A 4G-Connected Micro-Rover With Infinite Range

Peter J. Burke, *Senior Member, IEEE*

***Abstract*—The development of micro-rover technology could enable multiple applications. However, the range of control so far has been limited to the radio system used. If cellular (4G/5G) connectivity could be used, its control range would be practically infinite. This work describes the design of such a system. The micro-rover is controlled by an onboard GPS linked STM32 32-bit ARM-based microcontroller, which communicates with an onboard Linux computer. The control is over a 4G/LTE network, which can provide full manual (throttle and steering) or full autonomous (GPS waypoint controlled) navigation. A live HD video stream is used for driver awareness and navigation. With this approach, in this work, we demonstrate a 4G-connected micro-rover with semi-autonomous driving capability and a prototype command and control architecture with encryption that can punch through firewalls at the driver and rover end. At 0.3 m in length and 900-g weight, this is the smallest 4G-connected autonomous micro-rover ever demonstrated. Because of the 4G connectivity, its control range is practically infinite.**

***Index Terms*—4G, autonomous vehicle, self-driving car.**

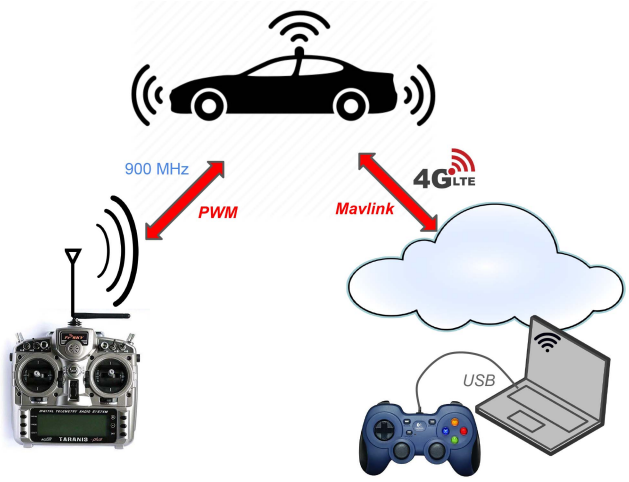

Fig. 1. Redundant command and control mechanisms. The local (RC) control has limited range whereas the 4G/LTE command and control can be from any Internet-connected point on the Earth.

I. Introduction

AUTONOMOUS and semi-autonomous remotely piloted vehicle technology is developing at a rapid pace for ground-based vehicles (self-driving cars, trucks, and rovers), aerial vehicles (drones, UAS, etc.), and even water-based vehicles. Several software tools based on open source are available for local control, but remote control is usually done via a custom RF interface designed specifically for the control link. On the other hand, Internet-based (including cellular) control is more versatile and with the prevalence of cellular data plans, offers a more economical, flexible, versatile, and broadly based communications technology with the potential to control fleets of cars, trucks, drones, boats, etc.

To date, although there have been full-size (human driver backup) demonstrations of "teleoperation" (i.e., remote driving) [1]–[4], no miniature fully autonomous 4G-connected rover has been demonstrated at this scale. In this work, we demonstrate the smallest (under 0.3 m long, and weight under 900 g) autonomous 4G-connected rover ever made. The applications are different than human transport and could include multiple scenarios where small size, low power, and low-cost rovers could be deployed at scale. The power for the electronics is only 10 W or less, much lower than the power used for full-size self-driving car navigation and remote control systems. We measure the glass-to-glass (GtoG) video latency as well as packet [round trip time (RTT)] latency in this setup, which has never been done before for such a small compute platform. We show the video latency in this miniature system, although still higher than desired for full manual control, is comparable to other demonstrated systems, and suggest avenues for improvement. Finally, we demonstrate a novel, self-healing network architecture that allows full encryption with simultaneous ability to tunnel through firewalls for the ultimate in flexibility and mobility, needed in a dynamically changing and unpredictable 4G network environment "in the field." Because of the 4G connectivity, the range of control is essentially infinite, limited only by the size of the Earth, a first for miniature rover technology.

Manuscript received July 11, 2020; revised August 17, 2020; accepted August 19, 2020. Date of publication August 21, 2020; date of current version December 10, 2020.

The author is with the Department of Electrical Engineering and Computer Science, University of California at Irvine, Irvine, CA 92697 USA (e-mail: pburke@uci.edu).

Digital Object Identifier 10.1109/JMASS.2020.3018660

II. Hardware/Software Stack

The hardware/software stack enables 4G/LTE command and control as shown in Fig. 1. The protocol used for the control is Mavlink [5]. The architecture is described next.

Fig. 2. Ground vehicle used for this demonstration and the laptop computer used for the remote pilot interface running Mission Planner software.

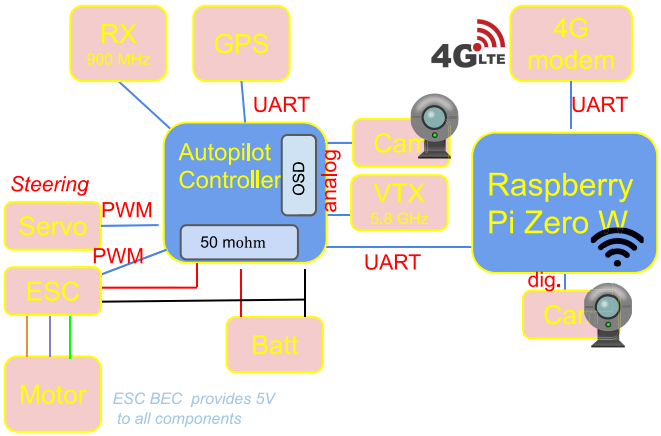

Fig. 3. Schematic of control electronics.

A. Hardware

1) Vehicle Platform: Although any ground vehicle of any size would do, for the purposes of ease and safety, in order to demonstrate the platform, a small vehicle was chosen. The vehicle is a 1/16 scale model with a 180-mm brushed motor with a servo-controlled 10-A capacity electronic speed controller and servo-controlled steering, of overall length 27 cm (Fig. 2). The truck form factor was chosen to allow for the 4G modem, control electronics, and GPS to fit in the back with easy access. The speed at full throttle was about 2 mph.

2) Control Electronics: In order to control the servos, an STM32-based microcontroller board designed for autonomous vehicle control (Omnibus F4 Pro) is used with onboard Ardupilot Rover firmware [6]. The microcontroller provides servo outputs to the steering and throttle and performs all the autonomous functions including GPS-guided waypoint missions. A 900-MHz RF receiver (TBS Crossfire Nano [7]) from a radio control is used when local manual control is desired. An onboard Raspberry Pi Zero W is interfaced to the controller through a UART port. The Pi is interfaced to the 4G/LTE network through a USB modem (Novatel USB730L). The battery is a 2S 18650 Li-Ion pack with 3200-mAh capacity. The overall electronic schematic is shown in Fig. 3, and the system network architecture in Fig. 4. Hardware photographs of the various control electronics are shown in Figs. 5–7.

B. Software

The software for the onboard firmware is Ardupilot Rover v. 3.5.2. The Raspberry Pi runs an open-source packet router called Mavlink Router [8], developed and maintained by Intel. Mavlink Router passes Mavlink packets to specific IP addresses. A cloud-based architecture, discussed below, guides the command and control packets to/from the vehicle to/from the remote driver. The remote driver runs an instance of Mission Planner software [9] on a local Windows 10 PC, with a USB joystick/steering wheel attached. Otherwise, the Mission Planner software can allow the driver to define waypoints and have the vehicle proceed to those waypoints autonomously. The local PC also uses a Web browser to access the live video stream.

III. Network Architecture

The advantage of using a 4G/LTE network for command and control, as opposed to a custom network created just for this use case, is as follows. First and foremost, the network infrastructure is already in place, and can be used in a very economical manner. Second, the network infrastructure allows the arbitrary distance between driver and vehicle. Finally, the network infrastructure is scalable to millions of users. Thus, this approach is advantageous compared to building and maintaining an architecture specifically for remote piloting of vehicles.

In order to leverage the 4G/LTE infrastructure, a networking architecture must be designed, developed, and demonstrated prior to mass deployment. This architecture must be able to penetrate through firewalls on potentially both ends of the connections, penetrate through NAT, be self-healing in case of lost connection, and be able to dynamically respond to changes in network address conditions as the vehicle is handed off from base station to base station. Ideally, the connection should be verifiable and encrypted to avoid software attackers from taking control of the vehicle for safety reasons.

In this work, we develop, demonstrate, and test two such architectures, one based on a cloud Linux system with fixed IP, and another based on a dynamic peer-to-peer network. Both are indicated in Fig. 4.

A. Fixed IP Cloud-Based Linux Architecture

The general idea here is to use a fixed IP address Linux instance in the cloud to coordinate the Mavlink packets from vehicle to driver. The cloud service provide used is AWS, but any Linux cloud machine will suffice. Reverse SSH is used to encrypt the traffic and penetrate firewalls. Auto-SSH is used with a redial time of 5 s to reconnect in case of lost connection. The only requirements for the cloud Linux machine are: 1) fixed IP address, so both the vehicle and driver PC know "who to call"; 2) auto SSH installed; and 3) running Mavlink Router. All of these software requirements are open source and readily available. This architecture was also developed and demonstrated by us for a flying wing UAV [10], and the setup scripts are available as open source under GPL3 license at www.gitlab.com/pjbca/4GUAV. However, they were not tested for latency quantitatively as we demonstrate below. The reason the video latency was not tested quantitatively in that work is that it is too dangerous to attempt manual flight. Therefore,

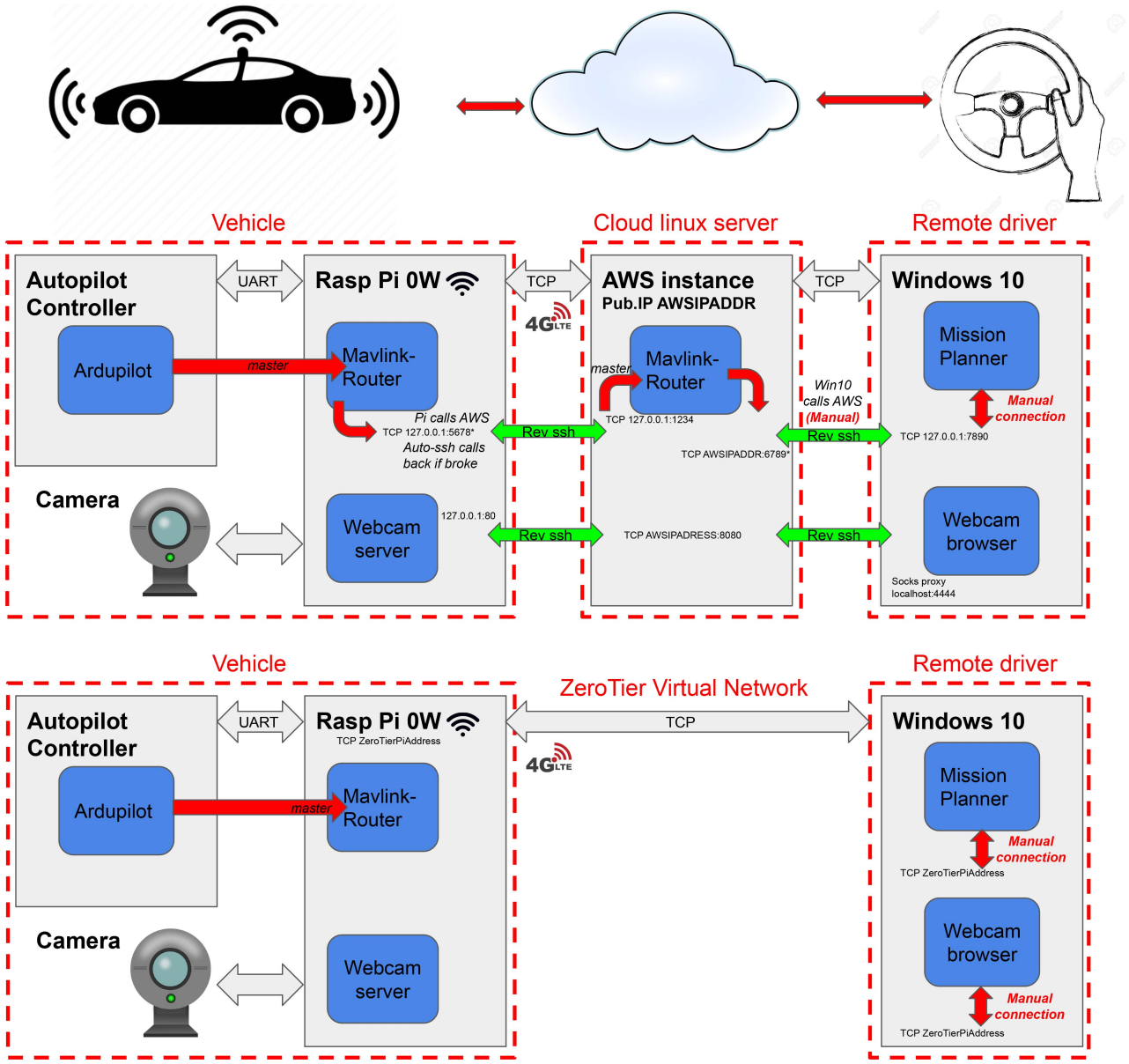

Fig. 4. Network architectures developed and demonstrated in this work.

this ground vehicle platform has a significant advantage as a testbed for 4G/LTE command and control of autonomous vehicles of all kinds, be they land, air, or sea.

B. Peer-to-Peer Based

Rather than route all traffic through a centralized server, an alternative architecture would set up an encrypted tunnel directly between the vehicle and local driver PC. Such a VPN solution is in general possible but must be easy to configure and be able to dynamically reconnect in case of changing network situations. It turns out that such a technological solution exists and is offered as a freemium service by ZeroTier [11]. The technology installs a client app on the local or onboard computer and creates an encrypted peer-to-peer virtual network in a way that is transparent to the user. The user gets an IP address and can directly contact any other device once it has joined the virtual network, using the other device's IP address. The handshakes and setup are handled by cloud-based servers, but once the peer-to-peer network is established, the centralized server is no longer used.

The use of ZeroTier has been demonstrated in a proprietary UAV case [12], and also an open-source platform for one specific flying wing which is no longer manufactured [13]. However, an end-to-end test of the latency as well as an open-source, general purpose demonstrated has not until now been demonstrated for ZeroTier for any vehicle (land, air, or sea).

C. Local Command and Control (Backup)

A separate local radio control link can also be used, but was only used for diagnostics. All of the driving missions in this article were performed with the local radio controller turned off, so all control was over the 4G/LTE connection.

D. ZeroTier Versus Custom Solution

The advantage of using ZeroTier is that the connection is automatic and transparent to the user, and requires little user input once properly configured. It is also much easier to configure in the beginning, and does not require a third-party server to be maintained by the user. A further advantage is that,

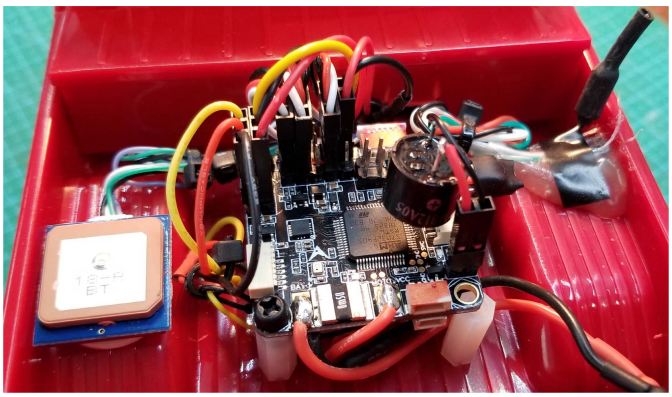

Fig. 5. On controller board (32-bit ARM processor visible) which runs the autopilot software. GPS is shown on the left.

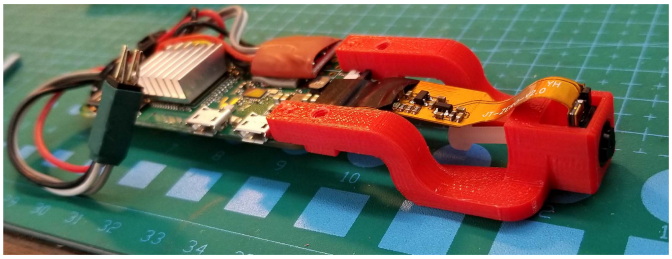

Fig. 6. The 1-g camera inside 3-D printed mount, connected to the Raspberry Pi Zero W Linux single-board computer.

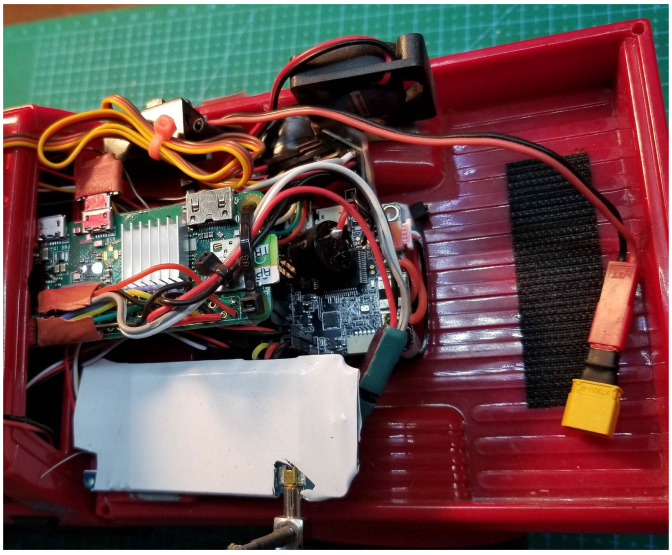

Fig. 7. Completed control electronics showing the Linux single-board computer, the autopilot control board underneath it, and the 4G modem (white) on the bottom of the image.

if there is a faster, more efficient route between vehicle and driver station, ZeroTier software will automatically reconfigure it as a peer-to-peer network connection, which can in some cases be much faster than going through a third-party server. (For example, if the vehicle and the remote driver computer are both on the same WiFi network, ZeroTier will dynamically adjust the path so that the traffic will not go beyond that local network, whereas in the other case, the traffic would go all the way up to the cloud and back.)

Fig. 8. Purple line represents path driven through a residential neighborhood over the 4G/LTE network. The remote driver was inside of a building for the entire test drive.

The disadvantage is that, if the connection is dropped or broken, the diagnostic feedback to the user is very minimal and the user is left to just wait for the system to reconnect, which takes up to a minute. Although the ZeroTier software is optimized for different use cases, in principle this could be adjusted in the future (ZeroTier Corporation, private communication). In contrast, with a custom solution, the user can manually or set up for auto reconnection and monitor all of the network connections using Linux command line open-source tools, which are much more mature.

Note that we did not find any additional appreciable CPU overhead for the ZeroTier client software on the Raspberry Pi Zero W, indicating that is a nonissue. (This was not clear in our prior publication UAV [10].)

IV. Demonstration of Performance

We discuss the performance of this system next in driving and network speed.

A. Driving

The Ardupilot software is very well developed with a variety of modes from complete manual control to complete autonomous control. Both are demonstrated in this work.

1) Manual: In manual control, the control inputs by the remote driver directly translate into the movement of the steering servo or throttle setting. In this mode, although there is significant latency, it is possible to remotely drive the vehicle using the streaming video. A test drive of 1.5 h on the sidewalks of a local neighborhood was performed to demonstrate the control over 4G/LTE. The driver was inside a building and had no visual contact with the vehicle during the entire test drive. The vehicle disconnected three times during the drive but quickly (within under 10 s) reconnected. A "spotter" walked along with the vehicle as a safety measure. Manually, piloting on a sidewalk was possible with some practice. The path driven is shown in Fig. 8.

2) Autonomous: In autonomous mode, the vehicle relies on the GPS for location information. Since the GPS precision is roughly of order 1 m, autonomous missions on the sidewalk

Fig. 9. Representative streaming video quality for various lighting conditions.

were not precise enough. However, autonomous waypoint missions on a street were possible (done on side streets to avoid traffic). In "guided" and "waypoint" modes, the rover drives to (navigates autonomously) predefined waypoints. The number of waypoints is limited only by memory but we have demonstrated proof of concept for few (10) waypoint missions only. The autonomous algorithm and performance expected are well documented in the Ardupilot software documentation, and so the reader is referred there for further description of the nuances of all the autonomous modes.

B. Video

The Raspberry Pi acts as a webcam streaming video, and also records to a local SD card. Although there is significant latency (see below), the quality was 1080 and very high. Fig. 9 shows example video in various lighting conditions.

C. Latency

One of the most significant challenges of long-range control is the latency of the network. We have characterized the "GtoG" and "RTT" latency in this system. In contrast, prior literature using Intel i7-based PCs with 100-W power consumption and much larger footprint (below), this work is the first characterization of these in a microcontroller-based system with a miniature single-board ARM-based Linux computer with low dc power consumption (below 10 W).

1) RTT Latency: RTT is the round trip time, defined as the time to send and receive a SYN-SYNACK message in a TCP architecture (SYN to SYNACK packet delay). In our case, this was under 50 ms for the 4G connection, and under 40 ms for the WiFi connection. We measured this with the PING utility in Linux, the common technique (see the comparison to other work section below). This latency is comparable to other studies (see the comparison to other work section below).

A speedtest.net on the Linux test showed a bandwidth of 20–50 mb/s, which is more sufficient for the video applications presented below.

2) Glass-to-Glass Latency Definition: While the RTT time is the network latency, the user experience is the GtoG video latency.

This includes all of the intermediate steps (handshakes, handoffs, hardware layer, software layer, encryption/decryption, video encoding/decoding, etc.)

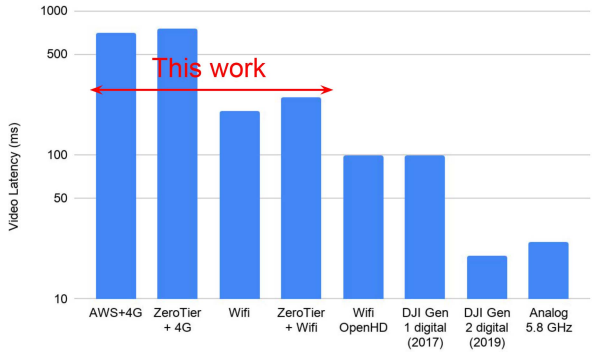

Fig. 10. Video latency for different configurations of network for this work (left four bars), and comparables from state of the art (right bars).

We first discuss the methodology, then the results, then a comparison to known SOA.

3) Glass-to-Glass Video Latency Methodology: The latency of the video was determined as follows. The rover camera was shown an image of a video with a timestamp (1-s progress bar, repeating) on the video. The rover camera then fed the video through either WiFi or 4G network to a client, which was a desktop PC. The desktop PC monitor was the same monitor used to show the video image with the progress bar. By comparing the two images (the original, and the "broadcasted" image), the GtoG video latency could be determined.

This method was invented by us independently, but after we implemented it we found that several research groups have used substantially similar methods or variations on this method (using the QR code or clock images in some cases) to determine GtoG video latency. The reader is referred to [14]–[18]. Therefore, our method is well established in the literature.

4) Glass-to-Glass Video Latency Results: The results for the video latency test are compared in Fig. 10. The 4G latency is of order 700 ms, which is a significant challenge for manual control. For autonomous waypoint missions, it is acceptable. A separate test used the local WiFi network for connectivity as a comparison. In this case, the AWS server is not used and a local network is set up, so that the Windows 10 PC and vehicle Raspberry Pi WiFi connection are both on the same subnet and do not pass traffic beyond the router. The video latency, in that case, is about 200 ms.

When the ZeroTier is used over the 4G or the wireless, there seems to be an additional 50 ms. We suspect this is because of the heavy load on the Raspberry Pi Zero W, which is already quite loaded with the video processing for the webserver. This likely is not there with a better processor.

Finally, when the vehicle was on both the WiFi and 4G simultaneously, the ZeroTier software was able to dynamically route the traffic so that the (faster) WiFi was used, when available, demonstrating that the ZeroTier approach does succeed at finding the fastest route between vehicle and remove driver PC.

5) Latency Compared to SOA: The origin of the extra 500-ms latency of 4G versus local WiFi is not clear. The speedtest of 30 mb/s and ping times of 50 ms or less show that the latency is not directly calculable from these singleton tests.

The 200 ms of WiFi is better studied. An open-source project to minimize latency in WiFi video transmission has shown that the WiFi protocol can be adjusted in firmware to reduce the number of packet handshakes and get the video

latency down to 100 ms [19]. We have demonstrated this in a separate experiment also [20]. The UAV community is very interested in latency, and as of last year the lowest latency HD digital video link was 100 ms [21]. Recently, using multiple antennas, MIMO, multiple 5.8 GHz channels, and relatively high TX power, DJI demonstrated a propriety system with 20-ms video latency [22]. Finally, for analog 5.8-GHz video, 20-ms latency is achievable and available commercially from a variety of sources [23]. This is possible because the overhead of DSP is gone in an analog system. However, the quality and resolution are lower than digital. These SOA results are indicated in Fig. 10 also.

6) Possible Ways to Improve Latency: The first step to improve the latency is to digest exactly which stage is causing the latency, i.e., encoding/decoding, encryption/decryption, packet loss, camera intrinsic speed, processor power (delay versus MIPS), TCP versus UDP, and h264 versus MJPEG encoding. Such a gargantuan task is beyond the scope of this article, but some progress in this direction for higher performance Intel i86 architecture systems has been made in the literature, discussed below. What is the ultimate performance limit of a 4G-connected video system and what is the MIPS and dc power requirements for that? We know one end (low MIPS low dc power) from this study. Prior studies (see below) give another data point. Eventually, in the future, a general, predictive description of video latency as a function of system MIPS and dc power should be developed in order to improve latency. Just like the Shannon limit gives information capacity in the presence of noise, there should be a similar "Burke" limit that gives video GtoG latency in 4G systems in the presence of finite compute power (MIPS). That study remains an important topic for future work.

V. Discussion

A. Drones

We recently developed a similar platform for a flying wing in UAV [10]. However, in this work, we have extended it to: 1) test the ZeroTier tunnel solution; 2) test the video latency; and 3) test full manual control. Due to the latency, full manual piloting of the flying wing was deemed too dangerous and not even attempted. This clearly shows how this ground vehicle, being much safer, can serve as an excellent testbed for remote control of all kinds of vehicles, be they land, air, or sea. We anticipate this will be useful for many future drone-based technologies as a safe, simple, and "rapid prototype" platform for future Internet-based command and control software, hardware, and cloud-based architectures.

B. UTM Test

One of the challenges of drones is air traffic management. An integrated network solution is likely going to be a significant component of this [24]. However, testing of the networking and hardware/software architecture can be done under much safer test conditions with ground-based vehicles as a testbed. This allows for more rapid prototype, failure modes to be more quickly identified and tested, and new innovations to be more quickly fielded and tested. Thus, this work also lays the foundation for a safe testbed of unmanned traffic management (UTM) for drones.

C. Vehicle-Independent Testbed

Rovers, drones, and sea vessels have one thing in common: they all move. The control electronics to control the movement and the software has many similar characteristics: PID control, GPS navigation, and motor and servo control. It turns out the Ardupilot software and flight controller hardware (such as the STM32 ARM-based microcontroller used here) can be used for all three modes of vehicles. The micro-rover platform, however, is the easiest and safest to operate. It is easiest because it cannot sink like a sea vehicle, and cannot fly away or crash if it stops moving like a drone or flying wing. The micro-rover platform presented here is much safer than a full-sized "self-driving car." The collision of a micro-rover with a human or other vehicle in general causes no damage to person or property.

VI. Comparison to Previous Studies

A. Teleoperation of Cars

In the state of California, it is a law that driverless cars must have a remote control backup. Several companies have given press releases [1]–[4], about teleoperation technologies, some over 4G, but all with full-size cars (not micro-rovers as we discuss here).

B. 4G-Connected Drones

The prior art on this was reviewed in our recent paper [10] and we refer the reader there for comparison of drone work to our network architecture. For example, we compared our approach to that of the software company UAVMatrix, which provides a proprietary solution based on ZeroTier for drones (but not yet rovers as we present here).

C. Latency: Glass-to-Glass Versus RTT Time

RTT latency and GtoG latency are two different quantities which are frequently confused in the literature. A typical claim is that 4G latency is below 100 ms, but that is the RTT or packet (for a single frame) latency, not the GtoG latency which a user experiences and therefore cares about.

Kaknjo *et al.* [25] considered latency in remote driving of a marine rover (ROV). They defined the GtoG latency as we did, i.e., "the time that passes from the moment when an event occurs in front of a camera lens to the moment when that event is displayed on a monitor. The monitor can be connected directly to the camera (monitor is in close proximity to the camera and connected with a single cable to it), or the monitor can be located anywhere in the world (and connected to the camera over the Internet, not in close proximity)."

Kaknjo *et al.* [25] found latency for packet transferal to be 40 ms over Internet, and under 10 ms with local area network (LAN). However, they found GtoG latencies of 500–1000 ms for LAN and Internet, depending on whether MJPEG/H264, UDP/TCP were used. This was with Intel i7 processor-based PCs. They also found webcam delay alone of 100–250 ms. So the packet traversal time/latency for a single UDP packet in a video streaming application is a very different metric than the GtoG time which is what is important from a user point of view.

While many studies have defined the latency in terms of packet round trip propagation time, that is NOT the latency

that matters to the end user, and only ONE of the components. Therefore, studies which only focus on that latency are misleading as they do not analyze the entire system chain.

Uitto *et al.* [26] evaluated 5G in the context of remote control using a 5G testbed and find network latencies of order 1 ms. Note again, this is not the GtoG latency. For example, Neumeier and Facchi [27] found an LTE GtoG latency of 200 ms, but only 50 ms of that was due to RTT, the remaining due to "system latency" and packet loss. If that RTT was reduced to 1 ms in 5G, the GtoG latency would still be 175 ms in a 5G deployment. It is very misleading for 5G papers to quote a 1-ms delay as if that is the end user experience. It is not.

In order to further study this, [28] has a comprehensive analysis of the delay of GtoG and even calls it GtoG. Network delay time (RTT) is only 1 of 13 different delay times in the total GtoG delay.

Therefore, claims of sub-100-ms latency over 4G are more like ping times and not GtoG latency. Our GtoG latencies are consistent with those of Kaknjo *et al.* [25], which used various ISPs and found GtoG latencies of up to 1 s.

D. Significance of Latency

The significance of the latency is linked to the mission of the device. For example, in some cases, the vehicle is under autopilot control heading to specific waypoints with no active control from the remote pilot. In this case, the latency is not a factor. In the other extreme, the vehicle is under fully manual control of the remote pilot, and may be confronted with a random, dynamic environment such as a falling obstacle or other vehicles. In this case, latency is critical for the operator to be able to adjust quickly to unpredictable challenges. Another example where latency is significant in navigating narrow paths. For example, we found it possible to navigate manually and remain on a sidewalk at low speeds but it required intense concentration due to the latency. In summary, the latency is important for manual operations but for autopilot missions, without dynamic obstacles, it is not important.

E. Detailed Breakdown of Prior Studies

Liu *et al.* [29] used a 1/10 scale RC car with simulated LTE delays of up to 350 ms. (They also measured RTT in LTE using packet delay of video frames UDP. They found under 100 ms typically.) They studied user workload and human performance on a test track and found that the variability in the delay was the most significant. It was a simulated LTE network, not a real one. Note the RTT was the packet delay for a single packet (frame), NOT the GtoG latency. In related work, [30] (also a major source, cited by 1058 papers) uses SYN to SYNACK packet delay as a measure of RTT, and find under 100 ms for small packets on LTE.

Juang [31] demonstrated a golf cart with teleoperation but did not have many details in the short conference paper based mostly on compression algorithms of nonvideo data back and forth between remote pilot and vehicle.

Kang *et al.* [32] considered the hypothetical case of human RC backup drivers for self-driving car applications in case of system failure. They evaluated the latency of LTE and said it can be 100 ms. They call this "frame latency" which is basically latency for transmitting a single frame of video, NOT the GtoG video latency. They used a custom android app to measure LTE and WiFi latency (two-way frame latency), frame size, and loss rate at various resolutions using h264. They found 50–100 ms for their h264 settings, taken from Skype and Google Hangout.

Huang *et al.* [33] developed a passive measurement tool to study the inefficiency in today's LTE networks. Reference [33] is a definitive authoritative study of LTE performance to date (cited 370 times to date), and defines network latency as RTT at the single TCP packet level: the time between SYN, SYNACK, and ACK. This is VERY different than the GtoG definition we use, since it is at a very low level (single TCP packet level). Even though the RTT can be below 100 ms for very small packet sizes (rising above 500 ms for larger packet sizes), this is not the same as GtoG.

Kang *et al.* [32] mentioned "In today's LTE networks, it is usually possible to accomplish a two-way communication latency of around 100 ms when streaming frames of various resolutions by using standard video compression algorithms [13]" However, [34] does not discuss latency at all!

Saeed *et al.* [35] considered the 5G and simulated available data rates for a hypothetical 5G system used for remote driving in a city, but latency is not discussed.

Neumeier and Facchi [27] studied the idea of adjusting the speed and route of remote ("teleoperated") cars to minimize delay and latency in video feeds. They find an RTT of 55 ms on LTE. They also find a total (GtoG) latency of 200 ms (55-ms RTT, 20-ms packet loss, and 125-ms system, without a detailed justification about "system latency"), thus demonstrating the RTT is only one component of the total GtoG latency in a video streaming teleoperation for remote driving applications.

Shen *et al.* [36] used an i7 PC on each end, NTP sync, and a QR code embedded in each for GtoG video latency measurement. They find WiFi video (GtoG) latency was found to be 105 ms, and 4G 180 ms. But the QR code is inserted after the image has been transferred to PC memory from camera so it is not a complete measure of GtoG latency.

In [37], they use ping to find RTT in LTE using PING, as we did in this article. They found under 100 ms for 4G/LTE, very similar to our results.

F. Prior Literature on Methods to Measure Latency

How is latency measured in the academic literature and in industry? Liu *et al.* [29] referred to a website (FierceWireless) which link is not up anymore saying average latency is "with the mean RTT ranging from 75 to 83 ms [15]." Other latency measures on the same website (FierceWireless) are listed, and the definition is given in website [38]. "For the latency test, OpenSignal runs three ICMP pings on google.com and takes the average." Therefore, it seems that even scholarly literature relies on ping as a test and definition of latency in LTE systems. As we mentioned above, we get pings from our rover to google.com of 50 ms, but that is VERY different than the GtoG latency. In sum, to say a ping test is the same as GtoG latency a user would experience is completely ignoring many of the most important delay factors. In our opinion, a ping should never be allowed in the academic literature on video latency.

Regarding the methods to measure GtoG latency: As Kaknjo pointed out: Kryczka *et al.* [14] compared the following techniques for transmitting timestamps in video frames: 1) EAN-8 barcode; 2) Arabic numerals; and 3) a progress bar. Kaknjo used a barcode and NTP sync PC to PC. This manuscript uses a progress bar. References [15], [16] take a similar approach. Our approach is different because both PCs do not need to be synchronized to the absolute time. Reference [39] is like ours: it sends a signal, then reads back the sent signal. They use a barcode. We use a progress bar. References [17] and [18] use running clocks as we do. Therefore, our GtoG latency technique is well established in the academic literature.

G. Human Factors

There have been some studies on the impact of latency on human, manual driving [32]. The effects of time lag on human performance were also investigated in [40]. In this work, we are not yet at low enough latency to envision manual control for high speeds with precision. However, we have demonstrated low-speed (1 mph) manual control keeping the vehicle on the sidewalk for a distance of over a km. The author was the subject and the author learned to compensate for the latency and did not have a single crash for the entire hour.

H. What Is New in This Work

There are multiple main differences between this work and previous studies. First, this work is the smallest 4G-connected autonomous rover ever constructed. Second, it is the lowest power budget ever shown, at under 10 W, using the lowest power ARM architecture ever shown. Third, this study shows several advances in networking architecture above and beyond prior studies, including an end-to-end encryption, a self-healing architecture in case of temporary lost 4G signal, ability to punch through firewalls at both ends of the link (important so the driver can be in any Internet-connected environment), and also evaluated a commercial peer-to-peer self-healing VPN solution. Third, the GtoG video latency and packet latency (RTT) are evaluated quantitatively using methods that have been vetted in the academic literature by multiple groups in the past ten years. The impact of encryption, low compute power, low electronic power budget, on video GtoG performance has never been evaluated in any remote driving system in the literature before, let alone for the micro-rover system we show here.

VII. Conclusion

We are using a very common, cheap 5, light (10 g) Linux board (Pi Zero W) with 1 g HD camera that uses about only 1 W of power. And we are using an open-source webcam program. The miniaturization is a unique aspect of this. The 4G modem is about 10 g. So the entire control, computing, and communications system is weighing in at under 40 g and lives in a few cubic cm and therefore is a very unique, micro-system that can be used for all sorts of micro-vehicles (air, land, and sea). The application space is very different than Tesla or Google self-driving cars which cost tens to hundreds of thousands of dollars, weigh literally tons, and have kilowatts of available power for computing and communication.

Thus, this article demonstrates the ultimate in miniaturization of autonomous, GPS-guided, cellular-connected, low-cost rover technology, with power budget of less than 10 W and size of order 0.1 m. Whereas all prior work in this field has focused on human-scale self-driving cars with kilowatt computer power budgets, this work opens the door to a myriad of applications where small size, low cost, and low-power budget are needed, such as chemical and biological weapons detection, remote sensing, environmentally friendly surveys, disaster relief efforts, low resource environments, such as off-grid remote locations, hostage rescue and reconnaissance, and medicine delivery in times of natural disaster. Finally, the micro-rover approach is safe and can be a powerful educational tool for university students and hobbyists interested in getting their feet wet in cellular-connected autonomous vehicles.

Note Added in Proof: An updated method of video streaming (which uses H264 instead of MJPEG encoding) with lower latency over 4G (under 250 ms) was added to the gitlab repo https://gitlab.com/pjbca/4guav after this paper was accepted for publication.

Appendix A
Table of Abbreviations

We present a table of abbreviations in Appendix A.

Abbreviation	Meaning
GPS	Global Positioning System
HD	High definition
UAS	Unmanned Aerial System
GtoG	Glass to glass
TBS	Team Black Sheep
UART	Universal asynchronous receiver-transmitter
USB	Universal Serial Bus
SSH	Secure Shell
AWS	Amazon Web Services
RX	Receive
TX	Transmit
MIPS	Millions of instructions per second
UTM	Unmanned traffic management
LAN	Local area network
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
NTP	Network Time Protocol
VPN	Virtual private network
ARM	Advanced RISC Machine
STM	STMicroelectronics
RTT	Round trip time

Appendix B
Multimedia

A multimedia demonstration of this work is available at https://www.youtube.com/watch?v=fY8QOdgm1-s.

Appendix C
Open-Source Code

All of the code for this project is open source. The autopilot firmware is available at www.github.com/ardupilot.

The Mavlink packet routing software is available at www.github.com/mavlink-router. The webcam software for the Raspberry Pi is available at https://github.com/silvanmelchior/RPi Cam Web Interface. The code developed by the author to coordinate all the SSH tunnels and link all the systems together (controller to Raspberry Pi to 4G modem to cloud Linux server to local Windows 10 PC) is all available on an open-source repository at www.gitlab.com/pjbca/4GUAV.

REFERENCES

[1] A. Davies, *The War to Remotely Control Self-Driving Cars Heats Up*, Wired, San Francisco, CA, USA, 2019. [Online]. Available: https://www.wired.com/story/designated-driver-teleoperations-self-driving-cars/

[2] M. Harris. (2018). *CES 2018: Phantom Auto Demonstrates First Remote-Controlled Car on Public Roads*. [Online]. Available: https://spectrum.ieee.org/cars-that-think/transportation/self-driving/ces-2018-phantom-auto-demonstrates-first-remotecontrolled-car-on-public-roads

[3] D. Etherington, *Starsky Robotics' Autonomous Transport Trucks Also Give Drivers Remote Control*, TechCrunch, San Francisco, CA, USA, 2017. [Online]. Available: https://techcrunch.com/2017/02/28/starsky-robotics-autonomous-transport-trucks-also-give-drivers-remote-control/

[4] A. Davies, *Nissan's Path to Self-Driving Cars? Humans in Call Centers*, Wired, San Francisco, CA, USA, 2017. [Online]. Available: https://www.wired.com/2017/01/nissans-self-driving-teleoperation/

[5] Mavlink. *Mavlink Developer Guide*. Accessed: Aug. 19, 2020. [Online]. Available: https://mavlink.io/en/

[6] Ardupilot. *Ardupilot*. Accessed: Aug. 19, 2020. [Online]. Available: www.github.com/ardupilot

[7] T. Blacksheep. *TBS Crossfire Nano Specifications*. Accessed: Aug. 19, 2020. [Online]. Available: https://www.team-blacksheep.com/products/prod:crossfire_nano_rx

[8] Intel. *Mavlink Router*. Accessed: Aug. 19, 2020. [Online]. Available: https://github.com/intel/mavlink-router

[9] M. Osborne. *Mission Planner*. Accessed: Aug. 19, 2020. [Online]. Available: http://ardupilot.org/planner/

[10] P. J. Burke, "A safe, open source, 4G connected self-flying plane with 1 hour flight time and all up weight (AUW) <300 g: Towards a new class of Internet enabled UAVs," *IEEE Access*, vol. 7, pp. 67833–67855, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8718270/

[11] *ZeroTier*. Accessed: Aug. 19, 2020. [Online]. Available: https://www.zerotier.com/

[12] *UAVMatrix*. Accessed: Aug. 19, 2020. [Online]. Available: https://uavmatrix.com/

[13] Softmod. *4G/LTE Softmod for the Parrot Disco*. Accessed: Aug. 19, 2020. [Online]. Available: https://github.com/uavpal/disco4g

[14] A. Kryczka, A. Arefin, and K. Nahrstedt, "AvCloak: A tool for black box latency measurements in video conferencing applications," in *Proc. IEEE Int. Symp. Multimedia (ISM)*, 2013, pp. 271–278.

[15] O. Boyaci, A. Forte, S. A. Baset, and H. Schulzrinne, "vDelay: A tool to measure capture-to-display latency and frame rate," in *Proc. 11th IEEE Int. Symp. Multimedia (ISM)*, 2009, pp. 194–200.

[16] O. Boyaci, A. Forte, S. A. Baset, and H. Schulzrinne, "Demonstration of vDelay: A tool to measure capture-to-display latency and frame rate," in *Proc. 11th IEEE Int. Symp. Multimedia (ISM)*, 2009, pp. 444–445.

[17] R. Hill, C. Madden, A. Van Den Hengel, H. Detmold, and A. Dick, "Measuring latency for video surveillance systems," in *Proc. Digit. Image Comput. Techn. Appl. (DICTA)*, 2009, pp. 89–95.

[18] J. MacCormick, "Video chat with multiple cameras," in *Proc. ACM Conf. Comput. Supported Cooperative Workshop (CSCW)*, 2013, pp. 195–198.

[19] Open Developers. *OpenHD Project Github Page*. Accessed: Aug. 19, 2020. [Online]. Available: https://github.com/HD-Fpv/Open.HD

[20] P. Burke. (2019). *OpenHD*. [Online]. Available: https://www.youtube.com/watch?v=MV-ciyX_O5Y

[21] DJI. *DJI LightBridge*. Accessed: Aug. 19, 2020. [Online]. Available: https://www.dji.com/dji-lightbridge/info#downloads

[22] DJI. *DJI Digital FPV System Specifications 2019*. Accessed: Aug. 19, 2020. [Online]. Available: https://www.dji.com/fpv/info#specs

[23] TBS. *TBS ZEROZERO V2*. Accessed: Aug. 19, 2020. [Online]. Available: https://www.team-blacksheep.com/

[24] "Unmanned aircraft system (UAS) traffic management (UTM) concept of operations version 1.0," FAA, Washington, DC, USA, Rep., 2018. [Online]. Available: https://utm.arc.nasa.gov/docs/2018-UTM-ConOps-v1.0.pdf.

[25] A. Kaknjo, M. Rao, E. Omerdic, L. Robinson, D. Toal, and T. Newe, "Real-time video latency measurement between a robot and its remote control station: Causes and mitigation," *Wireless Commun. Mobile Comput.*, vol. 2018, Nov. 2018, Art. no. 8638019.

[26] M. Uitto, M. Hoppari, T. Heikkila, P. Isto, A. Anttonen, and A. Mammela, "Remote control demonstrator development in 5G test network," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, 2019, pp. 101–105.

[27] S. Neumeier and C. Facchi, "Towards a driver support system for teleoperated driving," in *Proc. IEEE Intell. Transport. Syst. Conf. (ITSC)* 2019, pp. 4190–4196.

[28] C. Bachhuber, E. Steinbach, M. Freundl, and M. Reisslein, "On the minimization of glass-to-glass and glass-to-algorithm delay in video communication," *IEEE Trans. Multimedia*, vol. 20, no. 1, pp. 238–252, Jan. 2018.

[29] R. Liu, D. Kwak, S. Devarakonda, K. Bekris, and L. Iftode, "Investigating remote driving over the LTE network," in *Proc. 9th Int. ACM Conf. Autom. User Interfaces Interact. Veh. Appl. (AutomotiveUI)*, 2017, pp. 264–269.

[30] A. Gerber, Z. M. Mao, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks categories and subject descriptors," in *Proc. 10th Int. Conf. Mobile Syst. Appl. Services (MobiSys)*, 2012, pp. 225–238.

[31] R. T. Juang, "The implementation of remote monitoring for autonomous driving," in *Proc. 4th Asia–Pac. Conf. Intell. Robot Syst. (ACIRS)*, 2019, pp. 53–56.

[32] L. Kang, W. Zhao, B. Qi, and S. Banerjee, "Augmenting self-driving with remote control: Challenges and directions," in *Proc. 19th Int. Workshop Mobile Comput. Syst. Appl. (HotMobile)*, 2018, pp. 19–24.

[33] J. Huang *et al.*, "An in-depth study of LTE," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 363–374, 2013.

[34] D. Marpe, T. Wiegand, and G. J. Sullivan, "The H.264/MPEG4 advanced video coding standard and its applications," *IEEE Commun. Mag.*, vol. 44, no. 8, pp. 134–142, Aug. 2006.

[35] U. Saeed, J. Hamalainen, M. Garcia-Lozano, and G. David Gonzalez, "On the feasibility of remote driving application over dense 5G roadside networks," in *Proc. Int. Symp. Wireless Commun. Syst.*, 2019, pp. 271–276.

[36] X. Shen *et al.*, *Teleoperation of On-Road Vehicles via Immersive Telepresence Using Off-the-Shelf Components*. Cham, Switzerland: Springer, 2016, pp. 1419–1433. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08338-4

[37] S. Neumeier, E. A. Walelgne, V. Bajpai, J. Ott, and C. Facchi, "Measuring the feasibility of teleoperated driving in mobile networks," in *Proc. 3rd Netw. Traffic Meas. Anal. Conf. (TMA)*, 2019, pp. 113–120.

[38] M. Dano, *3G/4G Wireless Network Latency: How Did Verizon, AT&T, Sprint and T-Mobile Compare in Q3 2015?*, Fierce Wireless, Washington, DC, USA, 2016. [Online]. Available: https://www.fiercewireless.com/special-report/3g-4g-wireless-network-latency-how-did-verizon-at-t-sprint-and-t-mobile-compare-q3

[39] J. Jansen, "VideoLat—An extensible tool for multimedia delay measurements," in *Proc. ACM Conf. Multimedia (MM)*, 2014, pp. 683–686.

[40] J. Davis, C. Smyth, and K. McDowell, "The effects of time lag on driving performance and a possible mitigation," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 590–593, Jun. 2010.

Peter J. Burke (Senior Member, IEEE) received the Ph.D. degree in physics from Yale University, New Haven, CT, USA, in 1998.

From 1998 to 2001, he was a Sherman Fairchild Postdoctoral Scholar in physics with the California Institute of Technology, Pasadena, CA, USA. Since 2001, he has been a Faculty Member with the Department of Electrical Engineering and Computer Science, University of California at Irvine, Irvine, CA, USA. His current research interests include nanoelectronics, probes of cellular and subcellular electrophysiology, radio and antenna propagation and systems (including nanoelectromagnetics), and drones.

Dr. Burke has been the recipient of the Office of Naval Research Young Investigator Award and the Army Research Office Young Investigator Program Award.