BEE 495 – Capstone I

High-Level Redesign

Rathtana Duong | Miguel Huerta | Wesley Lai | Pooja Ravi

DESIGN DOCUMENT

3/11/2018
Rev. B

Introduction

The Elcano High-Level system is responsible for setting a desired vehicle speed and direction based on localization, pathfinding, and obstacle detection. The Low-Level system then has the job of making the vehicle execute the desired commands. At present, the High-Level system is partitioned among several processors:

- C7 (*Raspberry Pi*) processes visual information
- C6 (*Arduino Mega*) localization based on GPS, speed, and INU
- C5 (*Arduino Micro*) Lidar/sonar obstacle detection
- C4 (*Arduino Mega*) computes a path from digital map
- C3 (*Arduino Micro*) processes all data and sends instructions to Low-Level

The High-Level Redesign project seeks to port the C6, C4, and C3 processors to a single **Arduino DUE**. The new processor is designated as the C643 processor and has the following features:

- Elcano Low-Level Backwards Compatibility
- Simplified Serial Communication
- Single Processor Operation
- Arduino IDE Friendly
- Smaller Footprint
- CAN Bus Support
- 32-bit processing

Theory of Operations

The existing High-Level system has C4, C3, and C6 implemented on three different Arduino units. The C4 (mapping) and C6 (localization) are each implemented on separate Arduino Mega whereas the C3 (pilot) is implemented on an Arduino Micro.

C4 and C3 Theory of Operation

The C4 computes the route to the destination and sends that information to C3(pilot). The C3, in turn, transmits the speed of the wheel to the lower level system (i.e., C2). In addition, C5, implemented using an Arduino Micro, does obstacle detection with one version using the sonar and the other version using the lidar. This information is also sent to C3(pilot) so that the speed of the wheel could be changed accordingly. The C4 Arduino uses an SD card shield which is used to store digital maps on it so that C4 can read and compute the path to the destination (an orange cone). The C6(localization) gives the coordinates of the present location in terms of distance north and distance south as the vehicle covers a certain distance to C4.

C6 Theory of Operation

The C6(localization) uses a GPS shield which is used to give the coordinates of the current location using satellites. It also uses the Optical mouse for visual odometry which also gives an estimate of the current location. This is more accurate than the estimate of the current location that the cyclometer can give alone since the optical mouse uses a 2-dimensional axis. Therefore, the vehicle uses the estimate from the GPS, optical mouse and the cyclometer to get a better estimate of the current location. The compass feeds the information of which direction is north or south using its 3-dimensional axis and magnetic field. The C7 processes the visual information i.e., it computes the distance to the cone and the probability of a cone being present at any given location to C6 which then causes the behavior of the vehicle to change according to whether or not there is a cone at that specific distance.

Re-design of the existing system

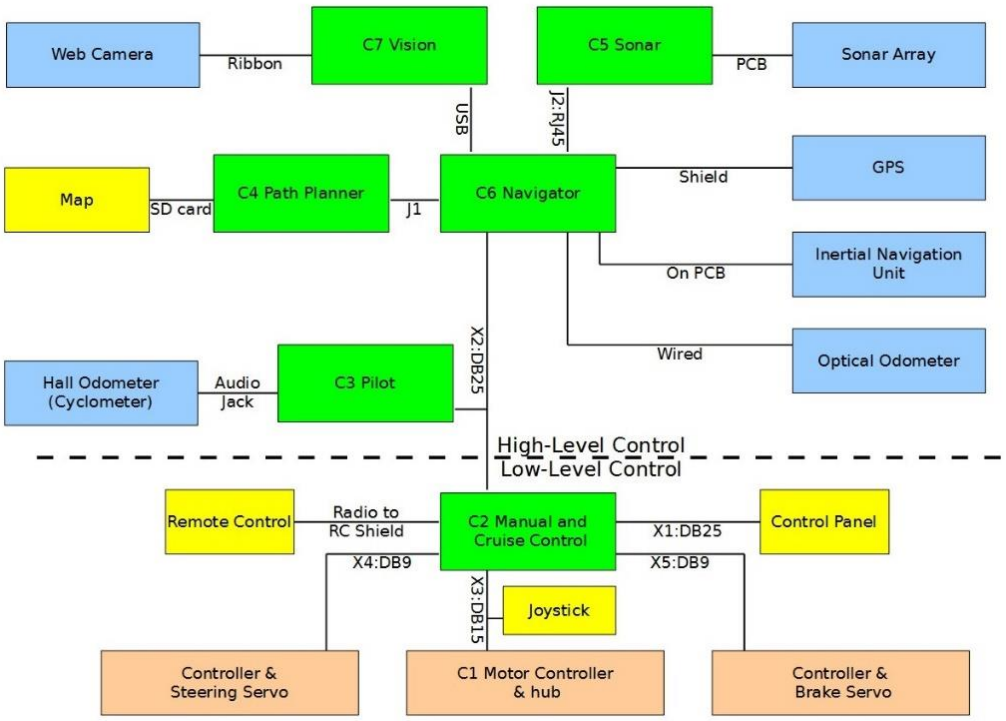
The processors currently use a 16 MHz clock. Therefore, the proposed re-design of the High-Level system targets to port the three processors i.e., the C3, C4, and C6 onto a single processor (Arduino Due) which has a clock rate of 84 MHz. This way, the space for the boards is optimized and the processors can run a lot faster. The system at present uses a UART type communication but is a lot slower since the current implementation uses three different processors. By porting the processors, the proposed new design uses software to handle the running of processors for a shorter time thereby computing the route at shorter intervals of time without having to compute the entire distance from the beginning until the destination.

System Block Diagrams

This section describes the hardware and the software blocks that make up the Elcano Project's system.

The overall system of the Elcano Project is illustrated in Figure 1. The Low-Level system controls the motion of the trike. This includes the steering, throttle, and braking system. The High-Level Redesign team is more interested in the High-Level system with a focus on the C3, C4, and C6 processors along with their peripherals. The High-Level Redesign team has no plans to modify the Low-Level system.

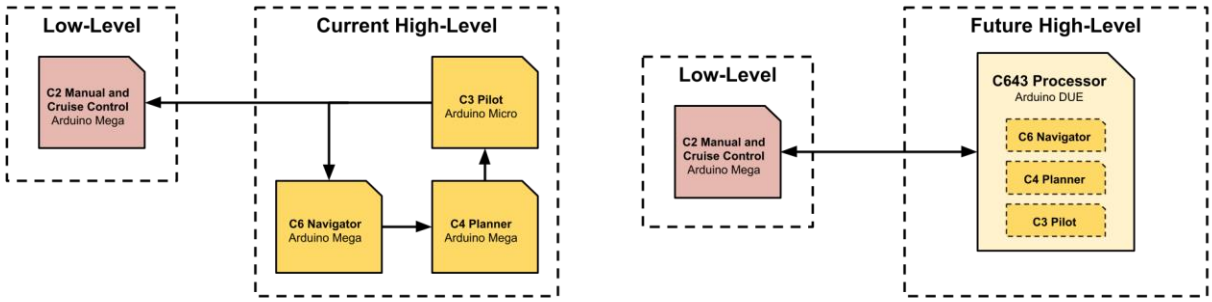
Figure 1. Current system for the Elcano Project



Software Blocks

The current and future software blocks are illustrated in Figure 2. The left diagram is the current software communication between the processors and the right diagram is the future communication.

Figure 2. Current and Future Software blocks



The current communication between the processors involves a UART serial loop from C6 → C4 → C3 → C2 → C6. The goal of the Elcano Serial library is to implement this system. At present, this method of communication is not fully functional. The future design combines

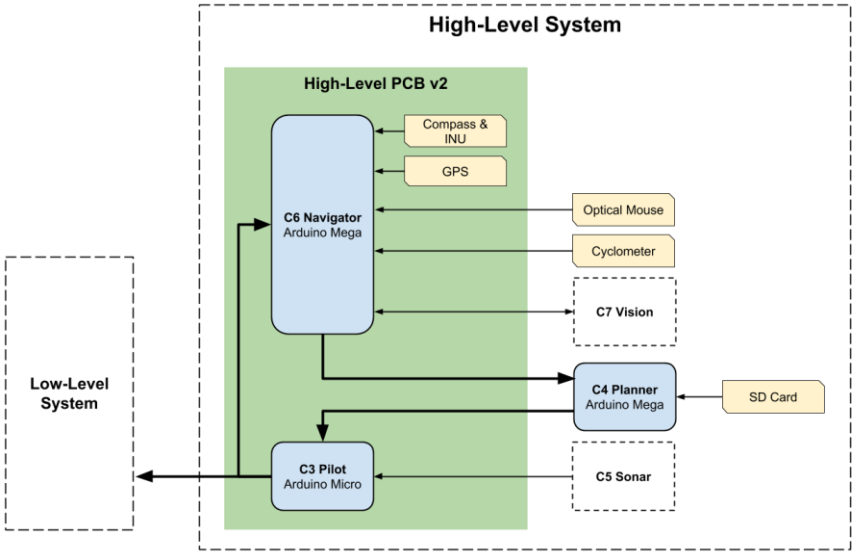
the C3, C4, and C6 system so standard serial communication can take place between the C643 processor and the C2 system.

The future design needs to implement a system that allocates enough resources to run the task of the three replaced processors. The simplest implementation would be to loop each replaced processor sequentially in a forever loop. This is not ideal since one processor may take too long to run. Instead, a scheduler system is needed to pause tasks that are taking too long in order to run a more time-sensitive task.

Hardware blocks

The current High-Level system is outlined in Figure 3. The current High-Level PCB contains headers to mount the C6 Arduino Mega, C3 Arduino Micro, compass & INU sensor, and GPS shield. Additional headers are used to interface with the C4 Arduino Mega, C7 Raspberry Pi, and an optical mouse sensor. These units are not mounted on the PCB and are interfaced with the use of wires. The SD card shield is mounted to C4. The cyclometer interfaces with C6 through an audio jack. Two RJ-45 ports are mounted on the PCB to connect to C5. A 25-pin D-sub connector is used to connect the High-Level PCB with the Low-Level PCB.

Figure 3. Current Elcano High-Level system

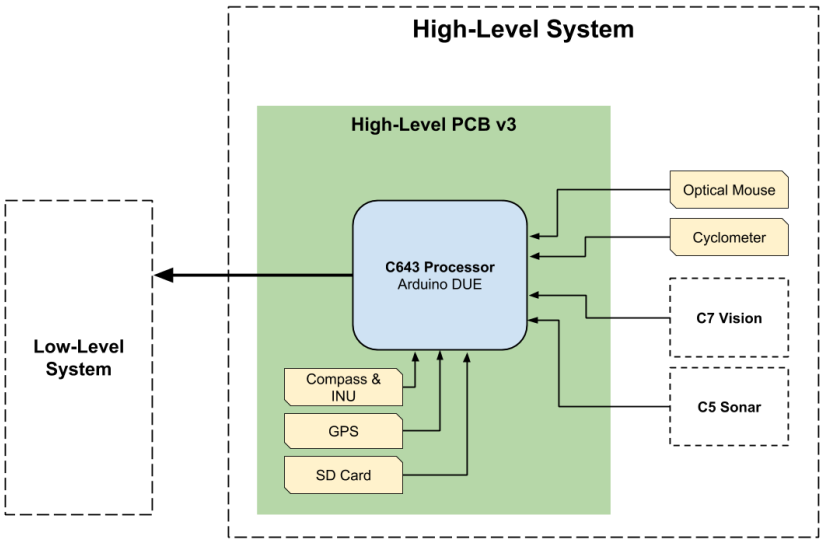


The future High-Level system is outlined in Figure 4. The future High-Level PCB will keep the current connectors to the Low-Level system, C5 system, C7 system, optical mouse, and cyclometer. In addition, the compass sensor and main processor will remain mounted to the PCB via headers.

The integration of the C3, C4, and C6 processors bring some hardware changes. Some peripherals are not 3.3V compatible, so logic level shifters ICs will be added to the PCB to allow 5V peripherals to interface with 3.3V logic. The current GPS shield is not plug-and-play

compatible with the Arduino DUE, so a standalone GPS module will act as a replacement. A standalone SD card reader will also replace the current SD card shield. Additional headers will be added to allow for future use of the CAN bus. The overall High-Level PCB dimensions and form factor will be as similar to the current PCB as possible to remain compatible with other parts of the Elcano Project.

Figure 4. Future Elcano High-Level system



Software design for Scheduler

The overall goal of scheduling is to give each processor enough time to run its program. In this case, we want to be able to run C3, C4, and C6. Part of our design is to test each processor and determine the time it takes to complete the task. Therefore, we want to determine the maximum time that C6, C4, and C3 need to complete its program before moving on. Once we determine that time, we want to run them in the correct order. According to our plan, we want C6 to run completely first, then C4 to run completely but only one time through until we pick a new destination, then run C3 completely and repeat the whole process. As can be noticed, C6 and C3 will run regularly, but C4 will only run if a new destination is picked. Although C4 will take the most time to run we're only running C4 when the destination changes, therefore, it won't need to run every time. Finally, it might be helpful to look into multitasking if our program requires more time to run, but because the Due is about 5 -16 times faster than the Mega, this scheduling system shouldn't be a problem with C4 only running in special cases. This means that we are not doing preemptive scheduling; instead, we assume that C3 and C6 will have enough time to run whereas C4 will only be run when the destination is changed.

Peripherals

The system contains 5 peripherals that act as sensors for the system and they connect as shields to the Arduino DUE. They are Compass, Cyclometer, GPS, Optical sensor and SD card reader\writer. For this project, 3.3V compatible devices are needed. A logic level shifter will be used in the case the shield/sensor is not 3.3v compatible.

Compass

The current compass is powered by an LSM303 IC. The purpose of this unit is to assist the GPS with navigation. This unit is 3.3V safe and can be connected directly to the Arduino Due without fear of damaging the unit. The communication type is I2C.

3.3V safe	Yes
Communication type	I2C

Compass pin mapping

Sensor Pin	Direction	DUE Pin	Notes
3V3	>	-	Not connected
Vin	<	3V3	Connected with jumper
GND	<>	GND	
DRDY	>	26	Data ready; different from previous
I1	>	30	Sensor generated Interrupt; different from previous
I2	>	28	Sensor generated Interrupt
SDA	<>	20	I2C
SCL	<	21	I2C

Triple-Axis Gyroscope

The gyro is based on the L3GD20H. Combined with the compass, this gyro can act as an inertial measurement unit (IMU). This unit is 3.3V safe and can be connected directly to the Arduino Due without fear of damaging the unit. The communication type is I2C or SPI.

3.3V safe	Yes
Communication type	I2C or SPI

Gyro pin mapping

Sensor Pin	Direction	DUE Pin	Notes
SCL	<	21	I2C or SPI
SDA	<>	20	I2C or SPI
SA0			SPI
CS	<		SPI
DRDY	>	41	Data ready; not connected
INT1	>	43	Motion wake-up
GND	<>	GND	
3Vo	>		Voltage from regulator
Vin	<	5V	Power supply

Cyclometer

The cyclometer model is Cateye Velo 9. The current system does not actually use the data collected by the cyclometer, and because of this, it can be removed. The way it operates is that there is a reed connected directly to the Arduino, and in the wheel, there is a magnet that rotates with the spinning of the wheels. As the magnet passes the reed a rising edge is created, which then the Arduino picks up as a logical high. The Arduino then counts the rising edges and acts as a cyclometer. The way it's currently set up is highly inaccurate. If the processor is tied up with a different operation, then a rising edge might be missed. An interrupt could be created, however that might not be very safe since a more important task could be interrupted. It might be more desirable to instead obtain the speed from the GPS unit. Source of operation http://www.elcanoproject.org/tutorial/lab3.php

3.3V safe	Yes, connect to a pull-up resistor
Communication type	Rising Edges

Cyclometer pin mapping

Sensor Pin	Direction	DUE Pin	Other	Notes
Click	>	2	X2 21	Needs pull-up to 3.3V

GPS module

The proposed GPS unit is an Adafruit Ultimate GPS Breakout. It is used for basic location finding and is used for navigation together with the compass. The communication type is a serial connection. This unit is 3.3V safe.

3.3V safe	Yes
Communication type	UART, Serial connection

GPS module pin mapping

Sensor Pin	Direction	DUE Pin	Notes

3.3V	>	-	Not connected
EN	<	-	Used to turn off the GPS. Not connected to DUE
VBAT	<	-	For secondary power. Not connected to DUE
FIX	>	24	Determines if there is GPS acquisition
TX	>	14	DUE's serial 3 line
RX	<	15	DUE's serial 3 line
GND	<>	GND	
VIN	<	5V	
PPS	>	9	For syncing the GPS with the host

Optical Sensor

The optical mouse (ADNS3080) is used for visual odometry. It detects changes in the road from which the speed can be computed. The software for this has not yet been implemented.

3.3V safe	Yes
Communication type	SPI

Optical sensor pin mapping

Sensor Pin	Direction	DUE Pin	Other	Notes
GND	<>	GND		
5V	<	5V		
3V	>	-		Not connected
NCS	<	4		Chip select
MISO	>	ICSP-1		
MOSI	<	ICSP-4		
SCLK	<	ICSP-3		
RST	<	RESET		System-wide reset
NPD	<	48		For power-saving mode
LED	<	49	X2 2	

MicroSD Card Breakout Reader

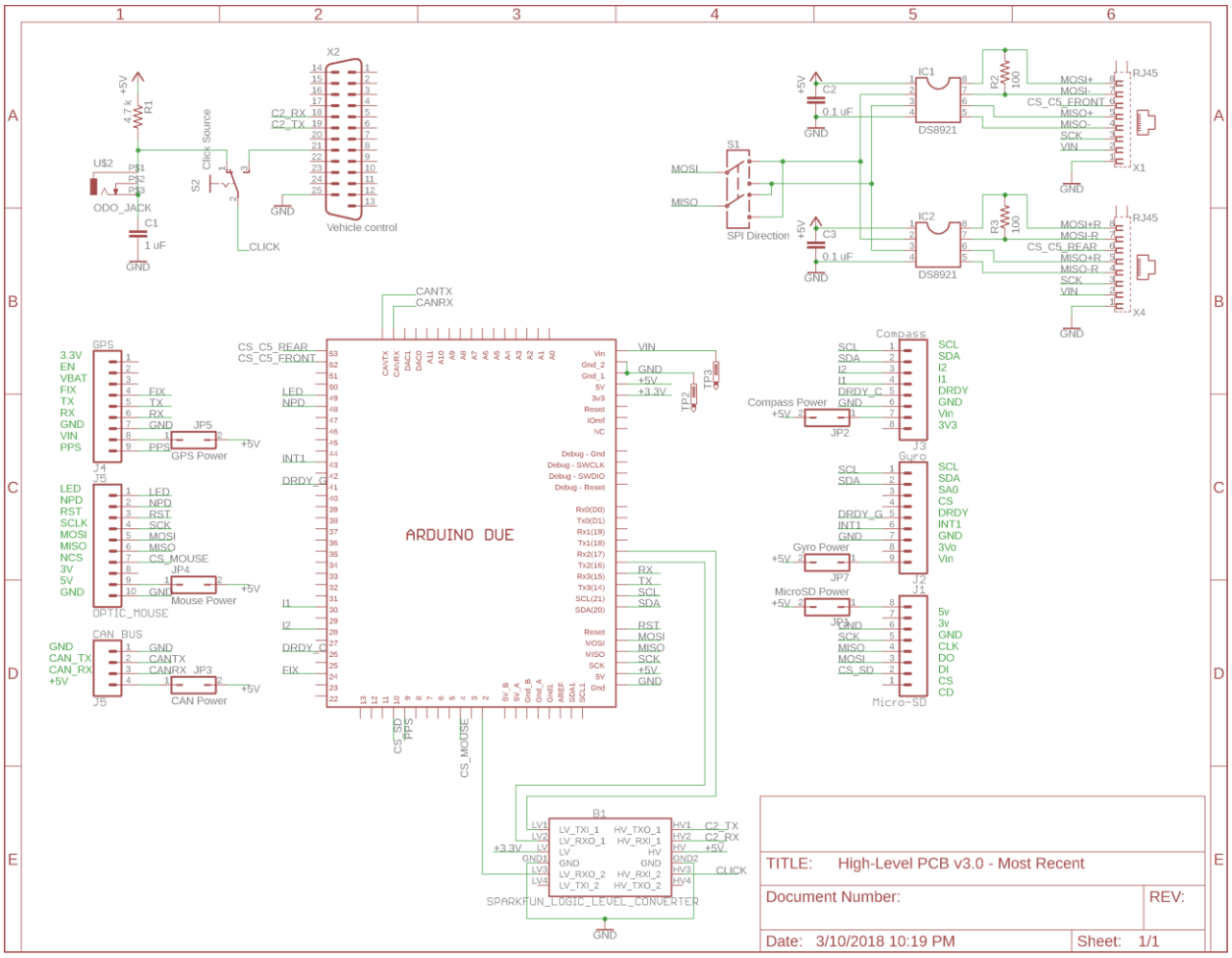
The Adafruit MicroSD card breakout board+ is used to hold the map and other storage needs. This unit is 3.3V safe and uses SPI.

3.3V safe	Yes
Communication type	SPI

MicroSD card breakout pin mapping

Device Pin	Direction	DUE Pin	Notes
CD	>	7	Detects card removal
CS	<	10	Default pin in code
DI	<	ICSP-4	MOSI
DO	>	ICSP-1	MISO
CLK	<	ICSP-3	SCK
GND	<>	GND	
3v		-	Not connected
5v	<	5V	

Schematic for the NEW high-level system*



*For a clearer image of the schematic, please click here

Test plan

Assumptions and dependencies

The High-Level Redesign project aims to port the current High-Level system. Therefore, some assumptions must be made. The team is assuming a partially functioning High-Level system. This means that the C6 processor and the C4 processor should each be operational on an Arduino Mega. At the time of writing, C3 is not operational, so testing cannot be performed on the Elcano Pilot system.

Some sensors are for future development and the necessary sensor code may not be implemented yet. For example, the C6 system has an interface to an optical mouse for visual odometry, but no code is in place to take advantage of it.

Testing on the physical Elcano trike is postponed while issues with the Low-Level system are being resolved.

Approach

This section goes over the steps to be performed for testing.

Verification of the CURRENT High-Level system

Verification of the current system is important to ensure that functional systems are getting ported.

Individual testing

1) C4 Individual Testing
   a) Should be able to select the correct map from the GPS
   b) Able to load a map from the SD Card
   c) Able to create and find/build the shortest path from starting to ending point using the A-star algorithm
   d) We can verify if the paths are correct by looking at the checkpoint that it took on our actual map that it took
2) C6 Individual Testing
   a) Verify the current location from the GPS
   b) Able to find the position from the odometer and compass
   c) Using the information above to estimate your current position
   d) Determine the range of percent error by using the standard deviation and mean.
   e) As of now, we will assume that the estimate percent error should <= 10% for estimating your current position. We can measure this as follows: the GPS used shows a 3 meters difference between what it might note and the actual position. Therefore, for the current system, we could measure this by keeping the vehicle at rest and reading the position from the GPS and the actual position from google maps. Then subtract the two values and square those to get the standard deviation. When the vehicle is moving, we could collect the series of coordinates as given by the GPS and then calculate the error by calculating the standard deviation as mentioned above. Note: This percent error will be lower once we start experimenting and testing.

Testing of the NEW High-Level system

Peripheral testing

1) Adafruit LSM303 compass

a) Wire up the sensor according to Adafruit's tutorial
b) Load example code to Arduino DUE
c) Use the serial monitor to check that sensor is functioning

2) Repeat for GPS and microSD modules

Software testing of C643

1) Test of timing on multitasking
   a) Set a digital line to high when running the C6 processor
   b) Set that line to low when C6 is finished running
   c) Set a digital line to high when running the C4 processor
   d) Set that line to low when C4 is finished running
   e) Set a digital line to high when running the C3 processor
   f) Set that line to low when C3 is finished running
   g) Come up with an estimate on the timing for each processor
2) Log time into the routines and study how this varies for each processor and make changes to the code accordingly.

High-Level PCB v3

1) Pre-fabrication
   a) Print PCB on paper using 1:1 scaling
   b) Overlay components, sensors, and other peripherals that will be mounted
      i) Note any locations where there may be mounting problems
      ii) Note the drilling hole sizes
      iii) Check if silkscreen labels remain visible with components overlaid
2) Post-fabrication (build 4 PCBs)
   a) Unpopulated board
      i) Measure resistance between power rails and ground
         Expect to read infinite
      ii) Check for continuity between power rail and connector supply points
   b) Populated board
      i) Note any poor solder joints
      ii) Test for continuity between traces
   c) Power-on
      i) Remove power jumpers to peripheral devices
      ii) Use a current limited power supply to slowly ramp voltage supply to 9.6V
      iii) Verify voltage on DUE's regulator steps down the voltage to 5V and 3.3V rails
      iv) Replace power jumpers individually with peripheral devices
      v) If problems persist at this point, try another PCB to see if problems can be replicated
         (1) Record any improvements or changes to be made in the next batch of PCBs

Item pass/fail criteria

This section goes over the criteria that will be used to determine whether each test item has passed or failed testing.

Hardware:

Sensors and peripherals

Sensors and peripherals that were functional on the old system were powered from 5V. The Arduino DUE runs on 3.3V so voltage levels must be considered to protect the lower voltage processor.

Voltage levels on input connections to the DUE's I/O pins will be checked for voltage compliance. A logic high input between 2.5V and 3.5V will be a "test pass" result while any higher voltage than 3.5V will be a "test fail" scenario.

High-Level PCB v3

The High-Level PCB v3 will be subjected to physical dimension compliance testing before fabrication. In other words, do the peripherals and connectors fit on the PCB. A "test pass" means components fit as intended, otherwise it would result in a "test fail".

Board layout test determines correct routing of PCB traces. A "test pass" results from traces connecting the correct components. A "test fail" means wrong trace connection and will need external modification such as trace cutting or adding jumper wires.

Software:

C4 Individual Testing

C4 should be able to able to select the correct map from the GPS and be able to load the map from the SD card. If so, the individual testing on C4 is complete and is designated as "test pass" else it would be "test fail" in which case the code would have to be changed to enable reading back from SD card.

C6 Individual Testing

The Arduino Due should be connected to the odometer and compass with proper voltage conversion circuits. The circuit, when connected to the stick wheel and taken outdoors, should give proper coordinates of the route covered. This can be verified by creating an excel chart based on the coordinates read from the SD card and comparing it to the actual route covered to see whether there is both are the same. In this case, if there isn't a match or if there are no coordinates of the route covered, then this would be a "test fail" which would mean that the circuit and the code need to be checked again to identify the issues otherwise it is a "test pass".

Combined Testing of C643

The first step would be to give a chance for each of the processors to run. A digital line would be set to high when each of the processors is running and then set low once that processor has completed running. From this, an estimate of the timing could be made. If this is the behavior seen when monitoring the digital lines, then it is a "test pass" else it is a "test fail" in which case, the code would need to be handled differently to tackle the problem of timing.

Test environment

In the absence of a working low-level system, a person can emulate the low-level system. The High-Level system can be mounted on a wheeled platform. A person can then push the platform and the high-level will give commands, which then a person can interpret as a low-level system would. With the odometer and the compass connected to the Arduino Due, the C6 can be tested using a stick on wheel when taken outdoors to verify whether it can come up with the correct coordinates of the route covered and whether the flow of information between C4 and C6 is as what is expected.

Debugging data from the High-Level system can be displayed through the Arduino serial monitor with a laptop connected to the Arduino DUE.

Risks

There is an infinite number of scenarios that can occur. Below are the six most likely risks facing the Elcano High-Level Redesign project.

- **Risk 1**: Fewer Arduino DUE suppliers since DUE is becoming obsolete
  Probability: 75%
  Impact: Need to look at non-genuine Arduinos that may have slight changes
  Mitigation plan: Purchase clones or switch to a similar microcontroller such as the chipKIT Max32

- **Risk 2**: Optical mouse sensor is discontinued. May be in short supply
  Probability: 75%
  Impact: Cost for optical mouse sensor may increase
  Mitigation plan: Many clones of the ADNS3080 exist on sites such as eBay. The predecessor to the Optical Flow sensor is the PX4FLOW

- **Risk 3**: First batch of PCB not good
  Probability: 50%
  Impact: Traces might need to be cut, and jumpers might need to be added
  Mitigation plan: Triple-check PCB design before sending to fabrication. If the PCB does have errors, a new PCB will be made

- **Risk 4**: C3 does not work properly
  Probability: 80%
  Impact: Improper information passed onto the lower system which would mean that the entire system would not work as it should
  Mitigation plan: write minimal placeholder routine for C3

- **Risk 5**: Incorrect data passed between C4 and C6
  Probability: 50%
  Impact: The current position as noted by the GPS would not be the actual position which would mean that following the route from C6 would not help reach the actual destination
  Mitigation plan: Pass dummy values and verify whether the same values as passed it are received.

- **Risk 6**: Arduino DUE does not have enough processing power to handle all three past processor tasks
  Probability: 10%
  Impact: A single Arduino DUE would not be sufficient to control the High-Level system
  Mitigation plan: Have an Arduino Mega as backups in case some tasks need to be moved to the Mega. Run a system that has both DUE and Mega