Ground Control to Major TOM

Cameron Mackintosh Zachary Pomper Stanley Zhang
18-500 ECE Design Experience, Team AA
Electrical and Computer Engineering, Carnegie Mellon University
{*cmackint*,*zbp*,*szz*}*@andrew.cmu.edu*

***Abstract*—In the process of building, testing, and eventually competing, FSAE teams may benefit from the ability to wirelessly relay signals from their race cars to ground crews. In particular, commercial CAN streaming solutions are often unsatisfactory for FSAE teams with niche power, form factor, and range requirements. We present our proposed design for an embedded system accomplishing this task. Further, we outline the relevant motivating factors, design considerations, and system constraints.**

Keywords—Automotive, CAN, embedded, PCB, racing, SPI, telemetry, Wi-Fi, wireless, ZigBee, 802.15.4

I. Introduction

Carnegie Mellon Racing (CMR) is Carnegie Mellon University's chapter of the Society of Automotive Engineers (SAE). Since the early 2000s, the team has participated annually in the Formula SAE (FSAE) student design competition. In 2014, the team began competing in FSAE Electric events across North America; each year-long season entails the design, manufacture, and testing of a new Formula-style electric race car.

To aid each car's development cycle, the team has long desired a vehicle telemetry system, through which data could be captured from various on-board sensors and modules. During the 2018 season, a prototype system was created: a "telemetry module" connected to the car's Controller Area Network (CAN) bus, with an Atmel AVR32 microcontroller unit (MCU) and Digi XBee ZigBee radio to communicate with an off-car laptop application (the "base station"). This system unidirectionally relayed internal car state from the CAN bus to the base station.

The ZigBee radio permitted a range of at least 2 km line-of-sight, sufficient for receiving live data during track testing. However, the radio's maximum effective bandwidth was approximately 80 kbps—insufficient for directly streaming the car's CAN bus. Thus, a custom delta-encoding scheme was designed to exploit redundancy in the car's CAN message fields; often, these fields represent sensor values whose rates of change were far slower than the message's actual frequency. Unfortunately, the encoding scheme's tight coupling to the car's message format led to complexity when adding or modifying messages. Furthermore, as the team transitioned in 2019 to a more advanced STMicroelectronics STM32-based MCU, the need for bidirectional communication and more complex base station-to-vehicle interaction became apparent.

CMR's telemetry system goals now encompass all wireless operations with the race car. General-purpose wireless connectivity would not only provide the previous system's real-time data acquisition features, but also allow for remote configuration management and over-the-air firmware updates. Thus, our project aims to redesign the telemetry system as a wireless CAN bridge, linking the car's CAN bus with a remote one. By adhering to the industry-standard CAN protocol, we ensure compatibility with commercial tools and data analysis applications; new client applications can also be easily created for the reconfiguration and firmware features. Through this novel wireless development platform, we hope to provide CMR with new research and development opportunities, ensuring the team remains on the cutting edge of racing technology.

II. Requirements

The telemetry system shall implement the following required functionality:

- The system will wirelessly relay CAN messages.
- The system will interface with the car's grounded low voltage (GLV) system. This entails acceptance of 24 V input, capabilities for transceiving on a 500 kbps CAN bus, and power and space claims of 5 W and 10 in3, respectively.
- The system will support persistent settings and a separate, wired interface for configuring these settings.
- The system will function at a maximum effective wireless bandwidth of 50 kbps at ranges under 1500 m, and at a maximum effective wireless bandwidth of 1 Mbps while ranging under 10 m.

The telemetry system shall not contravene any of the following stipulations:

- The system must be resilient to man-in-the middle attacks. This implies that the wireless data stream's contents cannot be trusted blindly, and incoming message identifiers must be screened against known lists of valid identifiers.
- While dropped packets are expected when modules go offline or ranges are exceeded, communication failures must not compromise system performance or safety.
- The system must not violate any FSAE Electric competition rules.

III. Solution Approach

A. System Components

The telemetry system is composed of at least one "Telemetry Operations Module" (TOM) and optionally one "Ground Control" laptop. The TOM is an embedded system comprised of a custom printed circuit board (PCB) stack physically connected to each CAN bus and is responsible for bridging them together over a wireless link. The TOM uses a custom PCB because it needs to be mechanically robust; form factor requirements rule out commercial-off-the-shelf (COTS) products. The Ground Control is a PC with software for viewing CAN data and vehicle state; it also supports TOM configuration via serial (UART) and CAN links.

The TOM's primary PCB houses power filtering, an MCU, a CAN transceiver, a JTAG programmer, and a UART header. A secondary board breaks out the two radio transceivers, which communicate with the MCU via two Serial Peripheral Interface (SPI) links. This stacked PCB configuration reduces footprint in the car's trunk and manufacturing cost/time for each revision.

The TOM uses an STMicroelectronics STM32F413RG microcontroller [1]. It has all of the required communication peripherals, provides a well-documented hardware abstraction layer (HAL), and is the team's chosen MCU for this and future seasons. Via two SPI links, the TOM's MCU communicates with two radio transceivers: a high-bandwidth, short-range Digi XBee S6B Wi-Fi module [2], as well as a low-bandwidth, long-range XBee-PRO S3B ZigBee module [3]. These modules were chosen as they theoretically fulfill our distance and bandwidth requirements.

B. Range Estimation

The XBee ZigBee module specifies a maximum 6.5 km range when operating at 200 kbps with a 2.1 dBi dipole antenna [7]. However, the XBee Wi-Fi module does not specify a maximum range. Thus, in order to estimate our system's feasibility, we compute the transmit power budget in (1), where the TP is transmitter power in decibel-milliwatts and TAG is transmit antenna gain in isotropic decibels. We compute the receive power budget in (2) where RS is receive sensitivity in decibel-milliwatts and RAG is receive antenna gain in isotropic decibels. We consider the free space path loss (FSPL) in (3), where f is the frequency in megahertz and d is distance in kilometers. The power margin (PM) is defined in (4), and represents the total losses and gains in the system. The system operating margin (SOM) in (5) is an error margin. These equations and terms are discussed further in [4] and [5].

$$TPB = TP + TAG \tag{1}$$

$$RPB = |RS| + RAG \tag{2}$$

$$FSPL = 20\log_{10}(f) + 20\log_{10}(d) + 32.44 \tag{3}$$

$$PM = TPB + RPB - FSPL \tag{4}$$

$$SOM = \frac{PM}{TPB} \tag{5}$$

By solving (4) and (5) for FSPL and (3) for d, we arrive at a maximum range estimation in terms of transmit/receive power budgets (TPB and RPB) and operating margin (SOM):

$$FSPL = TPB(1 - SOM) + RPL \tag{6}$$

$$d = 10^{(FSPL - 20\log_{10}(f) - 32.44)/20} \tag{7}$$

Fig. 1 summarizes our preliminary Wi-Fi range estimates. We considered a 9 dBi whip antenna [6], and a 3 dBi multi-band blade antenna [7]; these antennas are electromagnetically compatible with the Wi-Fi XBee and had the highest gain for their price range. The multi-band antenna is particularly appealing due to its low profile and 900 MHz (ZigBee) capability, reducing vehicle mounting complexity.

TABLE I. Wi-Fi Range Estimates

Data Rate (Mbps)	Ant. Gain (dBi)	Tx. Pwr. (dBm)	Rx. Sens. (dBm)	Distance (km)
6.5	9	15.0	-91	6.88
6.5	3	15.0	-91	2.13
65.0	9	8.5	-71	0.41
65.0	3	8.5	-71	0.13

Fig. 1. Ideal range estimates for 2.4 GHz Wi-Fi at the lowest and highest 802.11n data rates supported by the Wi-Fi XBee [6]. It is assumed that the transmitter and receiver use symmetrical antenna systems.

C. Firmware Features

In the Wi-Fi communication mode, the TOM supports full CAN bridging and higher-bandwidth add-on applications, such as car firmware flashing. However, these extensions are outside of our project's scope. On the other hand, the ZigBee mode supports filtered CAN bridging, wherein a subset of the full message stream is sampled for transmission. The Ground Control is responsible for activating the desired radio mode in the TOM's configuration. The race car will typically use Wi-Fi mode, with ZigBee mode manually selected as a fallback in the event of Wi-Fi range issues.

The TOM firmware uses the FreeRTOS kernel [8], as it ensures our timing requirements are met and is a well-documented industry and CMR standard. The firmware is primarily responsible for transcoding data between the CAN transceiver and the selected XBee transceiver. It uses a CAN firewall to ensure relayed messages are not sent to blacklisted local devices. The firmware further uses a custom CAN codec to better utilize the available transceiver bandwidth.

IV. Architecture

As previously described, the TOM functions as a CAN bridge. A full bridging setup relays CAN messages either between two connected TOMs, or between a TOM and a Ground Control laptop. Each TOM receives CAN messages, encodes them via a custom compression scheme, and sends encoded packets over SPI for wireless transmission by either the ZigBee or the Wi-Fi radio. Through a UART port, a user can configure the TOM's settings and access debugging facilities. These features are implemented by firmware modules flashed onto the on-board MCU, including peripheral interfaces, message codecs, and application programming interfaces (APIs) for communicating with either of the two radios. (See Appendix A for a system block diagram.)

Our firmware architecture (Fig. 2) makes use of message queues, interrupts, and periodic tasks—all facilitated by FreeRTOS [8]. CAN messages are received into a small, memory-mapped hardware queue; the CAN receive task, polling at the maximum frequency of 1 kHz, enqueues the messages into a larger encoding queue. The message encoding task uses the codec to format as many CAN messages as possible into packets placed on the SPI transmit queue. (The frequency of this task, 50 Hz, is governed by the worst-case latency on the receiving side.) From there, an XBee transceive task sends the packets to the radio module via SPI.

However, as soon as the XBee transceive task begins to send to the radio, it must also receive data as part of the full-duplex master-slave SPI protocol. While the receiving part of the task could be periodic, the transceive task also needs to activate when a radio has pending received data (i.e. in response to radio interrupts). Thus, the transceive task aperiodically places received packets in a decoding queue, from which CAN messages are extracted using the codec by the decoding task. This task runs at no more than 200 Hz, where this frequency is determined by the tolerable latency between a message arriving on the radio and its transmission onto the CAN bus. Finally, the CAN transmit task, running at the maximum frequency of 1 kHz, performs this transmission of decoded CAN messages.

CAN's arbitration protocol cannot resolve two nodes transmitting a message with the same identifier (ID). By accumulating a list of received IDs, a firewall prevents transmissions with IDs known to originate from other nodes on the bus. This provides security against an attacker requesting transmissions with conflicting IDs, which could cause a safety-critical message to miss its deadline. This firewall, like the other settings available, can be configured through UART; this serial link also allows the TOM to send out debugging information while still interfacing with the CAN bus.

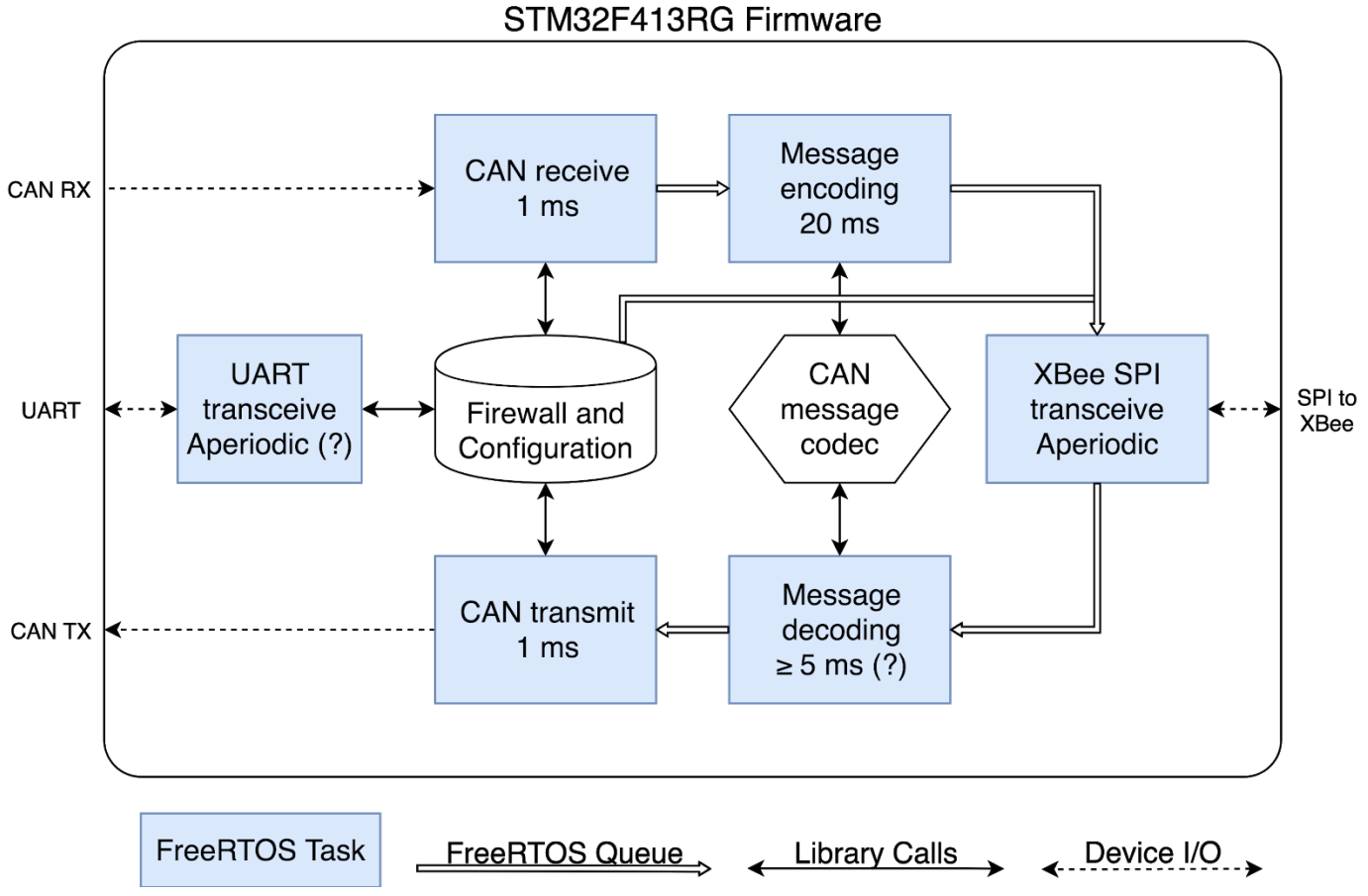


Fig. 2. Microcontroller firmware block diagram.

V. Implementation

We divide our implementation into hardware, including circuit design, PCB layout, and electrical verification, and firmware, including programming the C code that will run on the MCU, verification of said code, and system integration with CAN networks.

A. Hardware

Our implementation is composed of two hardware systems, namely, a primary board which houses the MCU, and a secondary board with the two radios. The MCU board makes use of an in-house standard component package to give the MCU 3.3 V power from a 24 V input, and to support an MCP2561 CAN transceiver. It adds MCU connections for two SPI buses, three UART buses, and of course the CAN bus, along with the relevant off-board connectors for those connections. The radio breakout board comprises the radios themselves, a connector to the MCU board, and user interface switches and LEDs. Each board is connected over a 32-pin ribbon cable. This cable has pins for two SPI and UART buses, several pins dedicated to power rails, pins for radio interfacing like the SPI message receive interrupt line, and some floating pins to be wired if the need arises.

The boards themselves are realized with custom PCB layouts. They use two power layers and no signal layers to cut costs at the expense of signal integrity. This loss in signal integrity is difficult to quantify within our production budget and timeframe, but has been demonstrated to be acceptable via the function of revision 0 of the MCU board, which is also routed on two layers. Moreover, only a select few of the components on either board implement functionality highly sensitive to signal integrity loss, such as analog signals or high-speed traces. To reduce impedance, copper pours are used in place of power planes when necessary, namely for higher current paths.

In terms of assembly, the radio board stacks on top of the MCU board via plastic spacers between aligned mounting holes. The radio board uses on-radio RP-SMA RF output ports, to be wired either directly or via coaxial cable to a high-gain antenna. The whole assembly will mount to the base of a waterproof box not supplied by our project, given in the use case of CMR's 2019 electric vehicle.

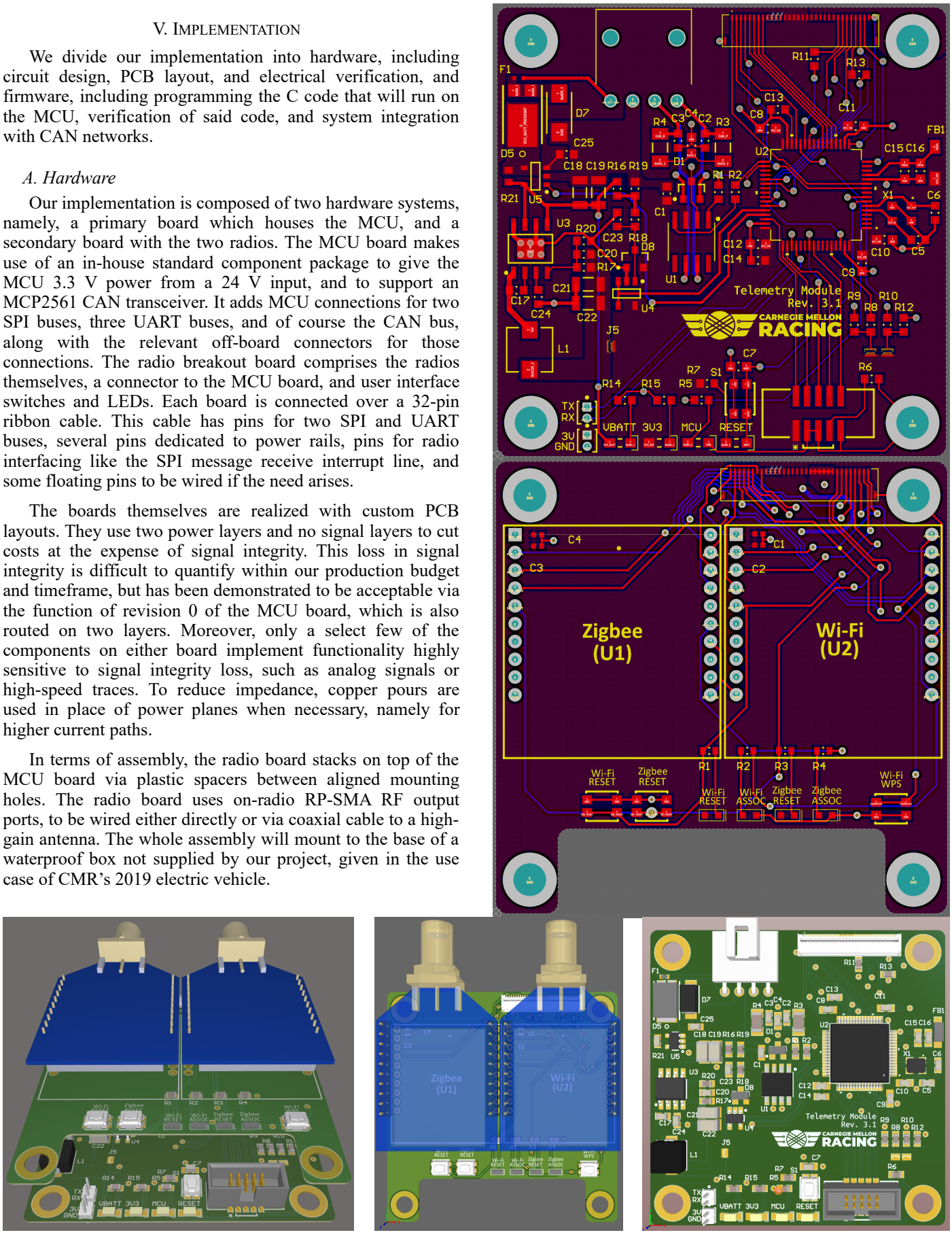


Fig. 3. From top to bottom: primary MCU board layout; secondary radio transceiver board; 3D renders of the board layouts.

B. Firmware

As previously discussed in our system architecture, our firmware's main purpose is to interface between the CAN and XBee transceivers. Thus, we have designed and implemented a driver and supporting libraries for managing SPI-based XBee communications. The most challenging aspect of designing this driver was the full-duplex nature of SPI: the XBee (i.e., the SPI slave) is permitted to transmit data whenever the MCU (i.e., the SPI master) has asserted the XBee's slave-select line and is driving the clock. This asynchrony is necessary, for example, when the XBee has received a radio packet and wishes to present it to the MCU.

However, because of SPI's master-slave architecture, slave devices cannot directly that they have data pending for receive by the MCU. To work around this, the XBees output a "SPI attention" signal [2][3] that is asserted under such conditions. We use this signal to trigger an interrupt on the MCU, which proceeds to handle it by waking the XBee transceive task when necessary. Notably, this task must handle both communication directions to be correctly full-duplex. All transmitted packets must be accompanied by an equally-sized receive buffer; otherwise, a frame arriving in the middle of a transmission would be dropped—significantly tarnishing our link's robustness.

Synchronization issues arise due to the arbitrary timing of both transmitted and received messages. FreeRTOS queues [8] are used throughout our implementation, as they are a convenient abstraction for synchronized message passing between tasks—a necessary feature in the producer-consumer models we have chosen. Semaphores [8], both binary and counting, are also used to wait for and signal various events (typically, the completion of a message).

As for the CAN infrastructure, we have incidentally performed much of the necessary driver implementation as part of a general CMR firmware bring-up effort. However, while most of the vehicle's nodes are concerned with sending and receiving periodic messages, we are taking a more generic approach to CAN. Thus, we have added general-purpose hooks for handling any type of received CAN message, and have implemented a CAN transmission interface that supports both periodic and aperiodic use cases. Furthermore, as shown in our firmware block diagram in Fig. 1, our implementation adds several new layers above the CAN driver for processing messages.

The CAN codec is responsible for losslessly compressing and decompressing CAN messages for better wireless bandwidth utilization. It operates on blocks of CAN messages for increased compression ratio. The block size is maximized such that latency is still considered acceptable and the XBee Wi-Fi module's 1400 byte maximum transmission unit (MTU) [2] is not exceeded in order to minimize packet header overhead. The codec initially applied an augmented run-length (ARL) algorithm that selectively encodes run-lengths greater than a threshold. A run-length encoding is three bytes: a "start command" that indicates the start of the encoding, a run-length, and the running byte. "Transposed" start and escape command bytes are also present in case these byte values appear in the uncompressed block.

The codec was updated to apply an embedded implementation of the Lempel–Ziv–Storer–Szymanski (LZSS) [9] dictionary encoding algorithm. The combined ARL and LZSS strategy achieves a compression ratio of 0.15 to 0.22, depending on the size and contents of the uncompressed message block. Fig. 4a shows statistics for several codec algorithms using a CAN message trace from CMR's 2018 vehicle. The number of CAN messages per block may be increased above the Wi-Fi MTU for improved compression ratios. Fig. 4b shows the ARL+LZSS and LZSS compression ratios at various block sizes. LZSS is slightly more efficient than ARL+LZSS for all block sizes, as it implicitly run length encodes. The ARL compression ratio is omitted, as it is independent of block size and does not perform well compared to the other techniques.

TABLE II. CODEC ALGORITHM COMPARISON

	Mean	Std. Dev.	Max.	Compression Ratio
ARL 170 msg/block	1357	11.5	1387	0.895
LZSS 800 msg/block	1144	83.4	1373	0.1699
ARL+LZSS 800 msg/block	1147	88.4	1382	0.1702

Fig. 4. Statistics of several compression algorithms. The number of CAN messages per block is chosen such that the maximum compressed block size is less than the Wi-Fi 1400-byte MTU.

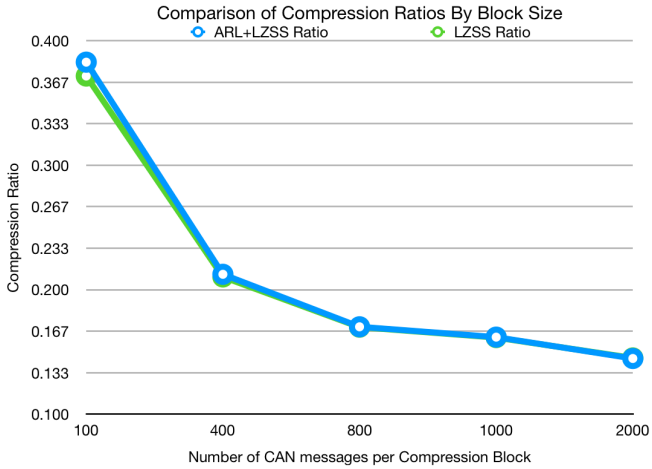


Fig. 5. Comparison of compression ratios for ARL+LZSS and LZSS.

The firewall is responsible for ensuring the safety and security of the CAN bus attached to the TOM. The firewall supports an implicit blacklist, a temporal blacklist, and an explicit blacklist. The firewall monitors the local CAN bus and populates the implicit blacklist with CAN IDs. It blocks all remote CAN messages that have an ID on this blacklist, as duplicate IDs across the bridge imply either a spoofing attempt or an unsafe configuration. The firewall may be configured with a temporal blacklist to block all remote CAN messages for a brief period to ensure it has fully populated the implicit blacklist; this functionality is particularly useful when all CAN messages are periodic. The firewall may also be configured with an explicit blacklist that blocks specified local CAN messages from bridging to the remote CAN bus.

The configuration system manages persistent settings stored in the TOM MCU's on-chip flash memory. This system is general-purpose, allowing the lower-level settings and flash management drivers to be used on other CMR boards that

require persistent settings. However, whereas most boards manage their settings solely through their CAN interfaces, the TOM also uses a UART link for this purpose. Through this link, the Ground Control software configures and monitors the TOM. Configurable settings include the firewall blacklist, the codec block size, the codec algorithm, the code algorithm-specific configuration (such as LZSS's number of lookahead bits), the radio communication mode, the radio-specific configuration (such as Wi-Fi MCS index). The system also has provisions for extension by future revisions.

VI. Testing

In order to measure our telemetry system's fulfillment of the previously specified requirements, we indicate the following metrics and methods of validation.

A. Metrics

The following table presents our most important metrics and targets.

Table III. Success Metrics

Metric	Target
CAN data rate	500 kbps, matching vehicle CAN bus
RF data rate @ 10 m	1.00 Mbps, full CAN bus
RF data rate @ 1500 m	0.05 Mbps, downsampled
Message spoofing robustness	Messages never sent with same-side IDs
Power consumption	< 5 W, passively cooled
Package size	< 10 in^3, fits in GLV trunk
Board and BOM costs	< $250 per module

Fig. 6. Important success metrics and our targets for the system.

These metrics have been derived from our requirements and anticipated use cases.

B. Validation

To verify that the TOM's CAN interface is compatible with a 500 kbps CAN bus, we will use CAN interfacing software to send and receive messages at that baud rate. To verify that the TOM can stream the entire 500 kbps bus, we will fully load each side of the bus using this aforementioned software, and verify that messages are received with no or minimal dropped messages on the other side.

Range testing includes both proof-of-concept tests with XBees attached to development setups (primarily laptop PCs with USB-UART adapters), as well as approximate recreations of the competition environment with the system's hardware mounted on the race car at track testing. Some initial testing has already been performed with a preliminary set of antennas; several locations have already been identified for further validation.

To characterize our RF data rate, we will make use of the fact that the SPI-to-radio link has a much higher bandwidth than each wireless link, and flood the SPI bus with outgoing messages while monitoring packet loss on the other side. We will sweep our packet size to determine the optimal packet size for each of {50, 1500} meter ranges.

To verify the TOM's robustness to spoofed messages, we will send a variety of messages conflicting with those previously sent on the other side of the wireless link and ensure that said messages are filtered by the TOM.

To test our power consumption, we will measure the current over an on-board shunt resistor at a known input voltage, and attempt to fully load the transmit bandwidth of both radios. We will assume that this power measurement encapsulates the TOM's worst-case power consumption, and, provided nothing is erroneous, we will not enforce or even monitor power consumption in software. In the even that this test exceeds our power budget, faulty circuitry is most likely the problem, so the solution will probably be to do hardware debugging and not software enforcement.

Verifying our package size and project cost metrics will be trivial; the TOM will be test-fitted into the trunk, and the final budget will be calculated for the last revision.

VII. Project Management

A. Schedule

Appendix B contains a Gantt chart outlining our project's timeline. The top-level tasks are TOM PCB creation, firmware implementation, and Ground Control development. Each of these tasks are split into modules corresponding to logical components in the system architecture. Our development has largely proceeded according to this schedule; we hope to remain on track with our implementation and testing cycles for the upcoming PCB revision.

B. Responsibilities

Cameron Mackintosh is in charge of writing the communication protocol firmware, i.e. packetization and transcoding of CAN messages. Zachary Pomper is in charge of PCB design and layout, as well as electrical verification. Stanley Zhang is in charge of writing firmware interfaces for communication between the MCU and the radios, between the MCU and user-facing UART, and for communicating between the MCU and each side of the CAN bus. We assume joint responsibility for RF design (namely antenna selection and range testing), integration, and manufacturing.

C. Budget

The following are allocations out of our $600 project budget. Some of these allocations have already been purchased from this budget, some have been funded externally, and some were given to us free of charge. In other words, the following table estimates the combined value of components used by our project upon completion.

Table III. Success Metrics

Item	Cost (USD)
Radio boards (x4)	44.00
MCU boards (x4)	44.00
Omni-directional antenna	20.00
SMD components	200.00
Directional antenna	50.00
Misc. RF components	10.00
ZigBee RF modules (x4)	60.00
Wi-Fi RF modules (x4)	60.00
Total	**488.00**

Fig. 7. Important success metrics and our targets for the system.

Board manufacturing cost quoted from PCBway, where there are two revisions of each board being budgeted. Each of these purchases comes with four free bonus PCB copies. SMD

component costs are aggregated across four module instances, allowing us to make two modules within each revision (as the extra boards are themselves free). While component BOMs may be transferable by salvaging a revision's parts, we assume that this transfer would incur an infeasible time overhead. RF modules, on the other hand, do not have to be soldered to the PCB, and are thus accounted for across only one revision.

VIII. Risk Management

In order to mitigate risks of not meeting the success targets we have outlined, we consider several fallback strategies for each metric. In case the CAN message rate exceeds our radio link's supported bandwidth, we expect that downsampling periodic, lower-priority messages will permit a meaningful, gracefully-degraded approximation of the full CAN bus traffic. As for range issues, the ZigBee mode allows us to provide this downsampled stream at a sufficiently long line-of-sight range.

The firewall is only as secure as the interfaces present to configure it; thus, we do not allow any configuration over the wireless link, and only trust the wired UART and CAN interfaces. Power diagnostics, other statistics, and debugging aids are also available over the UART link and in a configurable CAN heartbeat, providing ample support for validating correct operation. Finally, the stacked board layout was specifically chosen to fit in the vehicle's low-voltage trunk; their small nature and two-layer layout also helps in reducing our hardware costs.

IX. Related Work

CAN bus streaming solutions are not abundant on the open market, but they do exist. Among the most relevant for FSAE teams that we were able to find would be the PEAK CAN to WLAN Gateway [10], the ESD CAN-CBX-AIR/2 [11], and HRI's Vehicle Safety Controller (VSC) [12]. PEAK's solution houses an internal antenna with unspecified output power, making it unsuited for our range specification. ESD's CAN-CBX-AIR/2 has an external antenna connector, but has a transmit power of 0 dBm, limiting its maximum range. HRI's VSC was the most promising option on the market we found, meeting our range and bandwidth requirements; it is also IP66-rated, obviating the need for an enclosure on the car. A talk with HRI revealed that their CAN bridging solution runs at a price point of $1250 per module (no sales information is on their website); this puts it out of contention for CMR's purposes, but the device's features are still quite noteworthy. Additionally, a closed-source solution like any of the above is probably unsuited to the FSAE space, where limited product runs being met with ends-of-life can leave teams floundering.

X. Summary

The TOM platform is designed to fill a niche for FSAE teams using CAN bus who would like a wireless interface into their car while testing and during competitions. The platform iterates on functionality provided by the existing telemetry system in use by CMR, and adds functionality by way of CAN bridging at higher bandwidths. Its design is realized by way of custom hardware and firmware implementations which interface with an STM32F4 MCU. Users can make use of the TOM in one of two configurations: either as a CAN bridge between two TOM's, or between a single TOM and a Ground Control station. The former allows for system integration out of the hardware loop of a CAN bus, and the latter allows for users to view and modify the state of a CAN bus remotely.

The application package running on the Ground Control station can greatly enhance user experience. While we likely will be contributing to this package in anticipation of the 2019 FSAE season, and it is an important part of our system's overall design, the software itself is not in scope for our project. The addition of a Web-based view into data streaming over the TOM could prove useful and interesting to team members present at races.

Range limitations over Wi-Fi are largely a product of our choice of radio. These limitations could be increased significantly through the usage of RF amplifiers, either mounted in-line or on the transceiver board. We decided against the former on account of unit costs, and against the later on account of our collective inexperience with RF PCB design practices.

Acknowledgment

Thanks to all Carnegie Mellon Racing team members who have supported this project, not only throughout the course of this semester, but also since its inception during the 2015 season. Our appreciation goes especially to Sam Westenberg, Deepak Pallerla, Bolaji Bankole, and Ben Yates for their invaluable feedback, support, and mentorship.

References

[1] STMicroelectronics, N.V. "STM32F413xG STM32F413xH." Internet: https://www.st.com/resource/en/datasheet/stm32f413rg.pdf, [Feb. 1, 2019].

[2] Digi International. "XBee Wi-Fi RF Module User Guide." Internet: https://www.digi.com/resources/documentation/digidocs/PDFs/90002180.pdf, [Feb. 5, 2019].

[3] Digi International. "XBee-PRO 900HP/XSC RF Modules." Internet: https://www.digi.com/resources/documentation/digidocs/pdfs/90002173.pdf, [Feb. 5, 2019].

[4] ZyTrax, Inc. "Tech Stuff – Wireless Overview." Internet: http://www.zytrax.com/tech/wireless/intro.htm, Oct. 23, 2015 [Feb. 5, 2019].

[5] "Free Space Path Loss | Details & Calculator." Internet: https://www.electronics-notes.com/articles/antennas-propagation/propagation-overview/free-space-path-loss.php, [Feb. 5, 2019].

[6] "ANT-2WHIP9-SMARP RF Solutions." Internet: https://www.digikey.com/product-detail/en/rf-solutions/ANT-2WHIP9-SMARP/ANT-2WHIP9-SMARP-ND/9555705, [Feb. 5, 2019].

[7] "MAF94300 Laird Technologies IAS." Internet: https://www.digikey.com/product-detail/en/laird-technologies-ias/MAF94300/994-1125-ND/3511618, [Feb. 20, 2019].

[8] Amazon Web Services, Inc. "The FreeRTOS™ Reference Manual." Internet: https://freertos.org/Documentation/FreeRTOS_Reference_Manual_V10.0.0.pdf [Feb. 1, 2019].

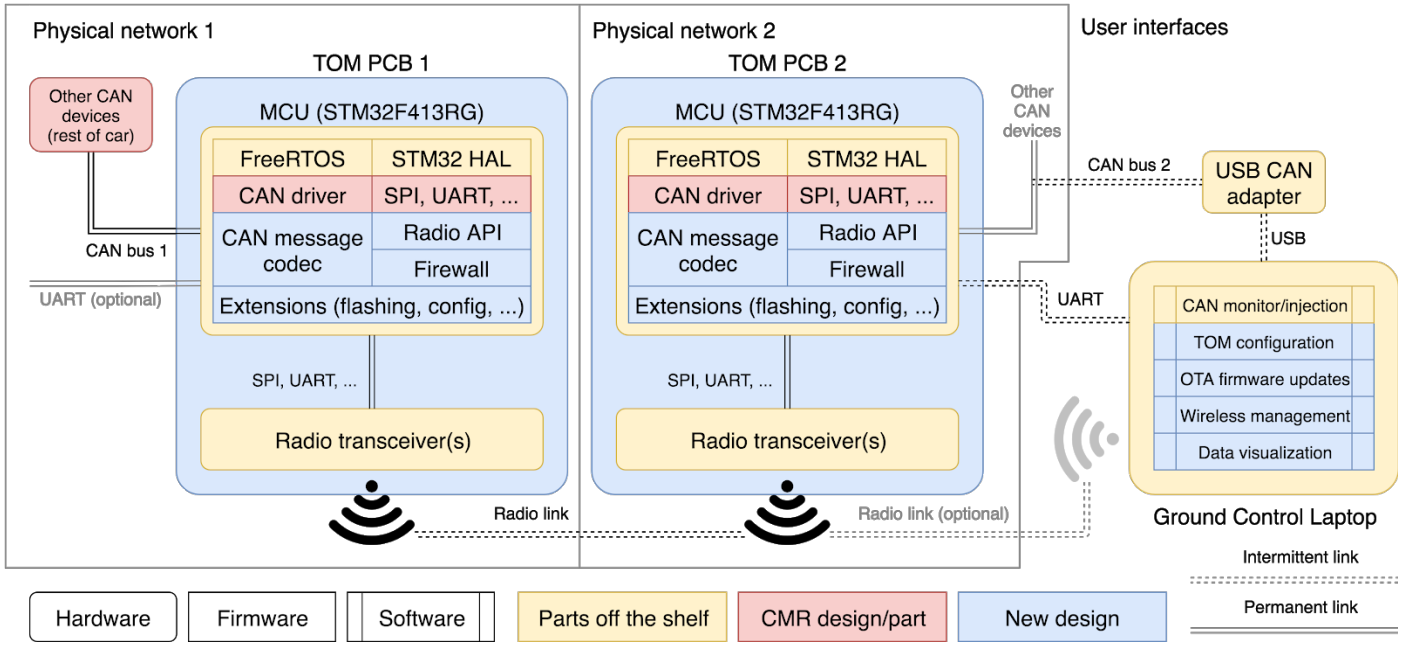
[9] Atomic Object. "heatshrink: An Embedded Data Compression Library." Internet: https://spin.atomicobject.com/2013/03/14/heatshrink-embedded-data-compression/, [Mar. 14, 2013]

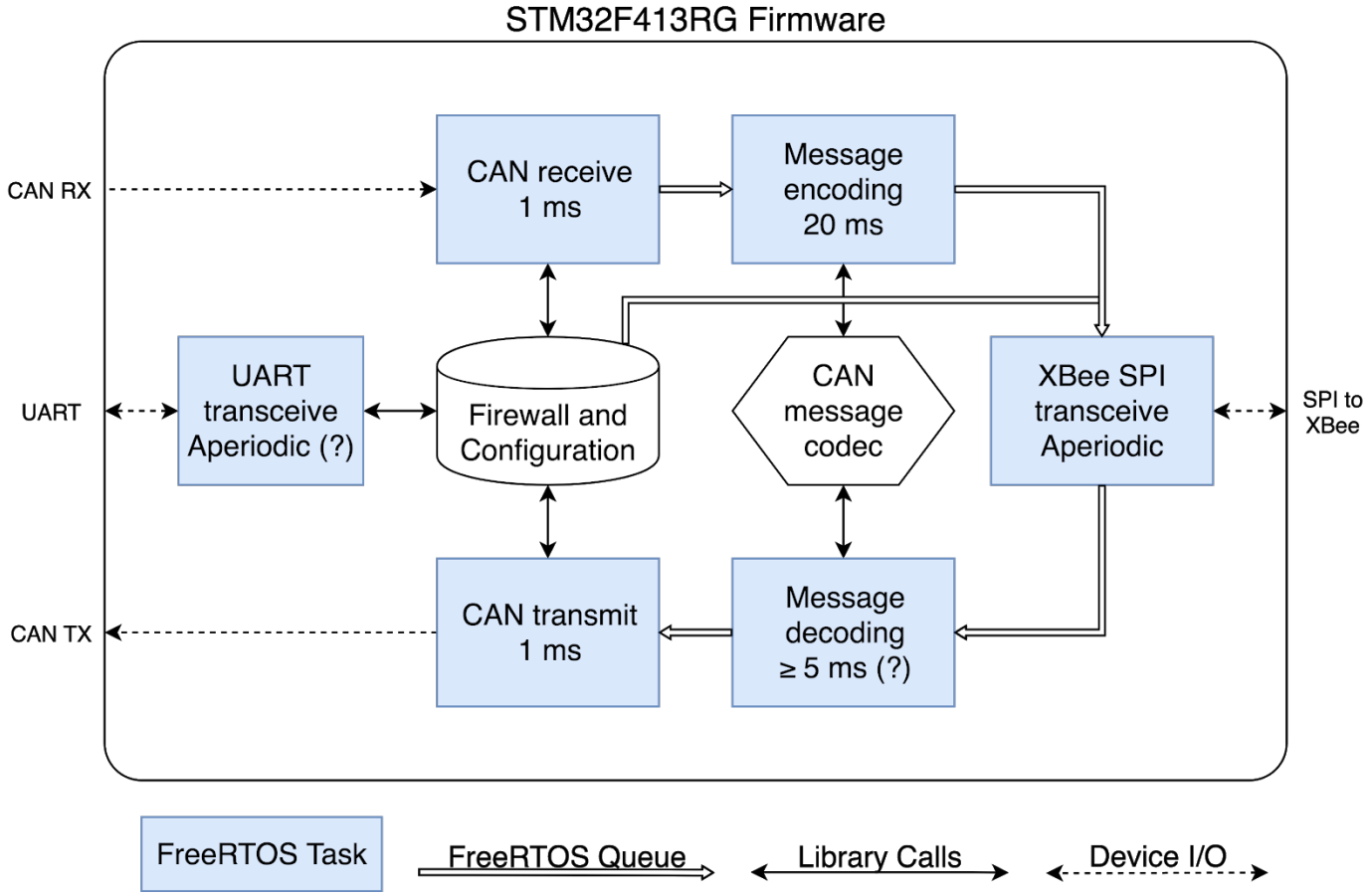
[10] PEAK-System Technik GmbH, "PCAN-Wireless Gateway." Internet: https://www.peak-system.com/PCAN-Wireless-Gateway.331.0.html, [Mar. 2, 2019].

[11] esd electronics. "CAN-CBX-AIR/2 | Wireless CAN Bridge with USB Interface." Internet: https://esd.eu/en/products/can-cbx-air2, [Mar. 2, 2019].

[12] Humanistic Robotics, Inc. "Vehicle Safety Controller." Internet: http://humanisticrobotics.com/vehicle-safety-controller/, [Feb. 17, 2019].

Appendix A: System Block Diagram





Appendix B: Gantt Chart