departure warning has been observed to reduce single-vehicle, side-swipe, and head-on crash severity by 11 percent and the severity of crash injury by 21 percent [6]. Lane departure warning systems solely alert a driver of a single nearby obstacle on each side of their car. V2V, on the other hand, provides the possibility to reduce common vehicular crashes by communicating warnings and the presence of more than two vehicles or obstacles to the drivers of connected vehicles.

II. V2V implementation

This section covers our V2V implementation, the features and the mockup.

A. Features

The two vehicles are represented by two ESP32s. Our implementation is scalable to allow for more vehicles. They each have a ToF sensor and an accelerometer. The ToF sensor is used to identify whether there is another "vehicle" in front

of the ESP32 and the accelerometer is used to know whether the other "vehicle" is moving. On a real vehicle, the CAN bus can be used to monitor and send each vehicle's velocity – and other information as needed, such as GPS for locating where the incoming UDP packet is coming from – in the UDP packet. The current UDP packet we are sending has an ID for the vehicle, the ToF distance, and the accelerometer measurement. On the other vehicle, this information is used to alert the driver via some LEDs. This UDP packet was standardized between vehicles using a defined C struct to contain the relevant information.

B. Hardware

The mockup can be found on Fig. 1. Each breadboard has an ESP32, a power supply, a ToF sensor, and two LEDs. Each could have an accelerometer, but only one "vehicle" had an accelerometer during testing.

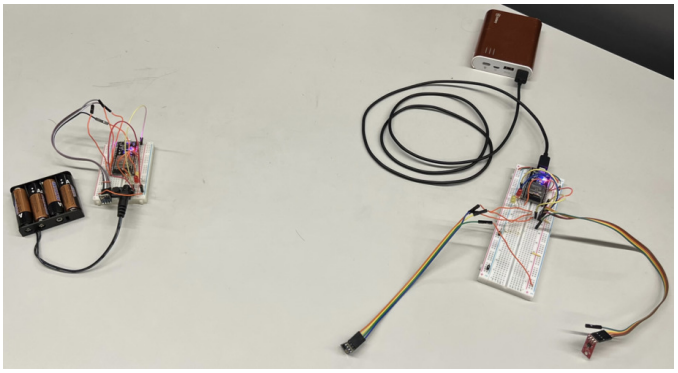


Fig. 1. The mockup used to test V2V, ESP32s are "vehicles".

Close-ups of each "vehicle" can be found in Fig. 2 and Fig. 3. The assembled "vehicles" can be found in Fig. 4 and Fig. 5. They are named "vehicle" B and U, respectively, from now on. B for battery pack and U for USB battery pack.

The driver has two LEDs available in this implementation: a yellow "caution" LED, and a red "warning" LED. The yellow LED turns on when there is a nearby vehicle (less than 100mm for this experiment) exchanging UDP packets with the driver's vehicle. The red LED turns on in cases where the driver should be more cautious. To simulate this, the red warning LED was turned on in the case that a caution message has been recieved, the ToF sensor has measured a distance less than or equal to 50mm, and the accelerometer observes a negative acceleration.

Two other lights, intended for debugging, were also incorporated into the design. The blue LED onboard the ESP32 was employed to indicate a successful boot and successful WiFi connection between the two ESP32s. The red LED oneboard the ESP32 was utilized to monitor the power supply and to ensure the ESP32 was powered on.

Throughout the project's development, several hardware issues were encountered. One of the issues was the original set of Time-of-Flight sensors not working. This delayed the project and prompted the need for new Time-of-Flight sensors. Fortunately, this issue was resolved quickly by ordering a new

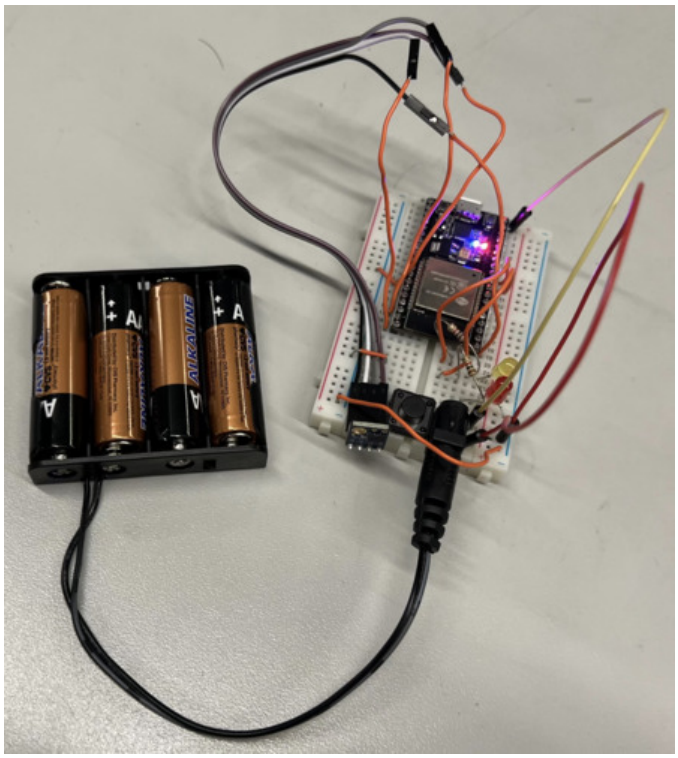


Fig. 2. One of the "vehicles", vehicle B, used to test our V2V implementation.

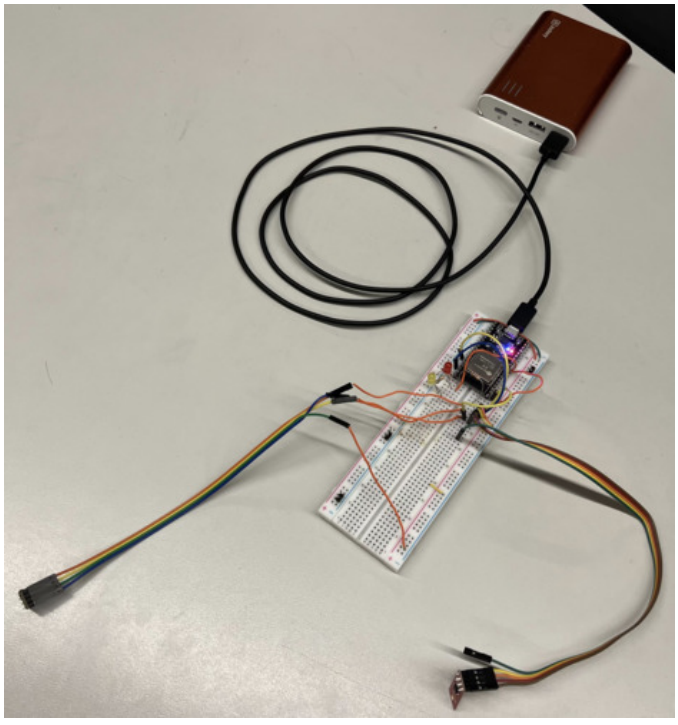


Fig. 3. Another "vehicles", vehicle U, used to test our V2V implementation.

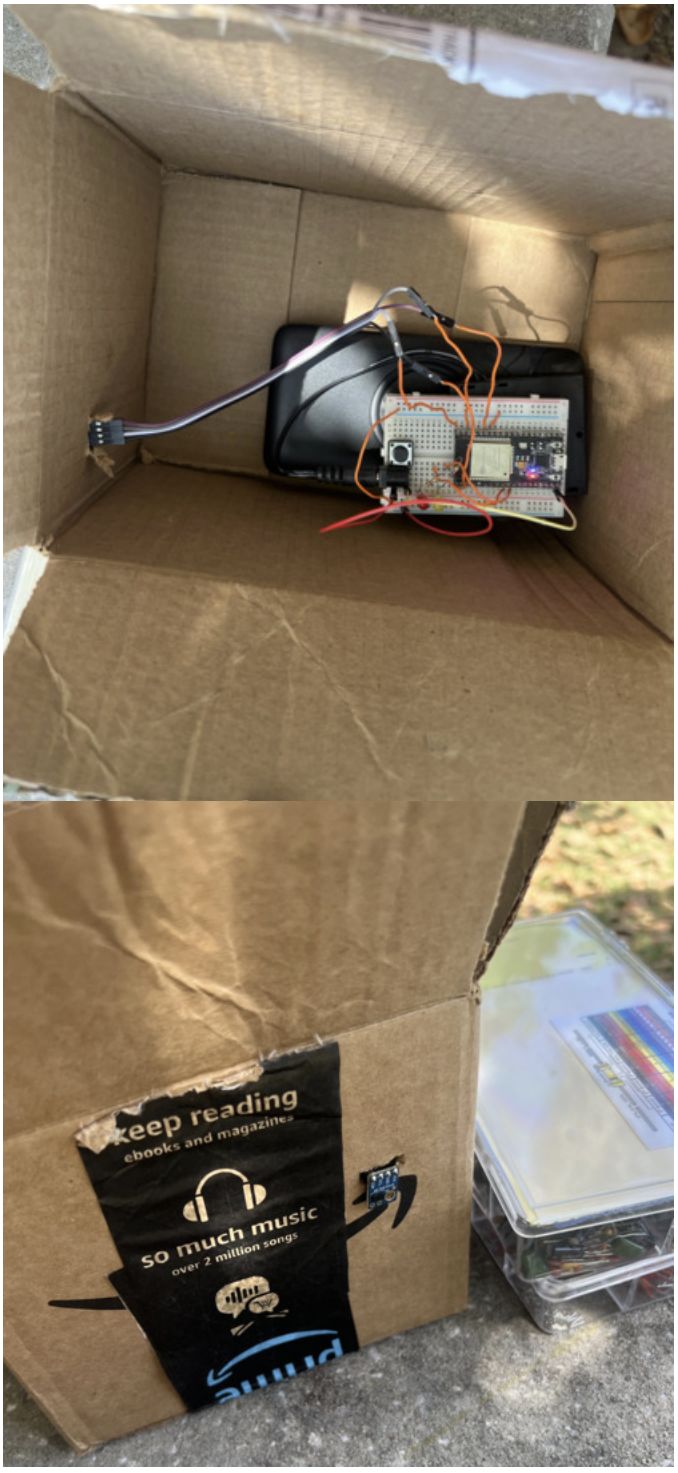


Fig. 4. Assembled "vehicle", vehicle B, in our testing environment.

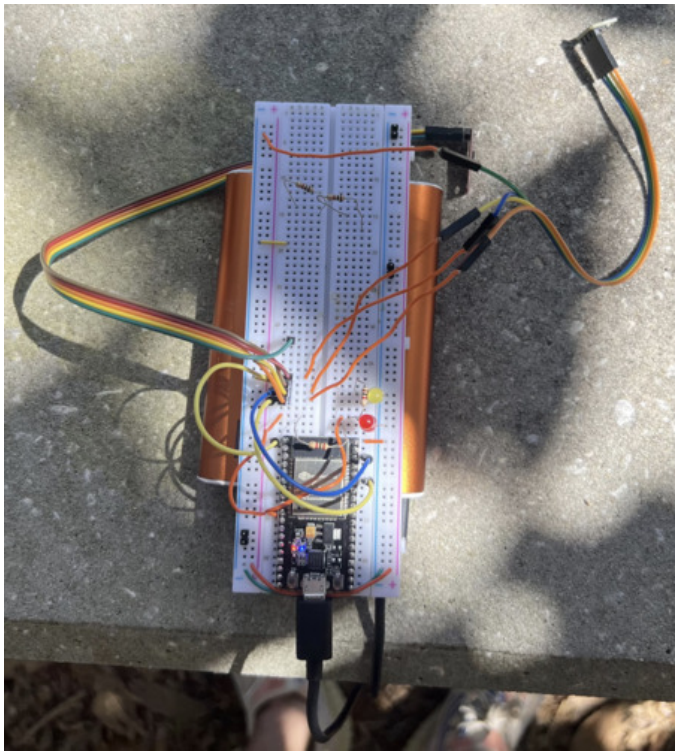


Fig. 5. Another assembled "vehicle", vehicle U, in our testing environment.

set. The new set did not work perfectly, but this was likely due to the quality of the components. There were intermittent disconnect issues or sporadic readings, but generally the sensors worked and fulfilled their role in this project. Another issue encountered during testing was power supply issues. Originally, two mobile phone battery banks were to be used to power each ESP32, but the minimal current draw of the ESP32 caused one of the battery banks to switch off. This battery had to be substituted for a battery consisting of four AA batteries in series. This battery provided 6V, and the internal power regulator of the ESP32 was able to step the voltage down to the system's 5V requirement.

III. Results

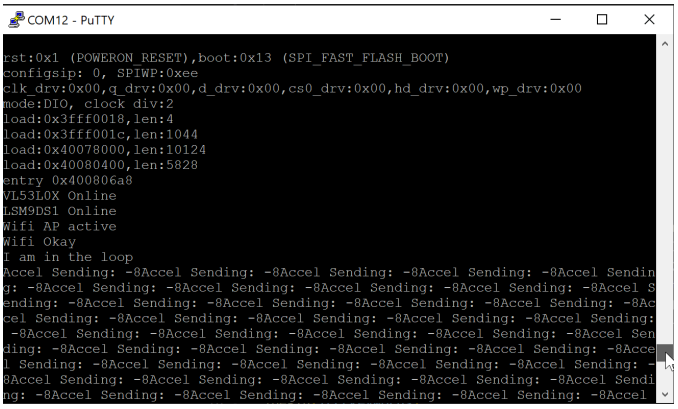
The two vehicles were able to be powered independently and in a portable nature. The platoon leader vehicle hosted a WiFi access point, which the subsequent vehicle connected to. ToF data, vehicle status, and accelerometer data was sent between each vehicle and proccessed accordingly.

The two "vehicles" communicate over UDP with each other. See Fig. 6 to see the "vehicle" sending a UDP packet, and Fig. 7 to see the other "vehicle" receive the UDP packet.

The code for this experiment can be found on Github: https://github.com/theriders/V2X_Example

IV. Security

Our implementation of V2V has room to improve. A V2V implementation usable on the market would address a few security concerns.


COM12 - PuTTY
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5828
entry 0x400806a8
VL53L0X Online
LSM9DS1 Online
Wifi AP active
Wifi Okay
I am in the loop
Accel Sending: -8Accel Sending: -8Accel Sending: -8