Modular PCB for John Deere Equipment

Sumandera Sanyal, Samuel Huhta, Zachary Hoegberg
Team 17
TA: W. David Null
8 May 2020

Abstract: John Deere currently manufactures an autonomous residential lawn mower called the Tango, which uses a buried wire to detect the boundary of a given yard. There are several inherent problems with this system, but eliminating the boundary wire requires replacing the entire VCU. John Deere proposed a modular design that could also be used to automate their other equipment, consisting of a universal main board, a machine-specific vehicle board, and a sensor-specific perception board, all of which use Ethernet to communicate with each other. We were tasked with designing a prototype of one board, and for our second project, we chose the Tango's vehicle board.

1. Second Project Motivation

1.1 Updated Problem Statement

John Deere currently manufactures an autonomous residential lawn mower, the Tango, that uses a buried wire to define the boundary of the yard. Installing this wire is a significant cost, requiring a site visit by specially-trained Deere technicians, and the related technology severely limits the mower's capabilities. For these reasons, John Deere is actively working to eliminate the boundary wire by using a localization algorithm fusing several sensors, such as stereo cameras, GNSS receivers, and ultrasonic sensors. However, the current Vehicle Control Unit (VCU) that operates the mower does not have the necessary processing power to run their localization and automation algorithms. Furthermore, John Deere would like to be able to apply their developed automation solution to other implements the company produces, but the current hardware is specific to the residential mower. John Deere would like to replace the VCU with modular boards that run Linux so their software engineers can write Linux apps instead of programming a board specific microcontroller. The boards will demonstrate modularity if at least one board can be used across several vehicles (the VCU can only be used on the Tango).

1.2 Updated Solution

We have decided to replace the existing VCU with a modular design consisting of a main board, a perception board, and a vehicle board. The main board will handle the high-level autonomy universal to all equipment, the perception board will apply machine-learning algorithms to interpret the data provided by a specific set of sensors, and the vehicle board will handle all low-level algorithms specific to a given piece of equipment. John Deere will be able to reuse this design in other equipment by simply swapping out the perception and vehicle boards, designing new ones as they implement different vehicles and sensors. To use the same main board across several vehicles, John Deere will have to write vehicle-specific programs to run on the main board. For our second project, we will design the vehicle board for the Tango mower (our first project dealt with the main board). The Tango vehicle board will demonstrate the modularity of the system by communicating with the main board to run the low-level vehicle control software.

1.3 Updated High-Level Requirements

- The 1-core processor on the vehicle board will boot a Linux operating system and run the provided algorithms to compute motor commands and mower state feedback at a minimum update rate of 20 Hz.
- The vehicle board will control the rotation speed and direction for all motors based on commands from the main board, ensuring that no motor spins faster than 40 rpm.
- The vehicle board will communicate with the main board via Ethernet through an off-the-shelf Ethernet switch, using standard Ethernet protocol, at a minimum update rate of 20 Hz. It will receive linear and angular velocity drive commands and blade motor commands from the main board, and it will provide feedback on vehicle odometry and blade motor state to the main board.

1.4 Updated Visual Aid

Figure 1: Tango mower, typical application

1.5 Updated Block Diagram

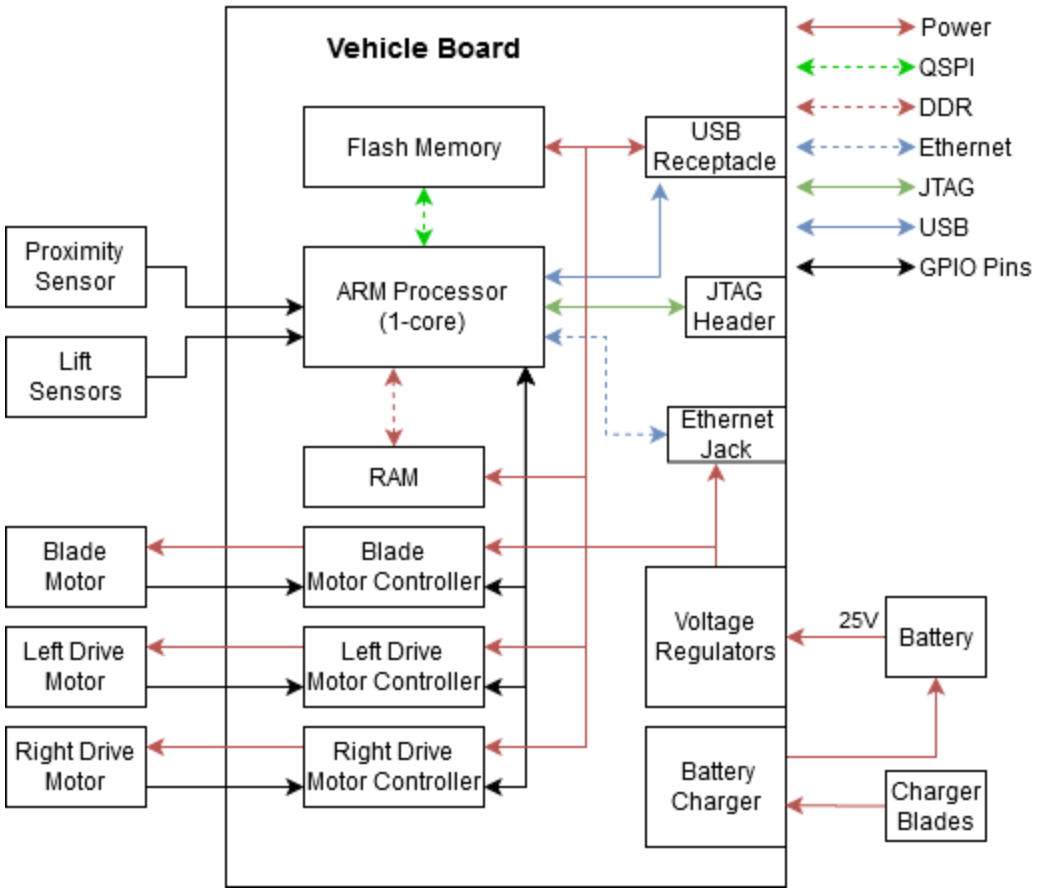

Figure 2: Block diagram for the vehicle board

2. Second Project Implementation

2.1 Implementation Details and Analysis

Since we didn't have time to design the actual board itself, our implementation focused on making sure we could have the required connectivity with the components we selected. We identified six processor features that provide the functionality we need, but most of them share signals with each other, so we had to find a way to connect everything without any overlap.

Of the six processor features mentioned above, the DDR3L SDRAM controller is the only one that doesn't overlap with anything else. This means we can use it alongside anything else we need to use. Six of the seven JTAG signals share pins with a GPIO unit, but the JTAG chain can be disabled by tying the seventh JTAG signal low, which frees up those six pins for other signals.

Unfortunately, all GPIO pins are also used by other signals, and based on the schematic for the provided motor controller circuit, we will need a total of 27 GPIO pins to drive the motors. Thankfully, the LS1012A has two GPIO units with a combined total of 50 pins. 6 of these pins are used by the QSPI Flash memory controller, 2 are used by the USB controller, and 15 are used by the Ethernet interface, leaving exactly 27 pins for GPIO! One of these pins can be used as a clock output, which we don't need, and another doubles as a tamper detector, but only when tamper detection is enabled, so we have just enough GPIO pins to run all of the motor controllers. Unfortunately, this means we can't run anything else on GPIO unless we free up some other signals first, which may be required in order to support the hardware needed to calculate vehicle odometry.

Getting the required connectivity for the JTAG chain is easy: we just need to connect the corresponding pins to a JTAG header, which is simply a 5x2-pin connector (either THT or SMD, depending on your PCB layout). Other connections will require a little more thought; for example, the Ethernet jack must be connected to an Ethernet transceiver, which then connects to the processor's Ethernet interface pins. The Ethernet transceiver we choose must be able to support a communication rate of at least 20 Hz, as well as Ethernet protocol compatible with the chosen Ethernet switch, in order to demonstrate proper Ethernet functionality.

We decided to reuse the Flash memory chip from our first project, and as a result, the connectivity was also unchanged. As shown below, this consists of a clock signal, a chip select bit, and four bits of data. Since we used this chip in our first project, we already had the KiCAD model for it, so drawing up the schematic was just a matter of creating a symbol to represent the QSPI interface of the LS1012A and connecting the corresponding pins. This is a critical element of our design, as the LS1012A can only boot Linux from Flash memory.

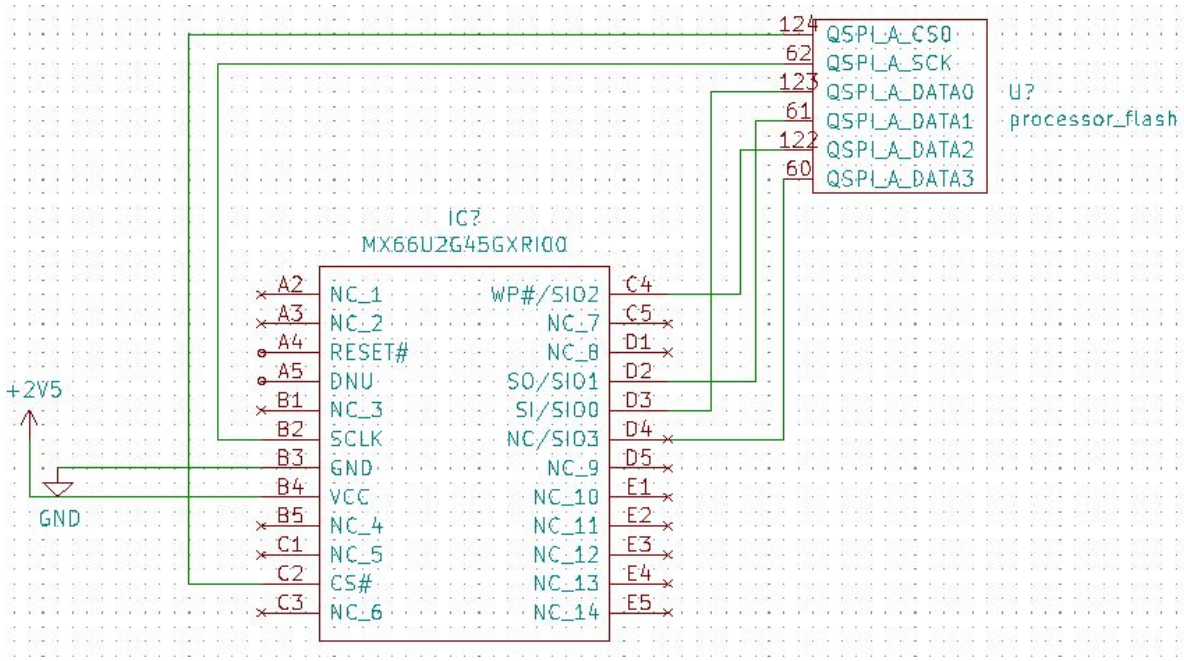

Figure 3: Flash memory setup (simplified)

Since our new processor contains a DDR3L DRAM controller instead of a DDR4, we had to use a different DRAM setup. We still used four 16-bit-wide chips wired together in parallel, since we still need a total bus width of 64 bits, but the LS1012A only has 16 DRAM data pins, indicating that the DDR3L controller doesn't send an entire value (64 bits in our case) all at once. Due to our lack of experience with different DRAM technologies, we took the same approach as in our first project: using the same DRAM chips as one of the processor's reference boards, in the same arrangement, to ensure that they're compatible.

3. Second Project Conclusions

3.1 Implementation Summary

Even though we weren't able to design the vehicle board for the Tango, we're confident that the components we selected can provide the connectivity needed to satisfy our high-level requirements without any signal overlap. The only possible conflict is in relation to the GPIO units, which share pins with signals that we need, but we have just enough pins available to run all three motor control circuits (including feedback provided by the motor controllers).

3.2 Unknowns, uncertainties, testing needed

The biggest uncertainty in our project is the ability to run Linux and scripts and programs on our board that demonstrate the desired functionality for the main board and vehicle board. This is because none of us have the experience in building a board and booting Linux on it, using a JTAG debugger to configure a first time boot, or writing the test scripts that would need to run on the board to demonstrate functionality. We would require support from John Deere software/hardware engineers in developing test scripts to run on the board and booting Linux.

Another uncertainty is the board design surrounding an ARM processor. The ARM processor is a sophisticated integrated circuit with thousands of pages of documentation; we are closely following the design in the reference design board(~50 pages,[2]) and design checklist(~50 pages,[4]) for constructing the board design. However, as none of us have actually gone through the process of building and testing a board with a ARM processor before, we would likely need to revise our design, reorder the board and test it once or twice before we actually had a functional prototype (at least that's how we were scheduling our board design before the COVID situation). Furthermore, we would likely need to get some help from NXP (the company that sells the LS1012A and LS1046A processors) engineers and John Deere engineers who are well-versed in booting Linux on ARM processors and the board design surrounding a ARM processor.

3.3 Ethics and Safety

Lawnmowers have a certain level of inherent safety risk due to the spinning blades, and in order to mitigate these risks our design will still follow guidelines from the IEC 60335-2-107 standard regarding battery-powered lawnmowers. However, our project should not have any direct impact on public safety, as the boards we develop will be for internal John Deere testing and development only, and will go through several revisions by their engineers before any possible production runs. Despite this, our design will still ensure that existing safety features on the Tango mower continue to function as expected.

In accordance with IEEE Code of Ethics #7, "to seek, accept, and offer honest criticism of technical work," we will continue to meet with John Deere engineers weekly to provide updates on our progress, ask questions, and seek feedback on our work.

In accordance with IEEE Code of Ethics #9, "to avoid injuring others … by false or malicious action," we have signed an NDA (non-disclosure agreement) with John Deere regarding the proprietary information they have shared with us to develop our project.

3.4 Project Improvements

Though incomplete (and not a functional unit by itself), our vehicle board is a significant improvement over the current VCU of the Tango mower. For example, the LS1012A is much faster than the current processor (1 GHz vs. the current 72 MHz) and can boot Linux, providing plenty of room for John Deere to improve the mower's capabilities in the future. It also supports communication over Ethernet, opening the doors for improved modularity (3 boards vs. the current 1 board) and thus improved versatility (each board can be redesigned independently).

4. Progress on First Project

Our original project focused on the main board, since its functionality is key to the success of the overall (modular) design. We were able to make significant progress on this board before losing access to the required testing equipment due to COVID-19, and we plan on presenting what we finished to John Deere to give them a head start on designing the actual board.

The quad-core ARM processor we chose for the main board has a ball-grid array (BGA) package, for which we were unable to find a socket. However, mounting a BGA directly to a PCB requires heating said PCB using equipment not available in the ECE building, and replacing a BGA requires reheating the entire PCB, which could damage other components. To prevent this issue (thereby extending the lifespan of our design), we decided to modularize the main board itself. Each sub-board was to contain the components needed for a specific process, and Zach started making a KiCAD schematic for the power sub-board.

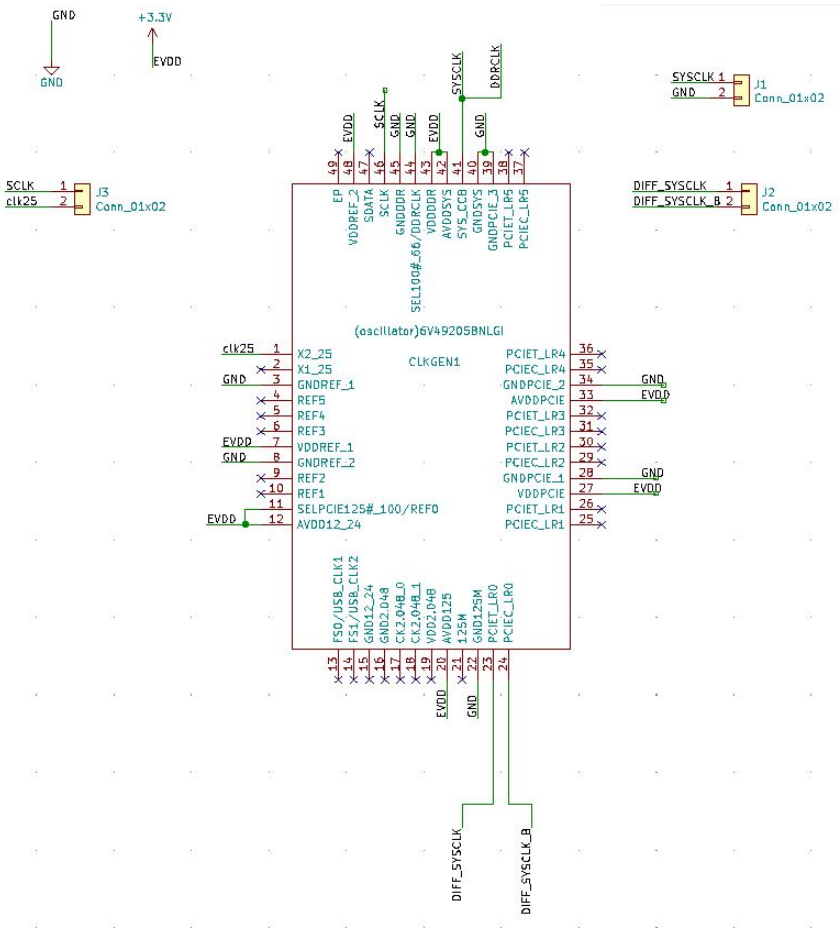

Figure 4: Rough draft of our power board design

Zach also selected all of the parts for our project, and Sam searched for KiCAD models of each of them, which we then used in our designs. Suman found the reference boards that we used, researched Code Warrior and how to use it, obtained most (if not all) of the datasheets for our parts, and (most importantly) wasn't afraid to ask questions when something was unclear.

5. References

[1]-"CodeWarrior Development Studio for QorIQ LS series for ARM v8 ISA," Creating, building bareboard project. [Online]. Available: https://docs.nxp.com/bundle/GUID-F97DCA91-E4C7-475C-B314-D15C603BAA10/page/GUID-5A152069-64A2-43A9-A2F5-A1BCE9B02155.html. [Accessed: 17-Apr-2020].

[2]-"Layerscape LS1012A Reference Design Board," NXP. [Online]. Available: https://www.nxp.com/design/qoriq-developer-resources/layerscape-ls1012a-reference-design-board:LS1012A-RDB. [Accessed: 17-Apr-2020].

[3]-“QorIQ SDK v2.0-1703 Documentation,” Recover system using CodeWarrior Flash Programmer. [Online]. Available: https://docs.nxp.com/bundle/GUID-39A0A446-70E5-4ED7-A580-E7508B61A5F1/page/GUID-94A3AC6E-2F0F-40B7-B4FE-C9819BA63A7C.html#GUID-6E44B664-2DFD-4695-9801-701AD8CB0B9D. [Accessed: 17-Apr-2020].

[4]-“QorIQ® Layerscape 1012A Low Power Communication Processor,” NXP. [Online]. Available: https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/layerscape-communication-process/qoriq-layerscape-1012a-low-power-communication-processor:LS1012A?tab=Documentation_Tab. [Accessed: 17-Apr-2020].

[5]-“IEEE Code of Ethics,” IEEE. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 23-Feb-2020].

[6]-“QorIQ® Layerscape 1046A and 1026A Multicore Communications Processors,” NXP. [Online]. Available: https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/layerscape-communication-process/qoriq-layerscape-1046a-and-1026a-multicore-communications-processors:LS1046A?&tab=Documentation_Tab. [Accessed: 01-Mar-2020].

[7]-“LS1046A Freeway Board,” NXP. [Online]. Available: https://www.nxp.com/design/qoriq-developer-resources/ls1046a-freeway-board:FRWY-LS1046A. [Accessed: 01-Mar-2020].

[8]-“QorIQ® LS1046A Development Board,” NXP. [Online]. Available: https://www.nxp.com/design/qoriq-developer-resources/qoriq-ls1046a-development-board:LS1046A-RDB. [Accessed: 01-Mar-2020].